Expanding the Wireless Communication Paradigm

by

Craig Music
B.S. Computer Science
Massachusetts Institute of Technology, 2001

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science
at the
Massachusetts Institute of Technology

June, 2002

Signatures of Authors_____
Department of Electrical Engineering and Computer Science
May 17, 2002

Certified by_____
Dr. James E. Hicks, Jr.
Principal Member of the Technical Staff, HP Labs
Research Affiliate, MIT

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Expanding the Wireless Communication Paradigm

by

Craig Music
B.S. Computer Science
Massachusetts Institute of Technology, 2001

Submitted to the
Department of Electrical Engineering and Computer Science

June, 2002

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

As portable computers become more pervasive within our culture and wireless technologies become prevalent, it is natural to explore how to expand the ways in which these portable systems can help users in their daily lives. We determined to build a system that would exemplify just how extra information, in this case in the form of location acquisition, could be used by a portable computer to make predictions and decisions which would provide as much smartness as possible in order to be an aid to its user. Using an iPAQ 3670 handheld, wireless 802.11b connectivity and the Cricket Location System deployed at MIT's Lab for Computer Science, we built a software system similar to a scheduler application that would constantly monitor its location and be able to actively make decisions based on its current location and the location of upcoming activities. The resulting system was successfully able to prevent presentation interruptions, advise the user on reminders based solely on location as well as actively alerting of upcoming meetings that the user would have been otherwise late to without the destination location based reminder. This work thus opens the path for more development into systems that taking advantage of location information can simplify the everyday life of users as well as raising questions and issues that future systems will have to deal with.

Thesis Supervisor:
Dr. James E. Hicks, Jr.
Principal Member of the Technical Staff, HP Labs
Research Affiliate, MIT

Table of Contents

I. Introduction

This thesis concentrates on exploring new ways to empower personal digital assistants (PDA's) that have wireless connectivity and knowledge of their user's location in order to expand an individual's communication options. Such added functionality can range from interacting with other people taking advantage of the added location information to modifying the behavior of simple applications depending on the location of its users.

Through collaboration with ongoing research at the MIT Lab for Computer Science's (LCS) Project Oxygen and Compaq's Cambridge Research Lab facility, we developed solutions that can potentially change the paradigm with which people see the interaction of computers and human beings. The project specifically focuses on "enabling people to do more by doing less, that is, to accomplish more with less work" [4].

Specifically, we combined Compaq's iPAQ H3670 handheld device with different wireless connectivity technologies and location retrieval services to present a demonstration of how its use can affect the lives of users in the near future. With this project, we hope to contribute to Project Oxygen's goal of bringing "abundant computation and communication, as pervasive and free as air, naturally into people's lives" [4].

I.1 Project Oxygen at MIT

As described in its mission statement, Project Oxygen has as its vision to look into the future when human beings will not interact with computers through our standard keyboards and mice [4]. Instead, computers will be accessible everywhere "like batteries

and power sockets, or oxygen in the air we breathe" [4]. The project is spearheaded by LCS with corporate sponsorship from Acer, Delta Electronics, Inc., Hewlett-Packard Corp, Nippon Telegraph and Telephone, Inc., Nokia Research Center, Philips Research, the US Government Information Technology Office (ITO), and the Department of Defense Advanced Research Projects Agency (DARPA). As described in its mission statement, the four goals of the project are to develop a system that is:

- *pervasive*--it must be everywhere, with every portal reaching into the same information base;
- *embedded*--it must live in our world, sensing and affecting it;
- *nomadic*--its users and computations must be free to move around according to their needs;
- *eternal*--it must never shut down or reboot; components may come and go in response to demand, errors, and upgrades, but Oxygen as a whole must be non-stop and forever.

"The Oxygen software environment is built to support change, which is inevitable if Oxygen is to provide a system that is adaptable, let alone eternal. Change is occasioned by anonymous devices customizing to users, by explicit user requests, by the needs of applications and their components, by current operating conditions, by the availability of new software and upgrades, by failures, or by any number of other causes" [4].

Within the Oxygen framework, speech and vision replace the commonplace keyboard and mouse. As mentioned before, the goal is to make the interaction with computation resources much more fluid and natural, for example, instead of logging into a machine and navigating through menus, a simple voice command such as "make a hardcopy of this document quickly" would print the current document being worked on.

Naturally, the goals of project oxygen require massive collaboration between numerous software agents, hardware resources and mobile and static elements. As mentioned in its mission statement, these technologies are grouped into:

- *Automation technologies*, which offer natural, easy-to-use, customizable, and adaptive mechanisms for automating and tuning repetitive information and control tasks. For example, they allow users to create scripts that control devices such as doors or heating systems according to their tastes.

- *Collaboration technologies*, which enable the formation of spontaneous collaborative regions that accommodate the needs of highly mobile people and computations. They also provide support for recording and archiving speech and video fragments from meetings, and for linking these fragments to issues, summaries, keywords, and annotations.

- *Knowledge access technologies*, which offer greatly improved access to information, customized to the needs of people, applications, and software systems. They allow users to access their own knowledge bases, the knowledge bases of friends and associates, and those on the web. They facilitate this access through semantic connection nets.

Within the Automation Technologies group, researchers are trying to apply location technologies in order to increase the system's knowledge of its physical context. One of the endeavors followed by this group is to deploy an indoor location solution called the Cricket Location System that will attempt to empower Oxygen to act upon the extra knowledge of location. Clearly, this is a response to the lacking functionality that GPS presents in an indoors scenario.

This indoor location element is a key ingredient to the development of this thesis and a more detailed explanation of how it will be utilized and its specifications will follow in subsequent chapters.

I.2 Handhelds.org Initiative

Handhelds.org is an open source software initiative aimed at "encouraging and facilitating the creation of open software solutions for use on handheld computing platforms" [2]. Sponsored by Compaq Computer Corporation, this group tries to support

a broad range of individuals from power system developers to end users who are interested in fomenting the use of handheld devices with open source platforms.

Following with the open source motto, the group strives to support as many combinations of Unix-like operating systems and hardware as possible as long as they are considered mobile. The backbone of the help provided by this group can be summarized by three key points:

- Information: The group makes available numerous how-to's, faq's, guides and other knowledge bases to help all kinds of users in their work with handheld devices.
- Communication: The website also strives to coordinate project teams and users with common interests that otherwise would have a harder time contacting each other
- Software Archiving and Distribution: The website also provides easy access to archiving servers that are readily available to individuals and groups interested in developing applications for handheld platforms.

The development of our application was greatly benefited by these resources offered by Hanhelds.org. We utilized a version of the Linux operating system for ease of development reasons that will be explained in a later chapter. The information available to us from this site as well as the personal help from numerous participants was invaluable.

I.3 Goal of Project

We utilized an open source development environment combined with a mobile computation solution in order to demonstrate how an application can change its behavior in response to location and thus simplify and expand the user's interaction with the computer.

The platform of choice was the Compaq iPAQ PDA inasmuch as it has been chosen as the de-facto platform to develop mobile software within Oxygen. This is in part due to its robust hardware, expandability, and compatibility with a freely modifiable operating system, in this case a specific distribution of Linux that will be discussed in the next chapter.

The application that we developed is a Smart Reminder system that serves as a scheduler that is able to interact with its context by determining the location of its users. In this way, an individual can not only pre-program reminders, meetings and the such, but can also specify their location. This extra piece of information allows us to make the application much more robust and self-adjustable to the user.

By knowing the current location of the PDA and the intended location of meetings, we are able to actively change the way in which the application responds. First, when a meeting or activity reminder is ready to trigger its alarm, the current location of the PDA is cross checked with the intended location of the activity to avoid interrupting a presentation or otherwise causing an unnecessary distraction. Furthermore, when more added functionality is desired, location reminders compute the time required to travel from the PDA's current location to upcoming activities and trigger alarms if it determines the individual is running short of time to reach his or her destination. And finally, the user has the capability of inputting reminders that are based solely on location with no specific time. This comes in useful if the user would like to be reminded to talk to a colleague whenever he or she walks close to that person's office or any variation of this scenario.

I.4 Collaboration with Marc Bourget

Another important aspect of the development of this thesis is the fact that the software was developed closely with Marc Bourget. Together, we defined the system modules and the interactions of those modules. We also divided the coding work at the class level although often we worked on each other's classes in order to revise as well as comment and become familiarized with the whole workings of the system. We both worked together with our advisor Jamey Hicks and developed the outline for this document jointly. The final system is therefore as much his at is mine, however, the actual development of this document was done independently.

II. Location Technologies

II.1 Cricket Location System

The Cricket Location System is an initiative from the MIT LCS Network and Mobile Systems Group. As part of Project Oxygen, the NMS group has been working on developing a system that would allow the acquisition of a user's physical context information and consequently enhance the possible computational experience. In this precise project, we are focusing on using the capabilities of this system to discover indoor locations where the Global Positioning System is ill-suited due to lack of the required line-of-sight contact with its satellites.

Within its specifications, the Cricket Location system boasts of being easily deployable, scalable, and requiring little or no central management while preventing its use as a "Big Brother" or unwanted tracking system. This immediately addresses concerns of privacy and feasibility which are of principal importance. The Cricket System is even able to estimate an x-y-z position to "within a few centimeters and ... detect boundaries to within 2 feet" [6]. Notwithstanding, the actual accuracy

demonstrated in our tests is not as precise as described here but this will be discussed later. Therefore, the x-y-z coordinate tracking method was substitute by a spatial differentiation alternative.

The actual implementation of the Cricket Location Technology is based on "beacons" that are deployed throughout a designated area and then a "listener" device is able detect RF and ultrasound signals emitted by the beacons and estimate its position.

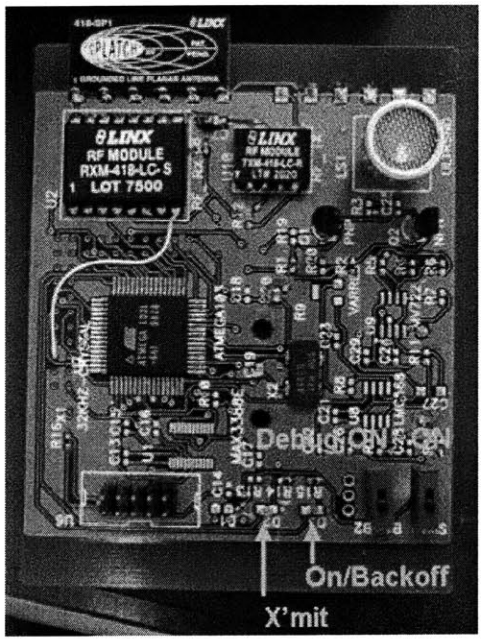

Figure 1: Cricket Beacon

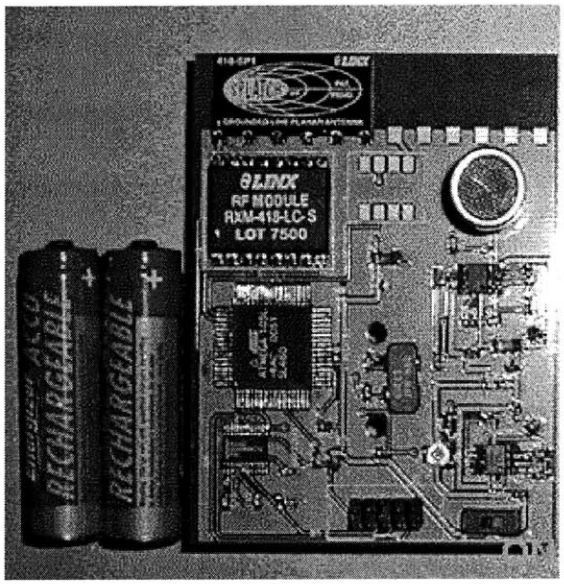

Figure 2: Cricket Listener

Each beacon emits an RF signal on the 418 MHz AM band at random time intervals with a concurrent ultrasound signal. In turn, each listener that receives this RF signal will wait to obtain the corresponding ultrasound signal and use the time at which it received each one to calculate its distance from the respective beacon. Specific algorithms are used to account for multiple beacons emitting signals simultaneously and ultrasound signals from different beacons getting confused with each other. A final, but crucial part of the system is that the RF signal sent by each beacon encodes information that specifically identifies it.

It should be clear then how the system protects the privacy of the user inasmuch as only the personal listener is able to determine the location by using distance estimates from multiple beacons. The user has complete control over whether this information is sent to a central database system or not. Moreover, since the beacons can detect if their RF signals interfere with each other, they use random exponential back-off algorithms so that adding more beacons is literally reduced to attaching them to the walls or ceilings and turning them on. No central system needs to be informed of any change. As mentioned before, the RF signal that the listener receives specifically identifies the beacon. Nonetheless, in order to match the beacon to a particular space, for example someone's office, it is practical to keep a matching database online and let the user's application fetch the necessary information. Again note that this by no means makes the user's location public.

The primary motivation behind this system is the identification of spaces or areas in which the user is located, e.g. rooms and lounges. However, it can also be used to determine the listener's precise position using x-y-z coordinates. This process however is more involved and requires measurements of the areas and configuration files that contain this data. In theory, triangulation from multiple beacons' readings pinpoints the location, although the practical implementation has met with problems that are still being dealt with. Although it was tested, this capability was not used in this thesis. More information about an attempt to utilize this capability for x-y-z positioning is available in Chapter V.3 about other experiences with the Cricket Location System.

II.2 IP Location System

As mentioned before, one of the major goals of this thesis has been to explore how a system can take advantage of having knowledge about its geographical context. The Cricket Location System is able to give precise information about the user's whereabouts as explained in the previous chapter, but only if it has already been deployed by placing Cricket beacons strategically throughout all the areas where location will be determined. We wanted to be able to go beyond this limitation inasmuch as Cricket beacons are only located in one building at MIT and one floor of CRL's building. Consequently we decided to take advantage of the fact the MIT has a different subnet for each building and therefore we could actually do a geographic mapping from IP to location. In this way, simply by comparing the system's current address to a database we can find out what building the user is located in. Clearly the usefulness of this alternative is limited since you do not have the accuracy of Cricket beacons; it is impossible to determine exactly in what room you are located. Notwithstanding, it is a great step beyond having no information and can be used for testing purposes.

II.3 GPS Location System

Although our main location discovery system serves our purposes fully while we are indoors, it is completely unable to function outdoors. For outdoor location purposes, the Global Positioning System is the best alternative to the Cricket Location System. It is widely known and therefore this description will not go into great depth. The GPS system is composed of a large network of 24 satellites and their 5 monitor ground stations located in Hawaii, Ascension Island, Diego Garcia, Kwajalein, and Colorado Springs.

This infrastructure put into place by the United State's military allows GPS receivers to obtain accurate positioning information to within a single meter or less, depending on whether the application is military or not.

The process followed to obtain these accurate readings is by triangulation using 4 distinct satellites and precise timing to know how far they are from the object in question. This however is not simple and requires multiple compensations for variations such as time synchronization, movement of the satellites and atmospheric conditions that will slow the signals sent from the satellites. The fact that we use 4 satellites instead of 3 comes from imperfect timing on the side of most GPS receivers but the measurement is corrected with this extra reading. The system has been widely successful and is now employed in multiple industries ranging from automotive to commercial air flight to weapon targeting.

III. Functionality of System

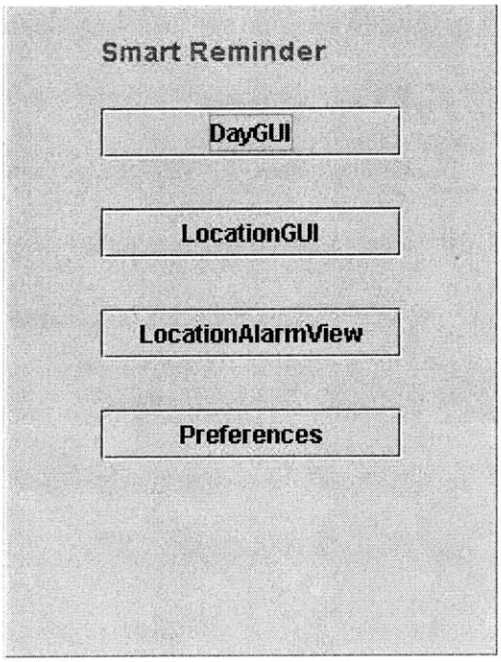

Figure 3: Main GUI

As briefly mentioned before, this system allows its user to input reminders in a scheduler type interface that also takes into account location. Secondly, the user can input reminders that trigger only based on location. These two reminders tie in with a set of interfaces that allow the user to customize locations and map them to different Cricket and IP labels as suits his or her needs.

III.1 Initial GUI

From the main GUI of the system, the user clicks on one of four buttons to access other GUI's that allow the input of time-based or location-based reminders. The second and fourth buttons bring up GUI's to manipulate the internal location database mappings and a simple interface to set multiple system settings.

III.2 Scheduled Reminders and Meeting Creation

This type of activity reminders are one of the two main options of our system. By calling on the DayGUI button of the main GUI the user is presented with a standard

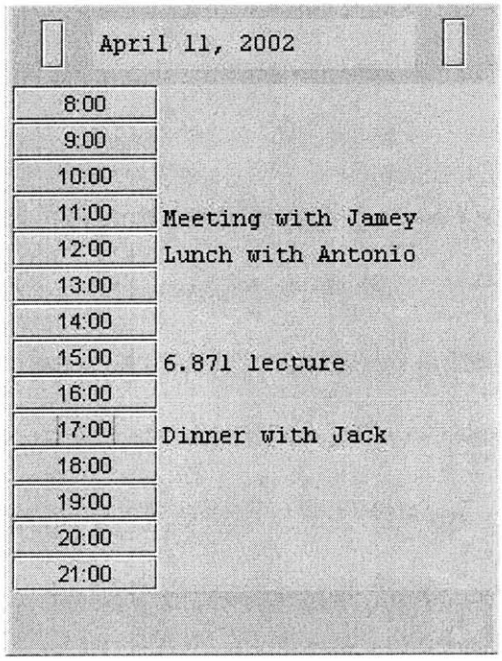

Figure 4: Calendar View

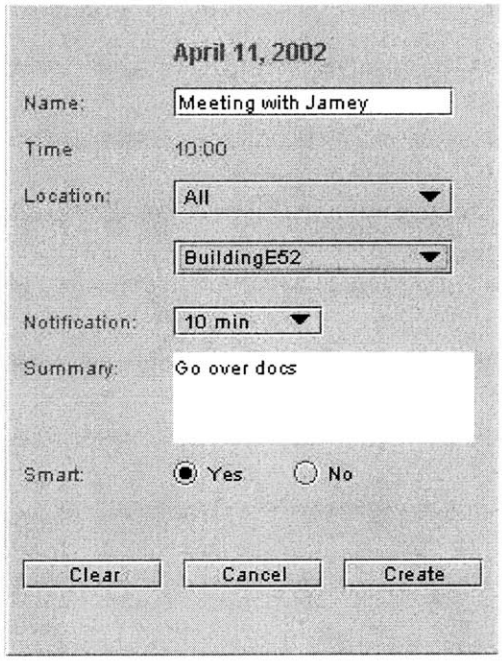

Figure 5: MeetingGUI

calendar style display, Figure 4. From here the user can easily scroll backwards and forwards using the two buttons at the right and left of the date. Furthermore, by clicking on any of the buttons that display a time of the day the user is presented with yet another GUI where he or she can create new meeting or else edit a pre-existing one.

Once the meeting creation GUI is displayed, refer to Figure 5, it is very straightforward to understand its functionality. Starting from top to button, the user can input a name for the meeting that will later be displayed in the calendar view. Below its name, the user has two scroll down displays where the location of the meeting can be selected. The top scroll down menu consists of the groups the system has in its database, each one is comprised of multiple individual locations. The user first selects the list from which the actual location is desired, and then the contents of the group are displayed in the lower scroll down menu. In this way, the user has the capability of easily filtering locations to find the desired one faster.

After the user has selected the location of this reminder, he has the option of inputting how much in advance of the actual activity the system should trigger a reminder. This is again a scroll down menu that ranges from 10 minutes to 1 day. A simple summary field follows in order to input yet more detail about the meeting, and finally but importantly the user has the option of selecting whether this should be a smart or a normal alarm. This is one of the most interesting features of our system. Once a smart alarm is created, it uses an estimation matrix to calculate the amount of time the user would take to travel from its current location to the meeting's location. If this amount is more than the time left until the meeting takes place, then an alarm is triggered informing the user he or she should start moving as soon as possible. Furthermore, this calculation is redone every time the system recognizes that the iPAQ has changed position.

III.3 Location Reminders

A second powerful ability of the system is to create reminders that are not linked to time at all. On the contrary, the user creates a reminder based solely on location and

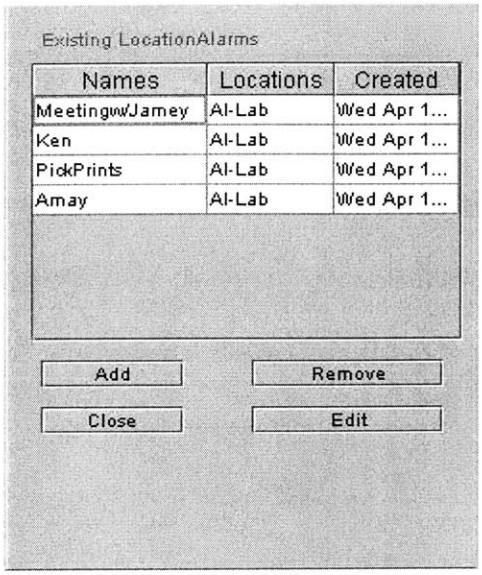

Figure 6: LocationAlarmViewer GUI

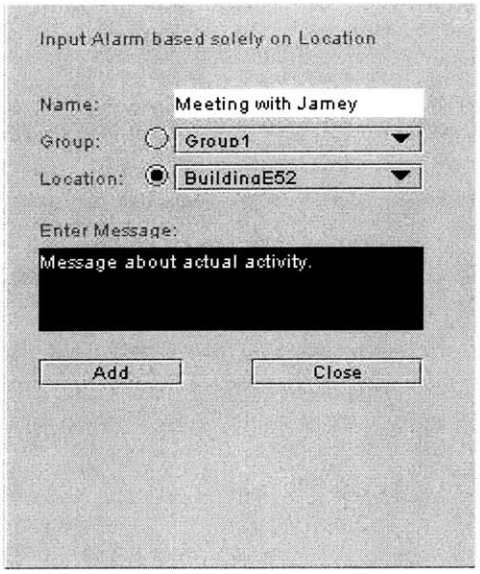

Figure 7: LocationAlarmGUI

every time the iPAQ is at that location a reminder GUI will be triggered. By pressing the third button on the main GUI, "LocationAlarmView", the user is presented with another GUI in which all the current LocationAlarms are displayed as well as providing buttons for creating, removing, or editing such reminders as shown in Figure 6.

The display clearly shows the name of each LocationAlarm in existence as well as its location and creation date. The user can then press the "add" button to enter the second display which allows him or her to create a new LocationAlarm as shown in Figure 7. This GUI simply allows for a particular name of the reminder as well as choosing between all the locations available in the lower scroll down menu and to chose form the existing groups in the upper scroll down menu. Finally, another field allows the

user to enter yet more detailed information about this reminder that will be displayed later in the actual alarm GUI when it is triggered.

III.4 Location Utilities

By pressing the second button from top to button on the main GUI shown in Figure 3, the user is presented with the LocationGUI in which he or she can see all the

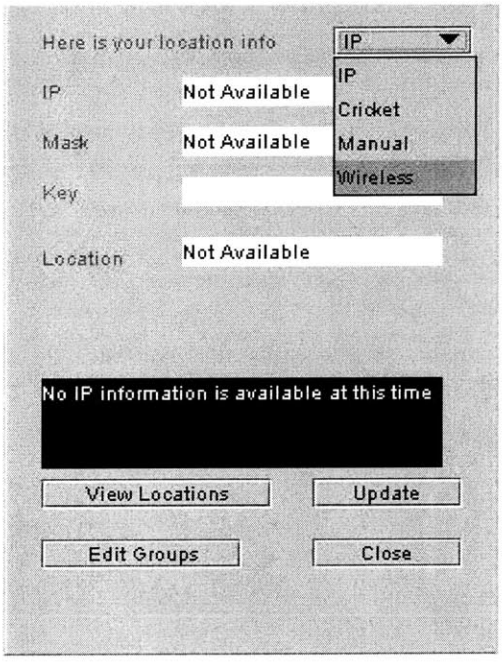

Figure 8: Location GUI

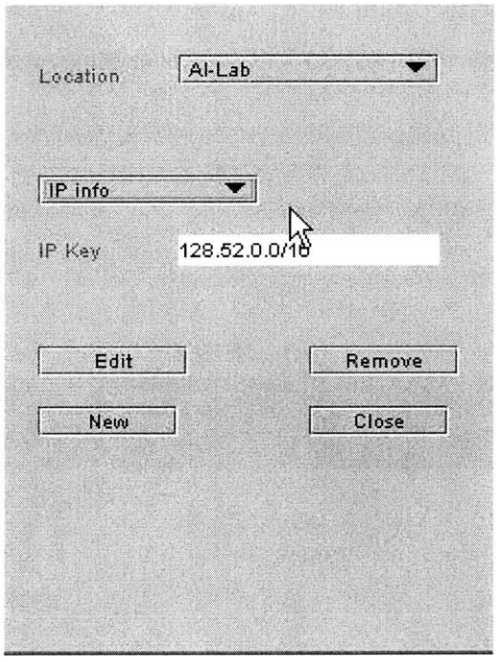

Figure 9: AllLoc GUI

information available about the iPAQ's current location. As shown on Figure 8, the user can select what form of location information is desired, be that IP, Crickets, manual selection and perhaps in the future wireless base station. If the particular location service is not available, the corresponding fields shows "Not Available", but if the system is indeed capable of obtaining the information then the fields are populated appropriately. It is also noteworthy that when the user selects manual on the scroll down menu, the GUI adds another scroll down menu with all the locations known by the system and then the

user can select one and trick the system into thinking that is the iPAQ's current location, an invaluable tool for testing purposes.

From the LocationGUI, the user can press the "Update" button at any time to refresh the location information or else press the ViewLocations or EditGroups buttons to access yet other location related GUI's. The Figure 9 shows the GUI that is displayed when the "View Locations" button of Figure 8 is pressed. This GUI allows the user to modify all the mappings between the location names the system has in its database and the actual location information that the IP or Cricket LocationProviders will return. For example, in Figure 9 the IP 128.52.0.0 is mapped to the AI-Lab, however, the user can go ahead and change this IP and save the new mapping. Likewise, this same manipulation can be done with the Cricket Location System information.

Now if instead of pressing the "ViewLocations" button from Figure 8, the

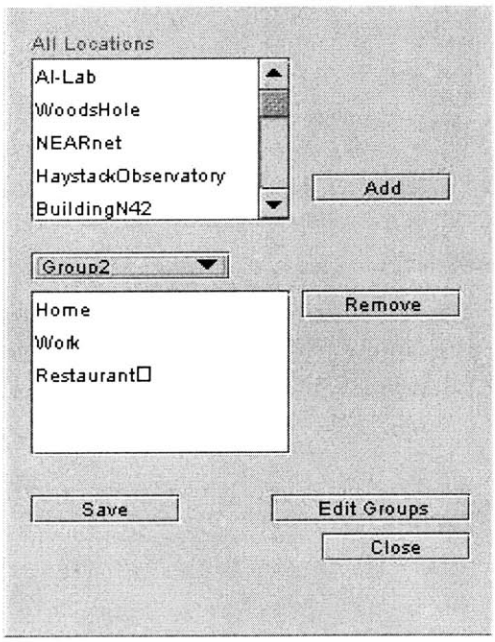

Figure 10: Editing Groups

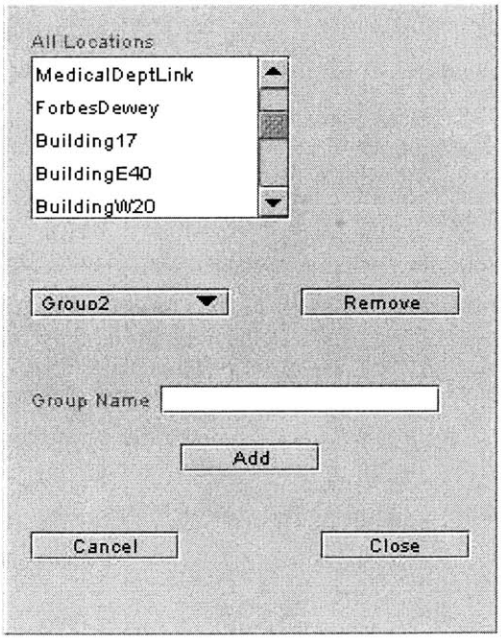

Figure 11: Creating and Deleting Groups

"EditGroups" button is pressed, the user will be presented with another GUI where he can start taking advantage of the power of grouping multiple locations under a single

name. Figure 10 shows the GUI that the user can use in order to select the already existing location groups and then add or remove individual locations. Finally if the user pressed the "EditGroups" button, the GUI shown in Figure 11 will be displayed. This new display allows the user to create or remove actual groups within the system.

All these interfaces allow the user to have full control over the location capabilities of the system. The user is then fully empowered to find out the current location of the system and override it manually if necessary, to re-map the specific details of all the locations, and finally to group all the locations for convenient access.

III.5 User Preferences

The bottom-most button on Figure 3 leads the user to the settings GUI where several key system parameters can be changed. The first drop down menu simply lets the user set the default location in case there is no available location provider, then the user can specify how often the location providers should query their respective systems to discover the iPAQ's location. The next to last setting comes to play when the system's meeting database is completely empty and the main thread sets itself to sleep in increments of time defined by this setting. And finally, the system time that we get from the operating system might not match exactly our time zone, for example, if we get GMT time back, the user can use the time offset to make up for the difference.

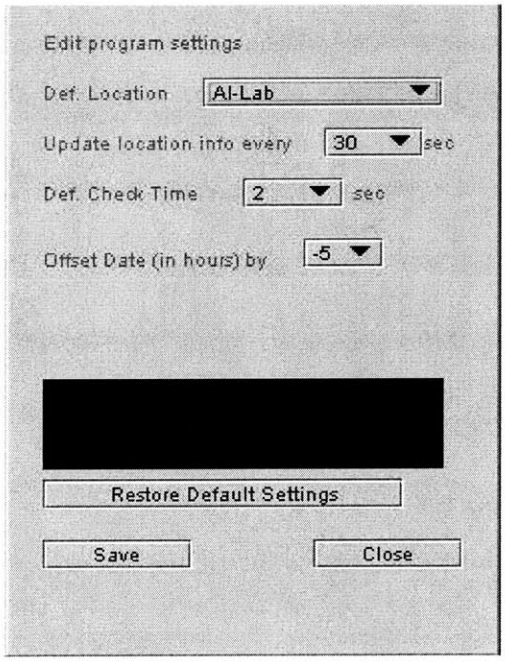

Figure 12: Settings GUI

IV. System Design

IV.1 Modules and their Interactions High Level

The main data structures of this system hold Alarm, Meeting and LocationAlarm elements that will be added, edited or removed through multiple GUI's. These main data structures are AlarmList, MeetingList and LocationAlarmList. It is very straightforward to understand how when a new activity or meeting reminder is created, a new object holding this information is stored in the MeetingList class. Moreover, the appropriate Alarm object for the new Meeting is also created and stored in the AlarmList data structure. Finally, when a new LocationAlarm object is created, it is stored in the LocationAlarmList data structure. These three data structures all have basic access functionality including adding, deleting, editing and sorting elements. With this as the core of our system, other user interfaces can access the data for display or manipulation purposes.

The second core part of our software consists of the classes that deal with location. At a high level, we have two LocationProvider classes, one for the Cricket Location System and one for our IP location mapping system. These two LocationProviders are queried by our LocationService class that actively queries the providers and notifies AlarmList and LocationAlarmList every time location changes. In this way, the flow of information reaches the Alarm and LocationAlarm elements inside our repository data structures and they trigger their alarms accordingly. All this ties in with the main thread of the system that wakes up periodically to check if any time based alarm should trigger, recall that some of our alarms trigger in relation to location and some in relation to the time left before an actual activity.

IV.2 Low Level Descriptions of Modules and Components

IV.2.1 Meeting

An object of type Meeting holds information regarding a specific activity reminder. It holds several Java GregorianCalendar objects that precisely encode the date and time of this particular Meeting object as well as the date of the Alarm object that is created depending on the advance notification time selected by the user. The GUI that creates a meeting gives the user the option of selecting this time from a scroll down menu ranging from 10 minutes to 1 day. Moreover, it holds String representations of the Meeting's name and any other detailed information the user might have added.

Also importantly, the Meeting object holds the location that the user inputted was the place where this activity was going to happen. This information is also passed on to the Alarm object created by this Meeting. Finally, a Meeting object is also aware whether the user wanted it to be a "smart" Meeting or not. The difference that this makes is that the Alarm object created will also be "smart" or not and it will care of its proximity to the Meeting's location and will alert the user if there is not enough time remaining to travel from its current location to the intended destination. This estimation is done though our estimation matrix class.

IV.2.2 Alarm

The Alarm object is one of the main building blocks of our system. These elements are created when a new Meeting object is created. They are stored inside the AlarmList repository and hold GregorianCalendar representations for the time in which they should trigger, the time of the Meeting from which they were created as well as information regarding the location of the final meeting and the current location of the

system which it receives from the AlarmList that contains it. Furthermore, the Alarm object also knows if it is a smart alarm or not, which ties in tightly with the Meeting that created it.

An important aspect of each Alarm object is that the locationChanged(String location) method allows the AlarmList object that holds it to change its representation of the system's current location. When this happens, the Alarm object re-computes the estimate of when exactly it should trigger an alert. This however changes whether the object is a "smart" Alarm or not. If the Alarm object is not smart then a change of location in the system makes absolutely no difference. On the other hand, if the alarm is indeed "smart" then when the Alarm object is notified of a change in location then it queries the estimation matrix, (the class is called DistanceToTravel) and gets back an estimate of the time necessarily to travel from the current location to the destination. Using this, the Alarm changes its internal estimation of when it should trigger and if necessary will even trigger immediately notifying the user that he or she must leave at once to be on time.

IV.2.3 LocationAlarm

These objects are very straightforward and are created directly by one of the system's GUI's. They represent reminders that are based solely on location and will trigger every time the iPAQ is at a particular place. Moreover, they also allow the user to specify group of locations instead of a particular one. These objects also hold representations of the names given by the user as well as an extra message field and the time in which it was created. All these objects are stored in the LocationAlarmList repository and are informed by this object when the system's location changes. When this

happens, each LocationAlarm object compares the current location with its particular location and if they match an alert GUI is triggered.

IV.2.4 MeetingList

As mentioned briefly before, this is one of the 3 main repository data structures in the system. It holds Meeting objects and allows the system's GUI's to add, delete, modify search and sort Meeting objects. However, anytime a Meeting is deleted, the MeetingList also accesses the AlarmList and deletes all corresponding Alarm objects. This class is also able to save its state directly to disk in order for the system to keep continuity throughout reboots.

IV.2.5 AlarmList

The AlarmList data structure holds all the Alarm objects, both the "smart" Alarms and the normal ones. This is another repository style data structure that gives other classes the capability to add, remove, edit and sort its elements. This data structure also has the capability of determining among all the Alarms within it the amount of time remaining until the Alarms will trigger and report the smallest amount of time. This information is used by the main thread of the system when it goes to sleep until the next time an Alarm will trigger. This class also has the capability of saving its state to disk in order for the user not to lose the information that has been inputted. A key feature of this class is that it is notified by the LocationService class when there is a location change and the AlarmList consequently passes this information to all the Alarm objects that then modify themselves accordingly.

IV.2.6 LocationAlarmList

This is the final of the three data structures that hold elements within the system. The LocationAlarmList class holds the LocationAlarm objects discussed before. This class also allows for standard access including adding, deleting, editing and sorting of the elements within it. Every time the LocationService class determines that the iPAQ's location has changed, one of the things it does is notify the LocationAlarmList. Once this happens, all the LocationAlarm objects inside the LocationAlarmList are notified of this change and then they trigger their alerts accordingly.

IV.2.7 LocationService

The LocationService object is a crucial element of our location system. This class consists of a thread that every fixed amount of time queries both the CricketLocationProvider and the IPLocationProvider in order to find the iPAQ's current location. The amount of time between each check is determined from the system settings GUI. Moreover, every time the LocationService realizes that the location has changed it notifies both the LocationAlarmList and the AlarmList, both of which propagate this information.

IV.2.8 CricketLocationProvider

This is one of the two location providers that our system uses in order to find out about the iPAQ's location. This class is a wrapper for code developed by Allen Miu which has extended functionality, albeit we only used the functionality of his code that grabs the raw data from the Cricket Listener and reports the beacon closest to the iPAQ. The code from Allen Miu returns the id of the closest Cricket beacon to the

CricketLocationProvider and this class in turn provides the LocationService with this information.

IV.2.9 IPLocationProvider

This is the second location provider. It uses the fact that since MIT is a class A IP, every building has a different subnet and can be differentiated by the particular IP the system is assigned by the network. Henceforth, as one moves throughout campus and is assigned different IP's per building, this current IP can be used to give a rough estimate of the iPAQ's location. In this way, the IPLocationProvider reports to the LocationService class the building that maps to the current IP of the iPAQ. Clearly this is not perfectly accurate but since the Cricket Location System is only deployed in one MIT building, we can extend our range of movement greatly by utilizing IP information.

IV.2.10 ManualLocationProvider

For situations in which we do not have access to Cricket beacons or our database is not able to map the IP to a building, the user can simply override the other two location providers by using a GUI to specify the location the system is in. This is incredibly useful for testing purposes.

IV.2.11 Groups

This class reads in the system file in which we specify group names and all the locations that belong to each one of them. The actual implementation is very straightforward and consists of various vectors, one of which contains the group names and a second one that contains lists of the actual locations. Moreover, the class provides repository style controls including adding, removing and deleting groups and locations.

IV.2.12 TimeToTravelMatrix

This is a crucial element for our "smart" Alarm system that is able to recommend when the user should start moving towards a Meeting's destination in order to be on time. Basically it allows the system to input two locations, a starting and ending point and the matrix will return an estimate of the amount of time a person would take to walk from one place to another. The actual information contained by the matrix in this version of our system is a random number, however, extensions to our system would include different types of transportation methods, e.g. by car, walking, etc as well as returning pre-measured times instead of random information. Since right now the system's goal is to become demo-able, the current functionality of the matrix is enough, but as mentioned before a fully functioning system would require more realistic values.

V. Evaluation of System

V.1 Design Alternatives

Often, situations were encountered were we had to judge between multiple alternatives in order to find a solution to a particular problem that would be the most beneficial for our thesis in terms of feasibility, performance and time constraints. Some of these were decided completely by us and in other cases we did have to follow limitations and procedures from either the practices already utilized by the Oxygen group or else the hardware that was available.

V.1.1 Hardware

The PDA hardware utilized included the Compaq iPAQ 3670 with a dual PCMCIA card expansion sleeve along with an 802.11b wireless card and a 1 gigabyte

IBM Microdrive. This combination of hardware allowed us to assume that we had the power of a small workstation available to us as we moved around the laboratory. The actual PDA is based on a StrongARM processor running at 206 MHz with a 4096-color LCD with resolution of 320 x 240 pixels. It also has up to 32 megabytes of SDRAM, up to 32 megabytes of flash ROM with a clip-on system for hardware expansion as well as a serial port [3].

Moreover, we also took advantage of the hardware expansion capability and used the dual PCMCIA sleeve in order to connect our wireless and extra storage cards. Fortunately, this dual card sleeve also has a battery of its own without which all the extra hardware would have drained the main iPAQ's batteries in a matter of about 2 hours.

Now in terms of connectivity we used a standard 802.11b wireless card manufactured by ENTERASYS NETWORKS and took advantage of the fact that all of MIT and LCS have already deployed a wireless network. Finally, in order to avoid space restrictions for our software, operating system and other applications, we also utilized a PCMCIA sized hard disk developed by IBM called the Microdive. We utilized the one gigabyte version of their Microdrive although we sometimes also used a smaller 128 Compact Flash Type II card for testing purposes or to exchange information.

One final aspect of the PDA's hardware that was invaluable was the fact that we could connect yet a third hardware device through the serial port. The Cricket Location System Listener that we used interfaced with the PDA through this port.

V.1.2 Java vs. C

Beyond the hardware that we utilized, it was essential to determine what programming language we were going to employ to develop our application. The main

options that we had at the beginning of our thesis were Java and C, both of which have support under the version of Linux we were running on the PDA. Clearly both languages have their advantages and disadvantages. On the one hand C would be able to execute faster since it is compiled and generally more efficient. On the other hand, several programs already developed for the Cricket Location System were already available in Java and personally both coders in this project have much more experience working with Java. Furthermore, the fact that java is more easily ported among different platforms promotes distribution considerably and this was yet another factor we took into account. Consequently, the decision was made to develop the software for this thesis using Swing GUI's as well as JDK 1.3 which was the latest version available for the StrongARM processor used by the iPAQ.

V.1.3 Linux

One of the most important decisions made in regard to the development platform for this thesis was the adoption of a version of Linux as the operating system for the iPAQ PDA. In order to maintain a fully modifiable environment, an open source operating system was invaluable. Moreover, the fact that we were working with both Project Oxygen and

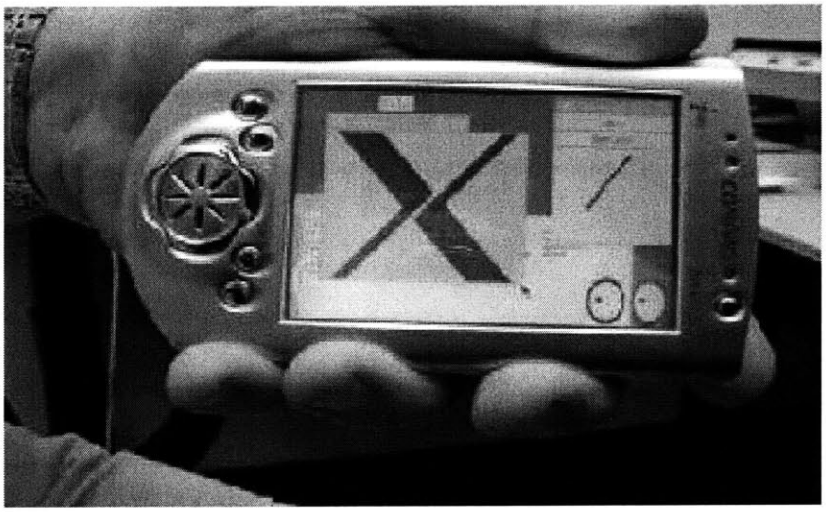

Figure 13: iPAQ running Linux

Compaq's Cambridge Research Labs made our access to several versions of Linux with the necessary characteristics extremely straightforward. Among the many advantages of using Linux, we have the powerful networking options such as IPv6 and Mobile IP capabilities along with a fully functioning JVM and X-windows, the capacity to use multiple file systems and the growing network of developers associated with the platform. We decided to utilize the distribution of Linux used by the MIT class 6.964 (Pervasive Computing) inasmuch as we could receive support from the teachers of the class as well as receiving help from CRL. The distribution is called Familiar which is based on Debian Linux and is maintained by handhelds.org.

V.1.4 Location Service

One of the crucial parts of our thesis was the way in which we were going to do location discovery. We had several options about threading and polling both of which had their pros and cons. Our main concern was whether we were going to have a push or a pull system. That means that our LocationService class could have actively queried the IPLocationProvider and CricketLocationProvider classes every set amount of time or it could have waited to be notified by the LocationProviders of a change in the PDA's location.

First of all, if we had a pull system in which the LocationService had to have a thread that constantly queried the LocationProviders we would be wasting system resources. For example, if the user stayed static in a location we would still be querying every 20 seconds or so. On the other hand, a push implementation would rely more heavily on the location providers and would not have a thread that polled every set amount of time. The best implementation possible therefore, would be to pass the

responsibility of determining a location change to the actual LocationProviders by having a more sophisticated implementation of these.

This notwithstanding, due to time issues and the fact that we wanted to get a working system as soon as possible, our actual LocationService class enforces a straightforward pull system that queries the LocationProviders every 30 seconds. In this way, it is fully functioning albeit not in the most efficient manner. Improvements to our software would seriously consider implementing this aspect in a more push-oriented manner.

V.1.5 List Sorting

One of the main aspects of data structures that hold our Alarm and LocationAlarm classes is sorting. As we add more elements to the MeetingList, AlarmList and LocationAlarmList data structures, as well as when PDA's location changes, we resort the elements. In the case of AlarmList and MeetingList we sort by the time at which the Alarm should trigger or the time of the actual Meeting. In the case of the LocationAlarmList we sort alphabetically by the name of the PDA's current location.

As of the writing of this document, all our data structures use the insertion sort algorithm to sort their elements, and when multiple elements within the list change the entire list is resorted by constructing a new list and adding the elements one by one. This might seem inefficient inasmuch as insertion sort takes $\theta(n^2)$ time in the worst case, however "because its inner loops are tight [...] it is a fast in-place sorting algorithm for small input sizes" [7]. Fortunately, our system only deals with very small lists that do not even reach a thousand elements. Henceforth, our decision to use this quick implementation was more than adequate for development and testing purposes. This

notwithstanding, we would definitely re-work our sorting algorithm for a final system that might have to deal with much larger data sets. It would be a straightforward task of re-implementing our sorting methods to utilize more powerful algorithms such as merge sort that can perform our same task in $\theta(n \lg n)$ time.

V.1.6 System Files

In order to keep the state of our system, we also determined to use several system files. First of all, to keep all our Meetings, Alarms and LocationAlarms even after we power-off or reboot the iPAQ we decided to use the serializable interface of JAVA which would enable us to easily save our classes to disk and then read them back at startup. We could have devised another way of saving our state, for example we could have made String representations of Meetings, Alarms and LocationAlarms and saved those to a text file. Afterwards, we could have parsed the data, created new objects and gone from there. The advantage of using this second String alternative is that if our original classes change in any way, Java's serialization method does not work and you would not be able to recreate objects from the files you have on disk anymore. Since we wanted a quick implementation of our state saving mechanism we decided that that price was not significant as long as we were well aware of it and the fact that our testing scenarios did not contain numerous elements helped as well. The names of these three files are AlarmList.dat, MeetingList.dat and LocationAlarmList.dat.

Moreover, we use another system file called Settings to keep records of the user preferences such as default location, offset date from the system time and others. In this case we decided not to use the Java serializable alternative but to actually save information as text on a file. The reason that we did this was that the settings do not

represent a single class, rather, they are accessed by different classes when they need the information. Henceforth, since we do not have a class that embodies these preferences this was the most logical alternative.

The last set of relevant system files that we used are the files that help us make sense of the location information we get from the Cricket Location System and from our IP masking approach. These files are called CricketTable and IPtable respectively and serve as a basic mapping between the encodings we receive from the CricketLocationProvider and the IPLocationProvider classes and the display the end user would actually find useful. In this way, both of these system files are text files that have a one to one mapping. For example, each Cricket beacon sends a specific id that is recognized by the listener and the text is what is read by the CricketLocationProvider. However, we would like to map this id (which can be any text String) to a location name that either the developers or the user has predetermined. Henceforth, this file will save this state. Similarly, the IPLocationProvider returns the current IP of the iPAQ and the IPtable file maps this IP to a location name. Moreover, for us to be able to easily modify this mapping it was sensible to implement these files as text files that could be directly edited instead of using a serializable approach is we did with other elements.

A third file that falls within this location category is called Groups and is able to make the creation of Meetings and Alarms easier because it groups multiple locations under a single name. In this way a user can create a Group that includes all the locations inside a particular building or any combination he or she desires. Again, the file itself is a simple text file with horizontal entries from left to right that start with the name of the group and then hold all the locations of the group to the right separated by colons. This

simple implementation allows us to easily modify the groups directly in the file if we need to for development purposes, but of course the user can also modify this through one of our GUI's.

The actual implementation of the Group file was indeed a question we discussed inasmuch as our current implementation does have faults that could lead to trouble later on. For example, if a group contains a location name that is deleted from the system, this location is not deleted from the group might cause incorrect behavior, or at least unexpected behavior. Our other alternative in order to avoid this problem was to create another data structure instead of a simple file and this data structure would then have had links to the actual locations specified by the user so that in case a location was deleted then the group would no longer have inaccurate data. This would have also required us to rethink the way we defined locations and gone from Strings to more complicated objects. After judging the alternatives we decided to choose a simple string representation although it might have inaccurate data since this implementation would be more than adequate to show a fully functional system and enable us to continue developing other areas without spending much time on it. Again, if we did decide to finish the coding for a release version our group ideology would change and we would make sure that we have no "ghost" locations in our system.

V.1.7 Example of System Files

As mentioned in the previous section, the system utilizes several files to keep track of its state. Some of these are text editable and some are not. Following, three text editable files will be discussed briefly with specific examples in order to make it possible

for the reader to modify these files directly if he or she so wishes. The files that will be treated are CricketTable, IPTable and Groups.

The CricketTable file provides a one to one mapping between Cricket beacon id's and the actual location label we want the system to utilize. Ex:

LR02-32 6FloorLounge1
LR01-32 6FloorLounge2
LR6-32 6FloorLounge3
600LL-33 6FloorLounge4
600LR-32 6FloorLounge5
617B-32 6FloorLounge6

There are two entries per line separated by white-space, the one on the left is the exact beacon id and the one on the right is the label that the system will utilize. Consequently, all the user has to do is add more id to label mappings, one per line or else modify already existing ones.

The second file in question is IPTable which follows a format very similar to that of CricketTable. It also has one to one mappings only in this case those are mappings from IP addresses to location labels. A small portion of a file is shown in the next example.

128.52.0.0/16 AI-Lab
128.128.0.0/16 WoodsHole
192.52.0.0/16 NEARnet
192.52.64.0/24 HaystackObservatory
18.2.0.0/21 BuildingN42
18.2.8.0/21 BuildingE52
18.2.16.0/16 BuildingE52
18.7.0.0/27 Akamai

As in the case of CricketTable, a user simply has to add one mapping per line or modify already existing ones. The left element in each row is the IP and the right element is the label. Something noteworthy is the extra information that appears after the IP address, the subnet mask. Consequently, right after the IP the user must input a "/" followed by the number of bits of the mask key. There is no white-space between them and the numbers

can range from 32 for the most significant to 16, any number smaller would work but would specify the entire MIT campus.

The third file is Groups and it allows the user to define location groups and the elements for each. The format is very simple and consists of the group name as the left most element of each line. This is followed by a colon and a series of locations, all separated by colons. Note that there is no white-space between elements in each row.

Group1:Lobby 7
Group2:Home:Work:Restaurant
Group3:Store:School
LCSOFFICES:KenOffice:JackOffice:TedOffice

V.2 Test Results

The majority of our testing took place on the sixth floor of the LCS lab where numerous Cricket beacons are deployed. Clearly our testing consisted of several stages starting from making sure the Cricket system was able to accurately determine our location to a full system check to make sure all our features functioned correctly.

After our initial testing with the Cricket system, which is explained more fully in Chapter V.3, we also performed multiple tests throughout MIT campus to make sure that our IPLocationProvider would be able to properly parse the network information and that the mappings where correct. We tested this system in MIT buildings, 1, 3, 5, the LCS building and N42. Indeed this module was able to accurately return the building in which we were located each time we tested it.

The next step in our testing consisted in interfacing both location modules and actually testing our reminder mechanism with normal alarms, smart alarms and location based alarms. First of all, our time-based reminders worked well within our specification,

both with our original threading approach that woke up every fixed amount of time and with our modified system that went to sleep until the next alarm was supposed to trigger.

The next logical step was to test the system's location capabilities and initially we did notice a mayor bug that forced us to modify several module interactions. We had problems with the precedence we had between the different location providers. For example, if the user had made a LocationAlarm at a particular location that was defined by both, Cricket Location system and IP terms, the Cricket Location system would always override the other. This means that if the system where getting Cricket readings it would not trigger an alarm that was defined by IP terms even if the IP the system where currently using was the correct one. This happened again because of the preference we gave to Crickets. This of course was fixed by making the system smartly check both the IP and Cricket location information and assign equal preference to both. After this change was made the system's bug was corrected and it behaved fully within our expectations.

V.3 Experiences with Location System

In experimenting with the Cricket Location Technology, we participated in a project from the class Pervasive Computing 6.964 where part of the final 4 student project consisting in tracking the position of 4 to 8 users simultaneously as part of a game. The infrastructure used for this location tracking was the Cricket Location System and we thought it would be an excellent test bed for the system before utilizing it on our thesis project.

Originally, the way in which we tried to track the users was by utilizing pre-existing code that interfaces with the Cricket System. This code was developed by Allen

Miu as part of his MEng Thesis. However, our efforts made us aware of multiple issues that made it particularly hard to utilize x-y-z coordinates with the system's current implementation. On the one hand the code performed some pretty computational expensive matrix calculations that significantly taxed the iPAQ's hardware. Moreover, the tracking itself was not precise or reliable enough for a practical game.

We utilized and tested two versions of Allen's x-y-z coordinate computing code, both of which returned coordinates in respect to a predetermined coordinate system and also allowed you to determine which beacon you are closest to. His original code required that a listener receive signals from 4 beacons in order to output xyz coordinates and at the same time the system was bogged down when it received readings from numerous beacons, i.e. approximately more than 7. This was a challenge for mobility since many areas within the playing field did not have access to 4 beacons and some areas received more than 7. Therefore we utilized a fault protection alternative which recognized when we were not receiving updated xyz coordinates and therefore reverted to simply finding the beacon closest to you. Following this, we used the knowledge that we had of the exact location of each beacon and used this as an approximation of the player's position. In this way, the tracking of the player was much more accurate than simply relying on the xyz feature set. Following this, we discussed our problems with Allen and he made available another version of his code that performed the actual triangulation computation differently. This version of the code was not limited when we received readings from numerous beacons and in fact gave more accurate estimates of location when it had more readings. Unfortunately the code was still too burdensome for the iPAQ and we had to look for yet other alternatives to determine xyz location. However, the

spatial recognition, the analysis of which beacons was closest, worked efficiently and accurately enough for our purposes.

As we tested our tracking system, we found that sometimes the hardware itself did not behave reliably. By this I am referring directly to the beacons. It was our experience that we set up multiple beacons on the second floor lounge of LCS as a testing ground for our system and only one of the beacons was emitting signals that the listeners could read. It is noteworthy to mention that we utilized Cricket beacons with different types of antennas and the ones on the second floor lounge used coiled wires. We then moved our iPAQ's to the sixth floor and got much better readings from the beacons there, which have other types of antennae. We have not tried to reproduce this problem but it is certainly noteworthy that in this case it seemed that there was a considerable difference between the two types of beacons.

Overall, we noticed severe limitations to the system particularly when we tried to track users with x-y-z positioning. As mentioned before, the reliability and preciseness of our measurements was lacking and we decided to focus on spatial differentiation which did behave much more robustly.

VI. Related Work

VI.1 NetGeo - The Internet Geographic Database

The NetGeo location system follows an idea very similar to our IP Location system. To a rough degree, it extrapolates geographic information from IP addresses, Autonomous System numbers, and domain names and creates a mapping database to longitude/latitude data. This is similar to our IP Location system because both these systems use the internet addressing system which is just a hierarchical structure with very

little location information embedded into it to allow users to have more information about location. This system is being developed by the Cooperative Association for Internet Data Analysis (CAIDA) whose mission is to provide "tools and analyses promoting the engineering and maintenance of a robust, scalable global Internet infrastructure" [9].

The specific way in which the NetGeo system works is by performing "whois" queries and then parsing the information that is returned, including city, state, zip code, country and even telephone numbers. From this information the NetGeo database attempts to infer the actual geographic information by using clues parsed from the "whois" search but also using applicable emails and the suffixes of domain names. The suffixes are helpful since many two letter ones are specific to countries.

This system does run into serious difficulties when you realize factors such as the existence numerous large intranets that are mapped to a single or few IP addresses in locations completely unrelated to the system we might want to map. For example, large networks such as the ones used by Hewlett-Packard and Microsoft will cause most computers in any country operated by these companies to be mapped to Palo Alto, California or Redwood, Washington. Even more complications arise when mobile IP's come into play, however the added information given by this system can be sufficiently accurate for systems to use location estimation to aid users with their end applications.

VII. More thoughts, Conclusions and Expansions

The final software that we have created for this thesis is a good example of the possibilities that are available once contextual information is used to expand the functionality of current applications. By adding the location of the system, the possibilities of helping a user are incremented significantly. A normal scheduler

application would never be able to provide the kind of advance reminders that our system is able to provide because of its time to travel estimation capabilities. In this way, we have exemplified the advanced capabilities that software can display once it has access to context information.

There are several enhancements however that we did not have time to fully implement or that we were unable to do work on because of lack of facilities. One major area that we would have liked to expand upon was the capability of providing our scheduler – reminder application for outdoor use. We wanted to provide the user the capability of adding reminders that would trigger by proximity as they were moving around freely outside a particular building, e.g. driving. We wanted to tap into online databases where our system could find information about multiple services and their exact location. Moreover, this goal would have required us to use another internet provider beyond the wireless 802.11b network that MIT provides. Of course this would have meant much slower speeds and that would have required yet more testing to measure if it were feasible at all. We would have liked to add these elements into our software and allow the user to input a reminder that would have triggered while driving a car and coming near to a theater, a particular store, etc.

Beyond providing functionality outside a building as I just mentioned, another aspect that this would have required would have been to actually make it possible to seamlessly transition from one network to another. For example, the system would have had to recognize when we are moving outside a building, and then moving from MIT's 802.11b network and obtain a signal from a wireless web provider as well as letting go of the Cricket Location System and accessing GPS location instead. This was our best

scenario goal, however, seamlessly transitioning between indoors and outdoors functionality is impossible to achieve with our current hardware. We are limited because the iPAQ that we are using has an expansion slot that allows us to use exactly 2 PCMCIA cards simultaneously and one device connected through the serial port. When we are indoors, we utilize one PCMCIA card for out wireless connectivity and a second one for secondary storage meanwhile we connect the Cricket system through the serial port. Now if we wanted to go outside a building, we need to physically disconnect the wireless 802.11b card and connect a wireless modem, and what is more, we need a GPS receiver that can be connected through the serial port, something that is not easy to find inasmuch as normally they come in PCMCIA card format. Consequently, the best that we can strive for is a system that would require no software change beyond the hardware reconfiguration, something that we have no way of avoiding at the present moment.

The capability of moving outdoors would have greatly enhanced our system, however, that will be left for a future upgrade. The hardware limitation that we experienced is by no means permanent and we believe future revisions of the iPAQ will be able to overcome these problems. In the meantime, we feel that the current functionality of the system is more than adequate to demonstrate how a user could take advantage of the added location features.

Although the current system is fully functional, we have thought of more improvements that are not as far reaching as providing dual outdoor functionality. Since one of our goals was to aid the development community, we wanted to provide all our source code in a convenient ipkg format that users could easily download. This form of packaging software is widely used within the development community both at MIT

Oxygen and at Handhelds.org. This was not possible due to time constraints and although the alternative of a tar file is not elaborate in itself, it is not as straightforward as an i-package.

Another way of aiding other developers work with our code and system was to try to make it as easily portable as possible. This was one of the reasons why the entire system was developed in java except for a daemon that runs locally on the iPAQ and reads information sent from the Cricket Location system via the serial port. Compatibility with the Windows 9x family of operating system from Microsoft would have been very helpful in order to expand the possible user base for your system. In its current state however the software is not fully compatible. It would need several modifications that, albeit not being extensive, make it impossible to fully function presently. These modifications include modifying the IP Location Provider module to use windows commands to acquire the system's current IP and consequently a change in our parsing procedures. Secondly, we would need to recompile the daemon that runs locally on the iPAQ and receives the information from the Cricket Location system to run on x86 processors instead of the ARM it currently is compiled for. With these relatively straightforward changes and additions, our system would be fully functional in a Windows environment, be that a laptop or a pocketPC handheld.

Finally, we would have liked to explore the possibility of using more sophisticated tracking mechanisms indoors that go beyond space recognition and function similar to the GPS system. This would therefore provide us with x-y-z coordinates or something similar. A version of this that was implemented by Allen Miu was tried but it was not sufficiently accurate or responsive. Moreover, for a correct functioning we would

need to constantly access an online database to map coordinates to locations. This however is not terribly complicated; it just required an accurate tracking system.

VIII. Resources

[1] Cormen; Thomas, Leiserson; Charles, Rivest; Ronald. Introduction to Algorithms. McGraw Hill. Cambridge, Massachusetts. 2000.

[2] "Handhelds.org". Marc 24, 2002 http://handhelds.org. (April 12, 2002).

[3] iPAQ H3600 Hardware Design Specification. http://handhelds.org/Compaq/iPAQH3600/iPAQ_H3600.html (March 02, 2002).

[4] "MIT Project Oxygen". http://oxygen.lcs.mit.edu/ (March 13, 2002).

[5] "NetGeo, the Internet Geographic Database". December 19, 2001. http://www.caida.org/tools/utilities/netgeo/ (1 April, 2002).

[6] "Cricket Indoor Location System". http://nms.lcs.mit.edu/projects/cricket/ (January 2, 2002).

[7] Priyantha, Nissanka B.; Chakraborty, Anit; Balakrishnan, Hari. *The Cricket Location Support System*. Mobicom 2000, The sixth Annual International Conference on Mobil Computing and Networking. Pg. 32-42. ACM Order Department, New York, NY. 2000.

[8] "Trimble, all about GPS". http://www.trimble.com/gps/index.html (March 30, 2002).

[9] "Where in the World is netgeo.caida.org?" December 17, 2001. http://www.caida.org/outreach/papers/2000/inet_netgeo/ (March 26, 2002).

IX. Apendix 1. Code

IX.1 Alarm.java

```
/**
 * Alarm Class maintains information about an event (meeting)
 * and alerts the user of the event at the notification time
 *
 * @author Marc Bourget
 */

package SmartReminder;

import java.util.*;
import java.io.Serializable;
import SmartReminder.Location.*;

public class Alarm extends GregorianCalendar implements Serializable {
    /**
     * alarmDate is the notification date
    */
    private GregorianCalendar alarmDate = null, meetingDate = null;
    private String destLocation = "", mName = "", currentLoc = "";
    private boolean smart = false;

    public static void main (String argv[]){
      GregorianCalendar today = new GregorianCalendar(2001, 4, 21, 23,
15);
      Alarm wakeup = new Alarm(today, "baker", false, today, "Baseball
Game");
      wakeup.isSmart();
      wakeup.getAlarmDate();
      wakeup.alert();
    }

    /**
     * Constructor for alarm, called by a Meeting object
     */
    public Alarm(GregorianCalendar aDate, String loc, boolean sm,
GregorianCalendar mDate, String name) {
      alarmDate = aDate;
      meetingDate = mDate;
      destLocation = loc;
      smart = sm;
      mName = name;
      locationChanged(LocationService.getLocation());
    }
    /**
     * tells if the alarm is smart or not
     */
    public boolean isSmart() {
      return( smart );
    }

    /**
```

```
     * returns the notification date for meeting
     */
    public GregorianCalendar getAlarmDate() {
      return( alarmDate );
    }

    /**
     * returns the meeting which this alarm belongs to
     */
    public GregorianCalendar getMeetingDate() {
      return( meetingDate );
    }

    /**
     * called when the present location has changed
     */
    public void locationChanged(String l) {
      currentLoc = l;
      System.out.println("------------Location Changed inside Alarm
called with location "+currentLoc);
      /**
       * only smart alarms care about the present location
       */
      if (smart) {
        long travelTime =
DistanceToTravel.getTimeToTravel(l,destLocation);
      int adjust = (int)(travelTime/60000); //convert travelTime to
minutes
      System.out.println("-----------Alarm Date being changed by
"+adjust);
      alarmDate.add(alarmDate.MINUTE, (-1 * adjust)); //adjust the
alarmDate accordingly
      }
    }

    /**
     * The AlarmList calls this method to trigger the alarm
     */
    public void alert() {
      if ( !( currentLoc.equals( destLocation ))) {
      (new AlertGUI(mName, meetingDate, false,
destLocation)).setVisible(true);
      System.out.println("This is an alarm. ALERT ALERT");
      }
    }
```

}IX.2 AlarmList.java

```
/**
 * AlarmList Class The AlarmList data structure holds all the Alarm
objects,
 * both the smart Alarms and the normal ones. This is a repository
 * style data structure that gives other classes the capability to add,
 * remove, edit and sort its elements.
 *
 * @author Craig Music
 */
```

```
package SmartReminder;
import java.util.*;
import java.io.*;

public class AlarmList {

    static private Vector tempVector = new Vector();   // used for
sorting
    static private Vector alarms = new Vector(); //holds the Alarm
objects
    static private int pointer;
    static private boolean debug = false;
    static private boolean debug2 = false;

    static public void e(String d)
    {
      if(debug2 == true)
          {
            System.out.println(d);
          }
    }
/*
* This method notifies every single Alarm object within this data
structure
* of a change in location.
*/
    static public void locationChanged(String currentLocation) {
      int index = 0;
      int size = alarms.size();
      for(index = 0; index < size; index++) {
      Alarm temp = (Alarm)alarms.elementAt(index);
      temp.locationChanged(currentLocation);
      alarms.set(index, temp);
      }
      AlarmList.updateState();
    }

    static public void updateState() {

System.out.println("*****************************\nalarmList.udpateSta
te called");
      AlarmList.resort();
      SR.updateTimeToNextAlarm();
    }
/*
* Used internaly to resort elements within the data structre
*/
    static public void resort() {
      AlarmList.tempVector = new Vector();
      int index = 0;
      int size = alarms.size();
      for(index = 0; index < size; index++) {
      Alarm a = (Alarm)alarms.elementAt(index);
      AlarmList.insert(a, tempVector);
      }
      AlarmList.alarms = new Vector();
```

```
      AlarmList.alarms = AlarmList.tempVector;

    }

/*
* Will query all the elements within the data structure for the time
left
* until their alarms should trigger and will return the smallest
quantity.
*/
    static public long timeToNextAlarm() {
      int size = alarms.size();
      if(size <= 0) {
      return SR.defaultWait;
      }
      Alarm firstAlarm = (Alarm)alarms.elementAt(0);
      GregorianCalendar nextTriggerCalendar =
firstAlarm.getAlarmDate();
      Date nextTriggerDate = nextTriggerCalendar.getTime();
      long timeOfNextAlarm = nextTriggerDate.getTime();
      long currentTime = CurrentDate.getCurrentTimeMillis(); //in
millis
      long timeToNextAlarm = timeOfNextAlarm - currentTime;
      if(timeToNextAlarm < 0) {
      timeToNextAlarm = SR.defaultWait;
      }
      System.out.println("Time until the next meeting from Alarm List:
" + timeToNextAlarm);
      return timeToNextAlarm;
    }

    static public void main(String args[])
    {

      AlarmList.renew();
      GregorianCalendar today = CurrentDate.getCurrentCalendar();
      next();
      e("\n\njust requested next, should isplay nothing");
      Alarm create = new Alarm(today, "BaseballGame", false, today,
"asd");
      insert(create);
      alarms.toString();

      Alarm a = next();
      e(a.getAlarmDate().toString());
      e("\n\njust requested next, should have returned and diaplayed
soomething");


      insert(create);
      e("\n\njust inserted alarm, the sane one");

      Alarm b = next();
      e(b.getAlarmDate().toString());
      e("\n\njust requested next, should return nothing");
    }
```

```
    static public void renew() {
      alarms = new Vector();
    }

    static public Alarm invalid() {
      GregorianCalendar iv = new GregorianCalendar(1970,1,1,1,1,1);
      Alarm ivv = new Alarm(iv, "invalid", false, iv, "invalid");
      return ivv;
    }
/*
* Method used to save state of the data structure to disk. Uses the
* serializable java interface.
*/
    static public boolean save()
    {
      boolean result = false;
      System.out.println("serializing Alarm Vector");
      try {
      FileOutputStream fout = new FileOutputStream("AlarmList.dat");
      ObjectOutputStream oos = new ObjectOutputStream(fout);
      oos.writeObject(alarms);
      oos.close();
      result = true;
      }
      catch (Exception e)
      {
         result = false;
         e.printStackTrace();
         return result;
      }
      return result;
    }

/*
* Method called to read the file AlarmList.dat from disk and restore
* the state of the system
*/
    static public boolean read()
    {
       alarms = new Vector();
       boolean result = false;
       // unserialize the vector
       System.out.println("unserializing AlarmList");
       try {
      FileInputStream fin = new FileInputStream("AlarmList.dat");
      ObjectInputStream ois = new ObjectInputStream(fin);
      alarms = (Vector) ois.readObject();
      ois.close();
      result = true;
      }
       catch (Exception e)
      {
        e.printStackTrace();
        result = false;
        return result;
      }
       return result;
```

```
       }
/*
* Method used for adding elements of type Alarm into the data structure
*/
    static public void insert(Alarm m)
    {
      //System.out.println("insert Alarm called");
      GregorianCalendar alarm = m.getAlarmDate();
      GregorianCalendar current = CurrentDate.getCurrentCalendar();
      int length = alarms.size() ;
      int index  = 0;
      if(alarm.before(current))
          {
            return;
          }
      while(index<=length)
          {
            if(length == 0 || index == length)
                {
                  alarms.insertElementAt(m,index);
                  AlarmList.updateState();
                  return;
                }
            GregorianCalendar alarmListCalendar =
((Alarm)alarms.elementAt(index)).getAlarmDate();
            if(alarm.before(alarmListCalendar) )
                {
                  alarms.insertElementAt(m,index);
                  AlarmList.updateState();
                  return;
                }
            else
                {
                  index++;
                }
          }
    }

/*
* Method used for adding elements of type Alarm into a temporary data
structure,
* used internally for sorting puposes.
*/
     static public void insert(Alarm m, Vector v)
    {
      // this is used only for resorting
      GregorianCalendar alarm = m.getAlarmDate();
      GregorianCalendar current = CurrentDate.getCurrentCalendar();
      int length = v.size() ;
      int index  = 0;
      if(alarm.before(current))
          {
            return;
          }
      while(index<=length)
          {
            if(length == 0 || index == length)
```

```
                {
                  v.insertElementAt(m,index);
                  return;
                }
             GregorianCalendar alarmListCalendar =
((Alarm)v.elementAt(index)).getAlarmDate();
             if(alarm.before(alarmListCalendar) )
                {
                  v.insertElementAt(m,index);
                  return;
                }
             else
                {
                  index++;
                }
          }
          AlarmList.updateState();
    }

/*
* Method used for deleting elements of type Alarm into the data
structure
*/
    static public void delete(Alarm m)
    {
      boolean happened = alarms.removeElement(m);
      d("the alarm was found and deleted " + happened);
      AlarmList.updateState();
    }
/*
* By specifying an object of type Meeting, all objects of type Alarm
within
* the data structure that refer to that Meeting will be deleted.
*/
    static public void delete(Meeting m) {
      GregorianCalendar meetCalendar = m.getMeetingDate();
      int index = 0;
      int length = alarms.size() - 1;
      while(index <= length) {
          GregorianCalendar alarmCalendar =
((Alarm)(alarms.elementAt(index))).getMeetingDate();
          if(meetCalendar.equals(alarmCalendar) ) {
            alarms.removeElementAt(index);
            length--;
          }
      }
      AlarmList.updateState();
    }
    public static void x(String d)
    {
      //System.out.print(d);
    }

    /*
    * This method queries all the elements within the data structure
and
```

```
    * returns the Alarm object that would trigger next from the current
time.
    */
    static public Alarm next() {  //still need to adjust for a few
second difference here
      if(alarms.size() == 0)
          {
            d("alarms list is empty");
            return(invalid());
          }
      GregorianCalendar currentCalendar =
CurrentDate.getCurrentCalendar();

      //currentCalendar.add(GregorianCalendar.SECOND, 1000);
      Alarm temp = (Alarm)alarms.elementAt(0);
      GregorianCalendar firstAlarmFromCalendar = temp.getAlarmDate();

      if(currentCalendar.before(firstAlarmFromCalendar)) {
          d(" No Alarms for the moment");

          x(" \n this is the date of the current alarm \n");
          x(" YEAR: " +
firstAlarmFromCalendar.get(firstAlarmFromCalendar.YEAR));
          x(" MONTH: " +
firstAlarmFromCalendar.get(firstAlarmFromCalendar.MONTH));
          x(" DAY_OF_MONTH: " +
firstAlarmFromCalendar.get(firstAlarmFromCalendar.DAY_OF_MONTH));
          x(" HOUR_OF_DAY: " +
firstAlarmFromCalendar.get(firstAlarmFromCalendar.HOUR_OF_DAY));
          x(" MINUTE: " +
firstAlarmFromCalendar.get(firstAlarmFromCalendar.MINUTE));
          x(" SECOND: " +
firstAlarmFromCalendar.get(firstAlarmFromCalendar.SECOND));
           x(" \n this is the date of the current date\n");
           x(" YEAR: " + currentCalendar.get(currentCalendar.YEAR));
           x(" MONTH: " + currentCalendar.get(currentCalendar.MONTH));
           x(" DAY_OF_MONTH: " +
currentCalendar.get(currentCalendar.DAY_OF_MONTH));
           x(" HOUR_OF_DAY: " +
currentCalendar.get(currentCalendar.HOUR_OF_DAY));
           x(" MINUTE: " +
currentCalendar.get(currentCalendar.MINUTE));
           x(" SECOND: " +
currentCalendar.get(currentCalendar.SECOND)+"\n\n\n");
          return(invalid());
      }
      //this should actually be within 5 minutes of current time
      if(currentCalendar.after(firstAlarmFromCalendar))
          {
            Alarm temp2 = (Alarm)alarms.elementAt(0);
            alarms.removeElementAt(0);
            // System.out.println("\n\n\nreturning Alarm with Date" +
temp2.getAlarmDate());
            return(temp2);
          }
      else {
          d("reached end of AlarmList");
```

```
            return(invalid());
        }
    }
    static public void d(String m) {
        if(debug == true) {
        System.out.println(m);
        }
    }

}
```

IX.3 AlertGUI.java

```
/**
 * AlertGIO Class displays a message alerting the user of an upcoming
event
 * Called by Alarm
 * author Marc Bourget
 */
package SmartReminder;

import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.lang.*;

public class AlertGUI extends javax.swing.JFrame
{
      public AlertGUI()
      {
            getContentPane().setLayout(null);
            setSize(240,320);
            setVisible(false);
            AlertL.setText("Meeting Name");
            getContentPane().add(AlertL);
            AlertL.setFont(new Font("Dialog", Font.BOLD, 16));
            AlertL.setBounds(15,12,120,30);
            MessageL.setEditable(false);
            getContentPane().add(MessageL);
            MessageL.setBounds(15,45,210,80);
            OkB.setBounds(85,150,70,25);
            getContentPane().add(OkB);
            setTitle("REMINDER");

            SymMouse aSymMouse = new SymMouse();
            OkB.addMouseListener(aSymMouse);
      }

    /**
     * Called by the alarm to display the alert message with the proper
info
     */
    public AlertGUI(String name, GregorianCalendar mDate, boolean
smart, String loc) {
      this();
      AlertL.setText(name);
```

```
      if ( smart ) {
        MessageL.setText("This appointment is at "
                  + mDate.get(mDate.HOUR_OF_DAY)
                  + ":00 on " + mDate.get(mDate.MONTH)
                  + "/" + mDate.get(mDate.DATE) + "/"
                  + mDate.get(mDate.YEAR) + "\nat "
                  + loc + ".\n   You should leave now.");
      }
      else {
      MessageL.setText("This appointment is at "
                    + mDate.get(mDate.HOUR_OF_DAY)
                  + ":00 on " + mDate.get(mDate.MONTH)
                  + "/" + mDate.get(mDate.DATE) + "/"
                  + mDate.get(mDate.YEAR) +"\nat "+loc);
      }
    }

    public void setVisible(boolean b) {
      if (b)
        setLocation(50, 50);
      super.setVisible(b);
    }

    static public void main(String args[]) {
      GregorianCalendar today = new GregorianCalendar(2001, 4, 20, 7,
30);
      (new AlertGUI("Baseball Game", today, false,
"Fenway")).setVisible(true);
    }

    public void addNotify() {
      Dimension size = getSize();
      super.addNotify();
      if (frameSizeAdjusted)
        return;
      frameSizeAdjusted = true;
      Insets insets = getInsets();
      javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
      int menuBarHeight = 0;
      if (menuBar != null)
        menuBarHeight = menuBar.getPreferredSize().height;
      setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
    }

    boolean frameSizeAdjusted = false;
    javax.swing.JLabel AlertL = new javax.swing.JLabel();
    javax.swing.JTextPane MessageL = new javax.swing.JTextPane();
    javax.swing.JButton OkB = new javax.swing.JButton("OK");

    class SymMouse extends java.awt.event.MouseAdapter {
      public void mouseClicked(java.awt.event.MouseEvent event) {
        Object object = event.getSource();
        if (object == OkB)
          setVisible(false);
      }
    }
```

```
}
```

IX.4 ALLLocGUI.java

```
/**
 * AllLocGUI allows the user to view, edit, add, and delete location
entries
 *
 * author Marc Bourget
 */

package SmartReminder.Location;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.*;

public class AllLocGUI extends javax.swing.JFrame
{
    private Font f = new Font("dialog",Font.PLAIN,10);
    private boolean name = false;
    private boolean info = false;
    private boolean showInfo = true;

    public AllLocGUI() {
      setTitle("AllLocGUI");
      getContentPane().setLayout(null);
      setSize(240,320);
      setVisible(true);

      locP.setBounds(15,85,110,15);
      getContentPane().add(locP);
      locP.setFont(f);

      loc_list = new javax.swing.JComboBox(Location.LIST_LOC);
      loc_list.setBounds(85,25,130,15);
      loc_list.setFont(f);
      getContentPane().add(loc_list);

      getContentPane().add(loc_L);
      loc_L.setBounds(15,25,50,20);
      loc_L.setFont(f);

      getContentPane().add(message);
      message.setBounds(15,255,210,20);

      loc.setBounds(85,50,130,15);
      getContentPane().add(loc);
      loc.setFont(f);
      loc.setVisible(false);

      newData.setBounds(85,140,130,15);
      getContentPane().add(newData);
      newData.setFont(f);
      newData.setVisible(false);
```

```
/**
 * IP Fields
 */
key_L.setText("IP Key");
getContentPane().add(key_L);
key_L.setBounds(15,115,50,15);
key_L.setFont(f);

raw.setBounds(85,115,130,15);
getContentPane().add(raw);
raw.setFont(f);
raw.setEditable(false);

/**
 * Cricket Fields
 */
getContentPane().add(beaconID_L);
beaconID_L.setBounds(15,115,50,15);
beaconID_L.setFont(f);
beaconID_L.setVisible(false);

loc0.setSelected(true);
loc0.setBounds(70,25,12,12);
getContentPane().add(loc0);
loc0.setVisible(false);

loc1.setBounds(70,50,12,12);
getContentPane().add(loc1);
loc1.setVisible(false);

data0.setSelected(true);
data0.setBounds(70,115,12,12);
getContentPane().add(data0);
data0.setVisible(false);

data1.setBounds(70,140,12,12);
getContentPane().add(data1);
data1.setVisible(false);

getContentPane().add(Edit_Btn);
Edit_Btn.setBounds(15,170,75,15);
Edit_Btn.setFont(f);

getContentPane().add(Remove_Btn);
Remove_Btn.setBounds(150,170,75,15);
Remove_Btn.setFont(f);

getContentPane().add(Save_Btn);
Save_Btn.setBounds(15,170,75,15);
Save_Btn.setFont(f);
Save_Btn.setVisible(false);

getContentPane().add(Cancel_Btn);
Cancel_Btn.setBounds(150,170,75,15);
Cancel_Btn.setFont(f);
Cancel_Btn.setVisible(false);
```

```
      getContentPane().add(Close_Btn);
      Close_Btn.setBounds(150,200,70,15);
      Close_Btn.setFont(f);

      getContentPane().add(New_Btn);
      New_Btn.setBounds(15,200,70,15);
      New_Btn.setFont(f);

      SymMouse aSymMouse = new SymMouse();
      SymAction aSymAction = new SymAction();
      Edit_Btn.addMouseListener(aSymMouse);
      New_Btn.addMouseListener(aSymMouse);
      Close_Btn.addMouseListener(aSymMouse);
      Remove_Btn.addMouseListener(aSymMouse);
      Save_Btn.addMouseListener(aSymMouse);
      Cancel_Btn.addMouseListener(aSymMouse);
      loc0.addMouseListener(aSymMouse);
      loc1.addMouseListener(aSymMouse);
      data0.addMouseListener(aSymMouse);
      data1.addMouseListener(aSymMouse);
      locP.addActionListener(aSymAction);
      loc_list.addActionListener(aSymAction);

      update((String)locP.getSelectedItem());
    }

    /**
     * This method resets the GUI to its initial state = displays
appropriate
     * components
     */
    public void clear() {
      showInfo = true;
      raw.setEditable(false);
      update((String)locP.getSelectedItem());
      loc.setVisible(false);
      data0.setVisible(false);
      data1.setVisible(false);
      newData.setVisible(false);
      loc0.setVisible(false);
      loc1.setVisible(false);
      Save_Btn.setVisible(false);
      Cancel_Btn.setVisible(false);
      Edit_Btn.setVisible(true);
      New_Btn.setVisible(true);
      Remove_Btn.setVisible(true);
      newData.setText("");
      loc.setText("");
      message.setText("");
    }

    /**
     * Display the proper info depending on what the user has selected
from the drop-down list
     */
    public void update(String s) {
      if (showInfo) {
```

```
      if (s.equals("IP info")) {
        if
(Location.LOC_ALL.contains((String)loc_list.getSelectedItem())) {

raw.setText((String)Location.IP_ALL.elementAt(Location.LOC_ALL.indexOf(
loc_list.getSelectedItem())));
        }
        else {
          raw.setText("unknown");
        }
      }
      else if (s.equals("Cricket info")) {
        if
(Location.CRICKET_LOCS.contains((String)loc_list.getSelectedItem())) {

raw.setText((String)Location.CRICKET_BEACON_IDS.elementAt(Location.CRIC
KET_LOCS.indexOf(loc_list.getSelectedItem())));
        }
        else {
          raw.setText("unknown");
        }
       }
      }
    }

    public void setVisible(boolean b) {
      if (b)
      setLocation(50, 50);
      super.setVisible(b);
    }

    static public void main(String args[]) {
      (new AllLocGUI()).setVisible(true);
    }

    public void addNotify() {
      Dimension size = getSize();
      super.addNotify();
      if (frameSizeAdjusted)
      return;
      frameSizeAdjusted = true;

      Insets insets = getInsets();
      javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
      int menuBarHeight = 0;
      if (menuBar != null)
      menuBarHeight = menuBar.getPreferredSize().height;
      setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
    }

    boolean frameSizeAdjusted = false;
    String[] data = {"IP info", "Cricket info"};

    javax.swing.JTextArea raw = new javax.swing.JTextArea();
    javax.swing.JLabel beaconID_L = new javax.swing.JLabel("BeaconID");
    javax.swing.JLabel key_L = new javax.swing.JLabel();
```

```
    javax.swing.JLabel loc_L = new javax.swing.JLabel("Location");
    javax.swing.JLabel message = new javax.swing.JLabel();
    javax.swing.JTextArea loc = new javax.swing.JTextArea();
    javax.swing.JTextArea newData = new javax.swing.JTextArea();
    javax.swing.JRadioButton loc0 = new javax.swing.JRadioButton();
    javax.swing.JRadioButton loc1 = new javax.swing.JRadioButton();
    javax.swing.JRadioButton data0 = new javax.swing.JRadioButton();
    javax.swing.JRadioButton data1 = new javax.swing.JRadioButton();
    javax.swing.JComboBox locP = new javax.swing.JComboBox(data);
    javax.swing.JComboBox loc_list = new
javax.swing.JComboBox(Location.LIST_LOC);
    javax.swing.JButton Edit_Btn = new javax.swing.JButton("Edit");
    javax.swing.JButton Close_Btn = new javax.swing.JButton("Close");
    javax.swing.JButton Remove_Btn = new javax.swing.JButton("Remove");
    javax.swing.JButton Save_Btn = new javax.swing.JButton("Save");
    javax.swing.JButton Cancel_Btn = new javax.swing.JButton("Cancel");
    javax.swing.JButton New_Btn = new javax.swing.JButton("New");

    class SymMouse extends java.awt.event.MouseAdapter {
      public void mouseClicked(java.awt.event.MouseEvent event) {
        Object object = event.getSource();
        if (object == Edit_Btn)
        Edit_Btn_mouseClicked(event);
      if (object == New_Btn)
        New_Btn_mouseClicked(event);
      if (object == Close_Btn)
        Close_Btn_mouseClicked(event);
      if (object == Remove_Btn)
        Remove_Btn_mouseClicked(event);
      if (object == Save_Btn)
        Save_Btn_mouseClicked(event);
      if (object == Cancel_Btn)
        clear();
      if (object == loc0) {
        name = false;
        loc0.setSelected(true);
        loc1.setSelected(false);
      }
      if (object == loc1) {
        name = true;
        loc0.setSelected(false);
        loc1.setSelected(true);
      }
      if (object == data0) {
        info = false;
        data0.setSelected(true);
        data1.setSelected(false);
      }
      if (object == data1) {
        info = true;
        data0.setSelected(false);
        data1.setSelected(true);
      }
      }
    }

    class SymAction implements ActionListener {
```

```
      public void actionPerformed(ActionEvent e) {
      JComboBox cb = (JComboBox)e.getSource();
      if (cb == locP) {
        clear();
        if (((String)cb.getSelectedItem()).equals("IP info")) {
          key_L.setVisible(true);
          beaconID_L.setVisible(false);
        }
        else if (((String)cb.getSelectedItem()).equals("Cricket info"))
{
          key_L.setVisible(false);
          beaconID_L.setVisible(true);
        }
        else if (((String)cb.getSelectedItem()).equals("Wireless
info")) {
          key_L.setVisible(false);
          beaconID_L.setVisible(false);
        }
      }
      else if (cb==loc_list){
        update((String)locP.getSelectedItem());
      }
      }
    }

    void Edit_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      name = info = false;
      loc0.setSelected(true);
      loc1.setSelected(false);
      data0.setSelected(true);
      data1.setSelected(false);
      showInfo = false;
      loc.setVisible(true);
      newData.setVisible(true);
      loc0.setVisible(true);
      loc1.setVisible(true);
      loc.setVisible(true);
      data0.setVisible(true);
      data1.setVisible(true);
      Save_Btn.setVisible(true);
      Cancel_Btn.setVisible(true);
      Remove_Btn.setVisible(false);
      Edit_Btn.setVisible(false);
      New_Btn.setVisible(false);
    }

    void New_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      name = info = false;
      loc0.setSelected(true);
      loc1.setSelected(false);
      showInfo = false;
      loc.setVisible(true);
      newData.setVisible(false);
      loc0.setVisible(true);
      loc1.setVisible(true);
      loc.setVisible(true);
      raw.setText("");
```

```
      raw.setEditable(true);
      data0.setVisible(false);
      data1.setVisible(false);
      Save_Btn.setVisible(true);
      Cancel_Btn.setVisible(true);
      Remove_Btn.setVisible(false);
      Edit_Btn.setVisible(false);
      New_Btn.setVisible(false);
    }

    void Remove_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      if (((String)locP.getSelectedItem()).equals("IP info")) {
      IPLocProvider.removeLocation(raw.getText());
      Group.remove((String)loc_list.getSelectedItem());
      update((String)locP.getSelectedItem());
      }
      else if (((String)locP.getSelectedItem()).equals("Cricket info"))
{
      CricketLocProvider.removeLocation(raw.getText());
      Group.remove((String)loc_list.getSelectedItem());
      update((String)locP.getSelectedItem());
      }
    }

    void Save_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      if ( name && (loc.getText().equals("")) ) {
      message.setText("Invalid location name");
      }
      else if ( info && (newData.getText().equals("")) ) {
      message.setText("Invalid location entry");
      }
      else if (!info && (raw.getText().equals("unknown") ||
raw.getText().equals("")) )  {
      message.setText("Invalid location entry");
      }
      else {
      String n,l;
      if (name) {
        n = loc.getText();
      }
      else {
        n = (String)loc_list.getSelectedItem();
      }
      if (info) {
        l = newData.getText();
      }
      else {
        l = raw.getText();
      }
      if (((String)locP.getSelectedItem()).equals("IP info")) {
        IPLocProvider.editLocation(l,n);
      }
      else if (((String)locP.getSelectedItem()).equals("Cricket info"))
{
        CricketLocProvider.editLocation(l,n);
      }
      clear();
```

```
      }
    }

    void Close_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      this.setVisible(false);
    }
}
```

IX.5 CricketLocProvider.java

```
/**
 * This class spawns a cricketdaemon object which is specified in code
 * written by Allen Miu with the given ip and provides a method
 * called getLocation that will return a point with the player's
current
 * location.
 *
 * @author Craig Music
 */


package SmartReminder.Location;

import cricketdaemon.clientlib.*;
import cricketdaemon.clientlib.data.*;
import java.net.*;
import java.io.*;
import javax.swing.table.*;
import java.util.*;
import beaconfinder.Util;
import java.awt.Point;
import java.util.Hashtable;
import SmartReminder.*;


public class CricketLocProvider implements Runnable, Callback
{
    public String currentBeacon = "";
    public static Random r;
    Broker cricket;
    private long timeLastUpdate;
    private Object[] beaconData, calcBeaconData, orderedCalcBeaconData;
    public int curBeaconInd;
    private static final long TIMEOUT = 20000;
    public String[] cricketargs;
    public static String location= "Not Available";
    public String locationName;
    public Hashtable locationHashtable;
    public boolean receivedCoords;
    public long startUpTime;
    private static long locationTime;
    public boolean d = false;

    public static void main(String[] args)
    {
```

```
      //String[] a = new String[] {"-g"+IP, "-S", "-P", "-
m600in.trans"};
      new CricketLocProvider(args);
    }
    public void run()
    {
      try
          {
            cricketdaemon.CricketDaemon cd = new
cricketdaemon.CricketDaemon(cricketargs); // this line invokes the
crickdaemon object from Allen Miu's code
          }
      catch(Exception e)
          {
            System.out.println("/n/n/n/n/n/n There has been an error
withing Player and cricketdaemon has thrown an exeption, will return
null from now on");
            e.printStackTrace();
          }
    }

    public CricketLocProvider(String[] args)
    {
      cricketargs = args;
      Thread t = new Thread(this);
      t.start();
      r = new Random(0);
      Broker s  = new ServerBroker();
      BitSet mask = new BitSet();
      mask.set(Broker.BEACONSTAT);
      mask.set(Broker.CALCBEACONDIST);
      s.register(this, mask);
      s.start();
      s.interrupt();
    }

    private boolean timeOut(Long lastUpdate) {
      if(lastUpdate == null)
          return false;

      return
          (lastUpdate.longValue() + TIMEOUT) <
CurrentDate.getCurrentTimeMillis();
    }

    private Object[] orderData(Object[] target, Object[] source)
    {
      Object[] result = new Object[target.length];

      for(int i = 0; i < result.length; i++) {
          Associable t = (Associable) target[i];
          for(int j = 0; j < source.length; j++) {
            Associable s = (Associable) source[j];
            if(s.isAssociated(t)) {
                result[i] = source[j];
                break;
            }
```

```
            }
        }
        return result;
    }

    private void updateBeaconsHeard()
    {
        if(beaconData == null) {
            curBeaconInd = 0;
            return;
        }
        ArrayList result = new ArrayList();
        for(int i = 0; i < beaconData.length; i++) {
            BeaconRecord br = (BeaconRecord) beaconData[i];
            if(!timeOut(br.lastUpdate))
              result.add(beaconData[i]);
        }
        beaconData = result.toArray();
        Arrays.sort(beaconData);
        curBeaconInd = findClosest(beaconData);
        if(calcBeaconData != null)
            orderedCalcBeaconData = orderData(beaconData,
calcBeaconData);
        else
            orderedCalcBeaconData = null;
    }

    private int findClosest(Object[] o)
    {
        double curMin = Double.MAX_VALUE;
        int curMinIndex = -1;

        for(int i = 0; i < o.length; i++) {
            BeaconRecord r = (BeaconRecord) o[i];
            double comp = r.distStat.mode;
            if(comp < curMin) {
              curMin = comp;
              curMinIndex = i;
            }
        }
        return curMinIndex;
    }

    private Object[] makeSet(Object[] orig, ArrayList more)
    {
        if(orig != null) {
            for(int i = 0; i < orig.length; i++) {
              if(!more.contains((BeaconRecord)orig[i]))
                  more.add(orig[i]);
            }
        }
        return more.toArray();
    }

    synchronized public void callback(CricketData data, BitSet mask)
    {
```

```
      //System.out.println("\****8CALLBACK CALLED
***************\n\n\n\n");

      if(mask.get(Broker.BEACONSTAT)) {
          ArrayList beaconsHeard = data.getBeaconsHeard();
          beaconData = makeSet(beaconData, beaconsHeard);
          receivedCoords = true;
          locationTime = CurrentDate.getCurrentTimeMillis();
      }
      updateBeaconsHeard();
      currentBeacon = (String)
((BeaconRecord)beaconData[curBeaconInd]).getUniqueIdentifier();
      location = currentBeacon;
      notify();
      if(d) {
        System.out.println("Cricket current location is this: " +
location);
        }
    }

    /*
    * Method that when queried will return a String representation of
the
    * system's current location.
    */
    static public String getLocation()
    {
      long currentTime = CurrentDate.getCurrentTimeMillis();
      System.out.println("old time: " + locationTime + " CurrentTime: "
+ currentTime);
      if(currentTime - locationTime > 10000)
      {
        location = "Not Available";
      }
      return location;
    }

    static public String getCurrentLoc() {
      String s = getLocation();
      String temp = getLoc(s);
      System.out.println("According to CricketLocProvider the location
is: " + temp);
      return(temp);
    }

    /*
    * This method initializes the CricketLocationProvider by reading
the
    * mapping between beacon id's and location names from the file
CricketTable
    */
    static public void init()
    {
      DataInputStream input;
      StringBuffer b = new StringBuffer();
      String data = "";
      int charint = 0;
```

```
      //vectors made from columns in CricketTable
      //---------------
      try
      {
          input = new DataInputStream(new FileInputStream(
"CricketTable" ) );
          while((charint=input.read())!=-1) {
            b.append((char)charint);
          }
      }
      catch (IOException e) {
          System.err.println( "Error opening or reading file\n" +
e.toString() );
          System.exit(1);
      }
      data = b.toString();
      // first separate the data into individual lines
      StringTokenizer st = new StringTokenizer(data, "\n\r");
      StringTokenizer lt;
      String line,beaconID,loc;

      while(st.hasMoreTokens())
      {
          line = st.nextToken();
          lt = new StringTokenizer(line, " \t");
          beaconID = lt.nextToken();
          loc = lt.nextToken();

          while(lt.hasMoreTokens()) {
            loc = loc + " " + lt.nextToken();
          }

          Location.CRICKET_LOCS.add(loc);
          Location.CRICKET_BEACON_IDS.add(beaconID);

          // Don't allow duplicates in the location and key (ip) list
          if (!Location.LIST_LOC.contains(loc)) {
            Location.LIST_LOC.add(loc);
          }
      }

    }

/*
 * Finds the location for the ip iputted as a string if it exists
 */
    static public String getLoc(String s)
    {
      if (Location.CRICKET_BEACON_IDS.contains(s))
      {
      return((String)Location.CRICKET_LOCS.elementAt(Location.CRICKET_B
EACON_IDS.indexOf(s)));
      }
      else
      {
      return("unknown");
      }
```

```
    }

/*
 * Removes the record for the location specified by String beaconID
 */
    static public void removeLocation(String beaconID) {
      //remove pair from the table
      //table.remove(key);
      //remove pair from the vector

      if (Location.CRICKET_BEACON_IDS.contains(beaconID)) {
      int m = Location.CRICKET_BEACON_IDS.indexOf(beaconID);
        Location.CRICKET_BEACON_IDS.removeElementAt(m);
        Location.CRICKET_LOCS.removeElementAt(m);
        // remove from the file by saving the new vectors
        String data = "";
        for (int i=0; i<Location.CRICKET_BEACON_IDS.size(); i++) {
           data = data + "\n" + Location.IP_ALL.elementAt(i) + "\t" +
Location.LOC_ALL.elementAt(i);
        }

        try {
          //FileWriter fw = new FileWriter(
"/java/SmartReminder/Location/IPtable", false );
          FileWriter fw = new FileWriter( "CricketTable", false );
          fw.write(data);
          fw.close();
          //after saving the new file, we should reinitialize
          CricketLocProvider.init();
        }
        catch (IOException e) {
          System.err.println( "Error opening or writing to file\n" +
e.toString() );
          System.exit(1);
        }
      }
      else {
      System.err.println("BeaconID: "+beaconID+" not found");
      }
    }


/*
 * Edits the record for the location specified by String beaconID
 */
    static public void editLocation(String beaconID, String loc) {
      //if neither the beaconID nor loc exist in the two vectors, treat
them as a new location entry
      if ( !(Location.CRICKET_BEACON_IDS.contains(beaconID)) &&
!(Location.CRICKET_LOCS.contains(loc)) ) {
      addLocation(beaconID,loc);
      System.out.println("New Location added");
      }
      else {
      int m;
      if (Location.CRICKET_BEACON_IDS.contains(beaconID)) {
```

```
        m = Location.CRICKET_BEACON_IDS.indexOf(beaconID);
        Location.CRICKET_LOCS.set(m, loc);
        Location.CRICKET_BEACON_IDS.set(m, beaconID);
        System.out.println("Location name editted");
      }
      else if (Location.CRICKET_LOCS.contains(loc)) {
        m = Location.CRICKET_LOCS.indexOf(loc);
        Location.CRICKET_LOCS.set(m, loc);
        Location.CRICKET_BEACON_IDS.set(m, beaconID);
        System.out.println("Location beaconID editted");
      }
      else {
        System.exit(1);
        System.err.print("Error with location edit");
      }

      String data = "";
      for (int i=0; i<Location.CRICKET_BEACON_IDS.size(); i++) {
        data = data + "\n" + Location.CRICKET_BEACON_IDS.elementAt(i) +
"\t" + Location.CRICKET_LOCS.elementAt(i);
      }
      try {
        //FileWriter fw = new FileWriter(
"/java/SmartReminder/Location/IPtable", false );
        FileWriter fw = new FileWriter( "CricketTable", false );
        fw.write(data);
        fw.close();
        //after saving the new file, we should reinitialize
        init();
      }
      catch (IOException e) {
        System.err.println( "Error opening or writing to file\n" +
e.toString() );
        System.exit(1);
      }
      }
    }


/*
 * Add the record for the location specified by String beaconID
 */
    static public void addLocation(String beaconID, String location) {
      String data = "";
      try {
          //          FileWriter fw = new FileWriter(
"/java/SmartReminder/Location/IPtable", true );
          FileWriter fw = new FileWriter( "CricketTable", true );
          data = "\n"+beaconID+"\t"+location;
          fw.write(data);
          fw.close();
      }
      catch (IOException e) {
          System.err.println( "Error opening or writing to file\n" +
e.toString() );
          System.exit(1);
      }
```

```
      // add location pair to the vectors
      Location.CRICKET_LOCS.add(location);
      Location.CRICKET_BEACON_IDS.add(beaconID);

      if (!Location.LIST_LOC.contains(location)) {
            Location.LIST_LOC.add(location);
      }

    }
}
```

IX.6 CurrentDate.java

```
/**
 * Class used by the system to obtain date information
 *
 * @author Craig Music
 */
package SmartReminder;
import java.util.*;

public  class CurrentDate {

  static int offset = SR.defaultOffset; // this is in hours

  /**
  * This method is used by other classes in our system that want
  * to access the current time and obtain in a java GregorianCalendar
object
  */
  public static GregorianCalendar getCurrentCalendar() {
      GregorianCalendar tempy = new GregorianCalendar();
        long realOffset = offset*60*60*1000;
        Date myDate = new Date(System.currentTimeMillis() +
realOffset);
        tempy.setTime(myDate);
        //System.out.println("\n"+myDate.toString()+"\n");
          return tempy;
    }

    /*
  * This method is used by other classes in our system that want
  * to access the current time and obtain in milliseconds.
  */
  public static long getCurrentTimeMillis() {
     long realOffset = offset*60*60*1000;
     Date myDate = new Date(System.currentTimeMillis() + realOffset);
     //System.out.println("\n"+myDate.toString()+"\n");

      return System.currentTimeMillis() + offset*60*60*1000;
    }
}
```

IX.7 DayGUI.java

```
/**
 * Class used as one of the GUI's of the system. It displays a
scheduler
 * style interface where the user can scroll through days and see a
visual
 * representation of Meeting objects which have been inputed in the
system.
 *
 * @author Craig Music
 */
package SmartReminder;
import SmartReminder.Location.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.*;


/**
 * A basic JFC 1.1 based application.
 */
public class DayGUI extends javax.swing.JFrame
{
    private int TEXT_HEIGHT = 18;
    private int TEXT_WIDTH = 170;
    private int TEXT_START = 39;
    private int TEXT_VER_START = 39;
    private int TEXT_HOR_START = 72;
    private GregorianCalendar today;
    private String seven = "";
    private String eight = "";
    private String nine= "";
    private String ten = "";
    private String eleven = "";
    private String twelve="";
    private String thirteen="";
    private String fourteen="";
    private String fifteen="";
    private String sixteen="";
    private String seventeen="";
    private String eighteen ="";
    private String nineteen="";
    private String twenty="";
    private String twentyone ="";
    private javax.swing.JTextPane EightText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane NineText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane TenText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane ElevenText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane TwelveText = new
javax.swing.JTextPane();
```

```
    private javax.swing.JTextPane ThirteenText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane FourteenText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane FifteenText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane SixteenText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane SeventeenText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane EighteenText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane NineteenText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane TwentyText = new
javax.swing.JTextPane();
    private javax.swing.JTextPane TwentyOneText = new
javax.swing.JTextPane();
    private String day;
    private boolean debug = false;
    static public boolean flag = true;
    public Font f = new Font("dialog", Font.PLAIN,10);
    public Font ff = new Font("dialog", Font.PLAIN,11);
    private Meeting copyForDelete;

    public DayGUI()
    {

      // set the current date
      today = CurrentDate.getCurrentCalendar();

      //get back here
      this.day = parseCalendar(today);

      setTitle("SMART REMINDER");
      setDefaultCloseOperation(javax.swing.JFrame.DO_NOTHING_ON_CLOSE);
      getContentPane().setLayout(null);
      getContentPane().setBackground(java.awt.Color.lightGray);
      setSize(240,320);
      setVisible(false);
      EightButton.setText("8:00");
      EightButton.setFont(ff);
      EightButton.setActionCommand("8:00");
      getContentPane().add(EightButton);
      EightButton.setBounds(0,39,73,18);
      NineButton.setText("9:00");
      NineButton.setActionCommand("9:00");
      NineButton.setFont(ff);
      getContentPane().add(NineButton);
      NineButton.setBounds(0,57,73,18);
      TenButton.setText("10:00");
      TenButton.setActionCommand("10:00");
      getContentPane().add(TenButton);
      TenButton.setBounds(0,75,73,18);
      TenButton.setFont(ff);
      ElevenButton.setText("11:00");
      ElevenButton.setActionCommand("11:00");
```

```
getContentPane().add(ElevenButton);
ElevenButton.setBounds(0,93,73,18);
ElevenButton.setFont(ff);
TwelveButton.setText("12:00");
TwelveButton.setActionCommand("12:00");
getContentPane().add(TwelveButton);
TwelveButton.setBounds(0,111,73,18);
TwelveButton.setFont(ff);
ThirteenButton.setFont(ff);
ThirteenButton.setText("13:00");
ThirteenButton.setActionCommand("13:00");
getContentPane().add(ThirteenButton);
ThirteenButton.setBounds(0,129,73,18);
FourteenButton.setText("14:00");
FourteenButton.setActionCommand("14:00");
getContentPane().add(FourteenButton);
FourteenButton.setBounds(0,147,73,18);
FourteenButton.setFont(ff);
FifteenButton.setText("15:00");
FifteenButton.setActionCommand("15:00");
getContentPane().add(FifteenButton);
FifteenButton.setBounds(0,165,73,18);
FifteenButton.setFont(ff);
SixteenButton.setText("16:00");
SixteenButton.setActionCommand("16:00");
SixteenButton.setFont(ff);
getContentPane().add(SixteenButton);
SixteenButton.setBounds(0,183,73,18);
SeventeenButton.setText("17:00");
SeventeenButton.setActionCommand("17:00");
getContentPane().add(SeventeenButton);
SeventeenButton.setBounds(0,201,73,18);
SeventeenButton.setFont(ff);
EighteenButton.setText("18:00");
EighteenButton.setActionCommand("7:00");
getContentPane().add(EighteenButton);
EighteenButton.setBounds(0,219,73,18);
EighteenButton.setFont(ff);
NineteenButton.setText("19:00");
NineteenButton.setActionCommand("18:00");
getContentPane().add(NineteenButton);
NineteenButton.setBounds(0,237,73,18);
NineteenButton.setFont(ff);
TwentyButtton.setText("20:00");
TwentyButtton.setActionCommand("19:00");
TwentyButtton.setFont(ff);
getContentPane().add(TwentyButtton);
TwentyButtton.setBounds(0,255,73,18);
TwentyOneButton.setText("21:00");
TwentyOneButton.setFont(ff);
TwentyOneButton.setActionCommand("20:00");
getContentPane().add(TwentyOneButton);
TwentyOneButton.setBounds(0,273,73,18);

DateDisplay.setDisabledTextColor(java.awt.Color.black);
DateDisplay.setEditable(false);
getContentPane().add(DateDisplay);
```

```
        DateDisplay.setText(day);
        DateDisplay.setBackground(new java.awt.Color(204,204,204));
        DateDisplay.setBounds(40,6,160,25);
        getContentPane().add(ForwardButton);
        ForwardButton.setBounds(214,6,12,25);
        getContentPane().add(BackButton);
        BackButton.setBounds(14,6,12,25);
        EightText.setText(eight);
        EightText.setDisabledTextColor(java.awt.Color.black);
        EightText.setEditable(false);
        getContentPane().add(EightText);
        EightText.setBackground(new java.awt.Color(204,204,204));
        EightText.setBounds(TEXT_HOR_START,TEXT_VER_START + 0 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        NineText.setText(nine);
        NineText.setDisabledTextColor(java.awt.Color.black);
        NineText.setEditable(false);
        getContentPane().add(NineText);
        NineText.setBackground(new java.awt.Color(204,204,204));
        NineText.setBounds(TEXT_HOR_START,TEXT_VER_START + 1 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        TenText.setDisabledTextColor(java.awt.Color.black);
        TenText.setEditable(false);
        getContentPane().add(TenText);
        TenText.setText(ten);
        TenText.setBackground(new java.awt.Color(204,204,204));
        TenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 2 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        ElevenText.setDisabledTextColor(java.awt.Color.black);
        ElevenText.setEditable(false);
        getContentPane().add(ElevenText);
        ElevenText.setText(eleven);
        ElevenText.setBackground(new java.awt.Color(204,204,204));
        ElevenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 3 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        TwelveText.setDisabledTextColor(java.awt.Color.black);
        TwelveText.setEditable(false);
        getContentPane().add(TwelveText);
        TwelveText.setText(twelve);
        TwelveText.setBackground(new java.awt.Color(204,204,204));
        TwelveText.setBounds(TEXT_HOR_START,TEXT_VER_START + 4 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        ThirteenText.setDisabledTextColor(java.awt.Color.black);
        ThirteenText.setEditable(false);
        getContentPane().add(ThirteenText);
        ThirteenText.setText(thirteen);
        ThirteenText.setBackground(new java.awt.Color(204,204,204));
        ThirteenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 5 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        FourteenText.setDisabledTextColor(java.awt.Color.black);
        FourteenText.setEditable(false);
        getContentPane().add(FourteenText);
        FourteenText.setText(fourteen);
        FourteenText.setBackground(new java.awt.Color(204,204,204));
        FourteenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 6 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        FifteenText.setDisabledTextColor(java.awt.Color.black);
```

```
        FifteenText.setEditable(false);
        getContentPane().add(FifteenText);
        FifteenText.setText(fifteen);
        FifteenText.setBackground(new java.awt.Color(204,204,204));
        FifteenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 7 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        SixteenText.setDisabledTextColor(java.awt.Color.black);
        SixteenText.setEditable(false);
        getContentPane().add(SixteenText);
        SixteenText.setText(sixteen);
        SixteenText.setBackground(new java.awt.Color(204,204,204));
        SixteenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 8 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        SeventeenText.setDisabledTextColor(java.awt.Color.black);
        SeventeenText.setEditable(false);
        getContentPane().add(SeventeenText);
        SeventeenText.setText(seventeen);
        SeventeenText.setBackground(new java.awt.Color(204,204,204));
        SeventeenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 9 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        EighteenText.setDisabledTextColor(java.awt.Color.black);
        EighteenText.setEditable(false);
        getContentPane().add(EighteenText);
        EighteenText.setText(eighteen);
        EighteenText.setBackground(new java.awt.Color(204,204,204));
        EighteenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 10 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        NineteenText.setDisabledTextColor(java.awt.Color.black);
        NineteenText.setEditable(false);
        getContentPane().add(NineteenText);
        NineteenText.setText(nineteen);
        NineteenText.setBackground(new java.awt.Color(204,204,204));
        NineteenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 11 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        TwentyText.setDisabledTextColor(java.awt.Color.black);
        TwentyText.setEditable(false);
        getContentPane().add(TwentyText);
        TwentyText.setText(twenty);
        TwentyText.setBackground(new java.awt.Color(204,204,204));
        TwentyText.setBounds(TEXT_HOR_START,TEXT_VER_START + 12 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
        TwentyOneText.setDisabledTextColor(java.awt.Color.black);
        TwentyOneText.setEditable(false);
        getContentPane().add(TwentyOneText);
        TwentyOneText.setText(twentyone);
        TwentyOneText.setBackground(new java.awt.Color(204,204,204));
        TwentyOneText.setBounds(TEXT_HOR_START,TEXT_VER_START + 13 *
TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);

        SymWindow aSymWindow = new SymWindow();
        this.addWindowListener(aSymWindow);
        SymAction lSymAction = new SymAction();
        SymMouse aSymMouse = new SymMouse();
        EightButton.addMouseListener(aSymMouse);
        NineButton.addMouseListener(aSymMouse);
        TenButton.addMouseListener(aSymMouse);
        ElevenButton.addMouseListener(aSymMouse);
```

```
        TwelveButton.addMouseListener(aSymMouse);
        ThirteenButton.addMouseListener(aSymMouse);
        FourteenButton.addMouseListener(aSymMouse);
        FifteenButton.addMouseListener(aSymMouse);
        SixteenButton.addMouseListener(aSymMouse);
        SeventeenButton.addMouseListener(aSymMouse);
        EighteenButton.addMouseListener(aSymMouse);
        NineteenButton.addMouseListener(aSymMouse);
        TwentyButtton.addMouseListener(aSymMouse);
        TwentyOneButton.addMouseListener(aSymMouse);
        ForwardButton.addMouseListener(aSymMouse);
        BackButton.addMouseListener(aSymMouse);
        //}}
        updateNames(today);
    }
 /**
 * Class used by the class to obtain date information
 *
 */
    public GregorianCalendar getDate() {
      return today;
    }
/**
 * Utility method used to format a String to display the date of a
 * GregorianCalendar
 */
    public String parseCalendar(GregorianCalendar d) {
      String temp;
      String month = null;
      int mo = d.get(d.MONTH);

      if ( mo == 0)
          month = "January";
      else if (mo == 1)
          month = "February";
      else if (mo == 2)
          month = "March";
      else if ( mo == 3)
          month = "April";
      else if ( mo == 4)
          month = "May";
      else if (mo == 5)
          month = "June";
      else if (mo == 6)
          month = "July";
      else if ( mo == 7)
          month = "August";
      else if ( mo == 8)
          month = "September";
      else if (mo == 9)
          month = "October";
      else if (mo == 10)
          month = "November";
      else if ( mo == 11)
          month = "December";

      temp  = (month + " " + d.get(d.DATE) + ", " + d.get(d.YEAR) );
```

```
      return temp;
    }

    /**
     * Creates a new instance of JFrame1 with the given title.
    */
    public DayGUI(String sTitle)
    {
      this();
      setTitle(sTitle);
    }

    /**
     * Updates the display of the DayGUI to show the contents of all
meetings
     * in the system whose dates are the same as the current day.
     */
    public void updateNames(GregorianCalendar day) {
      int index = 8;
      String names[]= new String[23];
      for(int y = 0; y<23; y++) {
          names[y] = "";
      }

      while( index < 22) {
          int yr = today.get(today.YEAR);
          int mo = today.get(today.MONTH);
          int dt = today.get(today.DATE);
          int hr = index;

          GregorianCalendar gc = new GregorianCalendar(yr,mo,dt,hr,5);
          Meeting temp = MeetingList.search(gc);
          GregorianCalendar tt = temp.getMeetingDate();
          d("This is the hour of the rtuend meeting " +
tt.get(tt.HOUR_OF_DAY));
          GregorianCalendar day2 = temp.getMeetingDate();

          if (isMeetingValid(temp) ) {
            names[index] = temp.getName();
          }
          else {
            names[index] = "";
          }
          index++;
      }

          EightText.setText(names[8] );
          NineText.setText(names[9] );
          TenText.setText(names[10] );
          ElevenText.setText(names[11] );
          TwelveText.setText(names[12] );
          ThirteenText.setText(names[13]);
          FourteenText.setText(names[14]);
          FifteenText.setText(names[15]);
          SixteenText.setText(names[16]);
          SeventeenText.setText(names[17]);
          EighteenText.setText(names[18]);
```

```
            NineteenText.setText(names[19]);
            TwentyText.setText(names[20]);
            TwentyOneText.setText(names[21]);

    }


    /**
     * The entry point for this application.
     * Sets the Look and Feel to the System Look and Feel.
     * Creates a new JFrame1 and makes it visible.
     */
    static public void main(String args[])
    {
      try {

            //Create a new instance of our application's frame, and make
it visible.
            new DayGUI().setVisible(true);
      }
      catch (Throwable t) {
            t.printStackTrace();
            //Ensure the application exits with an error condition.
            System.exit(1);
      }
    }

    /**
     * Notifies this component that it has been added to a container
     * This method should be called by <code>Container.add</code>, and
     * not by user code directly.
     * Overridden here to adjust the size of the frame if needed.
     * @see java.awt.Container#removeNotify
     */
    public void addNotify()
    {
      // Record the size of the window prior to calling parents
addNotify.
      Dimension size = getSize();

      super.addNotify();

      if (frameSizeAdjusted)
            return;
      frameSizeAdjusted = true;

      // Adjust size of frame according to the insets and menu bar
      javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
      int menuBarHeight = 0;
      if (menuBar != null)
            menuBarHeight = menuBar.getPreferredSize().height;
      Insets insets = getInsets();
      setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
    }

    // Used by addNotify
```

```
    boolean frameSizeAdjusted = false;

    //{{DECLARE_CONTROLS
    javax.swing.JButton EightButton = new javax.swing.JButton();
    javax.swing.JButton NineButton = new javax.swing.JButton();
    javax.swing.JButton TenButton = new javax.swing.JButton();
    javax.swing.JButton ElevenButton = new javax.swing.JButton();
    javax.swing.JButton TwelveButton = new javax.swing.JButton();
    javax.swing.JButton ThirteenButton = new javax.swing.JButton();
    javax.swing.JButton FourteenButton = new javax.swing.JButton();
    javax.swing.JButton FifteenButton = new javax.swing.JButton();
    javax.swing.JButton SixteenButton = new javax.swing.JButton();
    javax.swing.JButton SeventeenButton = new javax.swing.JButton();
    javax.swing.JButton EighteenButton = new javax.swing.JButton();
    javax.swing.JButton NineteenButton = new javax.swing.JButton();
    javax.swing.JButton TwentyButtton = new javax.swing.JButton();
    javax.swing.JButton TwentyOneButton = new javax.swing.JButton();
    javax.swing.JTextPane DateDisplay = new javax.swing.JTextPane();
    javax.swing.JButton ForwardButton = new javax.swing.JButton();
    javax.swing.JButton BackButton = new javax.swing.JButton();
    //}}

    //{{DECLARE_MENUS
    //}}

    void exitApplication()
    {
      try {
          // Beep
          Toolkit.getDefaultToolkit().beep();
          // Show a confirmation dialog
          int reply = JOptionPane.showConfirmDialog(this,
                                          "Do you really want to
exit?",
                                          "JFC Application - Exit" ,
                                          JOptionPane.YES_NO_OPTION,

JOptionPane.QUESTION_MESSAGE);
          // If the confirmation was affirmative, handle exiting.
          if (reply == JOptionPane.YES_OPTION)
            {
              this.setVisible(false);     // hide the Frame
              this.dispose();             // free the system resources
              System.exit(0);             // close the application
            }
      } catch (Exception e) {
      }
    }

    class SymWindow extends java.awt.event.WindowAdapter
    {
      public void windowClosing(java.awt.event.WindowEvent event)
      {
          Object object = event.getSource();
          if (object == DayGUI.this)
            DayGUI_windowClosing(event);
      }
```

```
  }
class MeetingGUI extends javax.swing.JFrame
{
  private GregorianCalendar thisDay;
  private Meeting display;
  private boolean smart = false;
  private Vector gList = new Vector(Group.NAMES);
  public Font f = new Font("dialog", Font.PLAIN,10);
  private String meetingLabelTime = "default";

    public MeetingGUI()
    {
          getContentPane().setLayout(null);
          setSize(240,320);
          setVisible(false);
          MeetingDate.setText("January 1, 2001");
          getContentPane().add(MeetingDate);
          MeetingDate.setFont(new Font("Dialog", Font.BOLD, 12));
          MeetingDate.setBounds(80,10,175,24);
          NameL.setText("Name:");
          getContentPane().add(NameL);
          NameL.setBounds(5,40,70,15);
          NameL.setFont(f);

          NameD.setText("Name");
          NameD.setBounds(80,40,140,15);
          NameD.setFont(f);

          mName.setBounds(80,40,140,15);
          mName.setFont(f);

          TimeL.setText("Time");
          getContentPane().add(TimeL);
          TimeL.setFont(f);
          TimeL.setBounds(5,63,60,15);
          Time.setText("Time Goes Here");
          getContentPane().add(Time);
          Time.setBounds(80,63,140,15);
          Time.setFont(f);

          LocationL.setText("Location:");
          getContentPane().add(LocationL);
          LocationL.setFont(f);
          LocationL.setBounds(5,86,70,15);

          LocationD.setText("BLAH");
          LocationD.setBounds(80,86,140,15);
          LocationD.setFont(f);
          loc_list.setBounds(80,116,140,18);
          loc_list.setFont(f);
              groups.setBounds(80,86,140,18);
          groups.setFont(f);

          SummaryL.setText("Summary:");
          getContentPane().add(SummaryL);
          SummaryL.setBounds(5,172,70,15);
          SummaryL.setFont(f);
```

```
SummaryD.setText("Summary");
SummaryD.setBounds(80,172,150,45);
SummaryD.setFont(f);
summary.setBounds(80,172,150,45);
summary.setLineWrap(true);
summary.setFont(f);

SmartL.setText("Smart:");
getContentPane().add(SmartL);
SmartL.setBounds(5,225,70,15);
SmartL.setFont(f);
SmartD.setText("YES");
SmartD.setFont(f);
SmartD.setBounds(80,225,48,15);
smartN.setSelected(true);
smartN.setText("No");
smartN.setBounds(140,225,48,15);
smartN.setFont(f);
smartY.setText("Yes");
smartY.setBounds(80,225,48,15);
smartY.setFont(f);

NotificationL.setText("Notification:");
getContentPane().add(NotificationL);
NotificationL.setBounds(5,149,70,15);
NotificationL.setFont(f);
NotificationD.setText("Notification");
NotificationD.setFont(f);
NotificationD.setBounds(80,149,96,15);

not.setBounds(80,149,75,15);
not.setFont(f);

// Buttons
ClearB.setText("Clear");
ClearB.setBounds(5,275,70,15);
ClearB.setFont(f);
CancelB.setText("Cancel");
CancelB.setBounds(85,275,70,15);
CancelB.setFont(f);
CreateB.setText("Create");
CreateB.setBounds(165,275,70,15);
CreateB.setFont(f);


OkB.setText("Ok");
OkB.setBounds(5,275,70,15);
OkB.setFont(f);
EditB.setText("Edit");
EditB.setBounds(85,275,70,15);
EditB.setFont(f);
DeleteB.setText("Delete");
DeleteB.setBounds(165,275,70,15);
DeleteB.setFont(f);
DoneB.setText("Done");
DoneB.setBounds(165,275,70,15);
DoneB.setFont(f);
```

```
            //{{REGISTER_LISTENERS
            SymMouse aSymMouse = new SymMouse();
                SymAction aSymAction = new SymAction();
            smartY.addMouseListener(aSymMouse);
            smartN.addMouseListener(aSymMouse);
            ClearB.addMouseListener(aSymMouse);
            CreateB.addMouseListener(aSymMouse);
            CancelB.addMouseListener(aSymMouse);
            OkB.addMouseListener(aSymMouse);
            EditB.addMouseListener(aSymMouse);
            DeleteB.addMouseListener(aSymMouse);
            DoneB.addMouseListener(aSymMouse);
            groups.addActionListener(aSymAction);

                //}}
      }

//This is the the MeetingGUI constructor that we will use: Called by
the DayGUI
    public MeetingGUI(GregorianCalendar today)
      {
            this();
            if (!((String)gList.elementAt(0)).equals("All")) {
                  gList.add(0,"All");
            }
            groups.setSelectedIndex(0);
                getContentPane().add(mName);
            getContentPane().add(loc_list);
                getContentPane().add(groups);
            getContentPane().add(summary);
            getContentPane().add(smartN);
            getContentPane().add(smartY);
            getContentPane().add(not);
            getContentPane().add(CancelB);
            getContentPane().add(ClearB);
            getContentPane().add(CreateB);

            setTitle("Smart Reminder");

            thisDay = today;
            Time.setText(today.get(today.HOUR_OF_DAY) + ":00");

            String month = null;
            int mo = today.get(today.MONTH);
            month = getMonth(mo);
            MeetingDate.setText(month + " " + today.get(today.DATE) +
", " + today.get(today.YEAR) );
    }

    public GregorianCalendar copyDate( GregorianCalendar date ) {
      GregorianCalendar copy = new GregorianCalendar(
date.get(date.YEAR), date.get(date.MONTH), date.get(date.DATE),
date.get(date.HOUR_OF_DAY), date.get(date.MINUTE) );
      return( copy );
    }
```

```
    //Called by the DayGUI to display an existing meeting
    public MeetingGUI(Meeting meet)
      {
            this();
            display=meet;
            getContentPane().add(NameD);
            getContentPane().add(LocationD);
            getContentPane().add(SummaryD);
            getContentPane().add(SmartD);
            getContentPane().add(NotificationD);
            getContentPane().add(OkB);
            getContentPane().add(EditB);
            getContentPane().add(DeleteB);
            NameD.setText(meet.getName());
            LocationD.setText(meet.getLocation());
            SummaryD.setText(meet.getSummary());
            setTitle("Smart Reminder");
            GregorianCalendar today = copyDate( meet.getMeetingDate()
);
            Time.setText(today.get(today.HOUR_OF_DAY) + ":00");
            if (meet.isSmart()) {
                SmartD.setText("YES");
            }
            else SmartD.setText("NO");
            if (meet.getNotification()==0)
                NotificationD.setText("10 minutes");
            else if (meet.getNotification()==1)
                NotificationD.setText("20 minutes");
            else if (meet.getNotification()==2)
                NotificationD.setText("30 minutes");
            else if (meet.getNotification()==3)
                NotificationD.setText("40 minutes");
            else if (meet.getNotification()==4)
                NotificationD.setText("50 minutes");
            else if (meet.getNotification()==5)
                NotificationD.setText("1 hour");
            else if (meet.getNotification()==6)
                NotificationD.setText("1 day");


            thisDay = meet.getMeetingDate();
            Time.setText(thisDay.get(thisDay.HOUR_OF_DAY) + ":00");

            String month = null;
            int mo = thisDay.get(thisDay.MONTH);
            month = getMonth(mo);
            MeetingDate.setText(month + " " + thisDay.get(thisDay.DATE)
+ ", " + thisDay.get(thisDay.YEAR) );
      }

    public MeetingGUI(Meeting meet, int empty)
    {
      this();
      if (!((String)gList.elementAt(0)).equals("All")) {
          gList.add(0,"All");
        }
```

```
        groups.setSelectedIndex(0);
        getContentPane().add(mName);
      getContentPane().add(loc_list);
      getContentPane().add(groups);
      getContentPane().add(summary);
      getContentPane().add(smartN);
      getContentPane().add(smartY);
      getContentPane().add(not);
      getContentPane().add(CancelB);
      getContentPane().add(ClearB);
      getContentPane().add(DoneB);

      setTitle("Smart Reminder");
      mName.setText(meet.getName());
        loc_list.setSelectedItem(meet.getLocation());
      //groups.setSelectedIndex("All");
        summary.setText(meet.getSummary());

      if (meet.isSmart()) {
          smartN.setSelected(false);
          smartY.setSelected(true);
          smart = true;
      }

      not.setSelectedIndex(meet.getNotification());

        thisDay = meet.getMeetingDate();
        Time.setText(thisDay.get(thisDay.HOUR_OF_DAY) + ":00");

        String month = null;
        int mo = thisDay.get(thisDay.MONTH);
        month = getMonth(mo);
        MeetingDate.setText(month + " " + thisDay.get(thisDay.DATE) +
", " + thisDay.get(thisDay.YEAR) );
    }


    public String getMonth(int mo)
    {
      if ( mo == 0)
          return("January");
      else if ( mo == 1 )
          return("February");
      else if ( mo == 2 )
          return("March");
      else if ( mo == 3)
          return("April");
      else if ( mo == 4 )
          return("May");
      else if ( mo == 5 )
          return("June");
      else if ( mo == 6 )
          return("July");
      else if ( mo == 7 )
          return("August");
      else if ( mo == 8 )
          return("September");
```

```
      else if ( mo == 9 )
          return("October");
      else if ( mo == 10 )
          return ("November");
      else if ( mo == 11 )
          return( "December" );
      else return( "error" );
    }

    public void setVisible(boolean b)
    {
      if (b)
          setLocation(50, 50);
      super.setVisible(b);
    }

    public void addNotify()
    {
      // Record the size of the window prior to calling parents
addNotify.
      Dimension size = getSize();

      super.addNotify();

      if (frameSizeAdjusted)
          return;
      frameSizeAdjusted = true;

      // Adjust size of frame according to the insets and menu bar
      Insets insets = getInsets();
      javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
      int menuBarHeight = 0;
      if (menuBar != null)
          menuBarHeight = menuBar.getPreferredSize().height;
      setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
    }

    // Used by addNotify
    boolean frameSizeAdjusted = false;



    //{{DECLARE_CONTROLS
    String[] notList = {"10 min","20 min","30 min","40 min","50 min","1
hour","1 day"};
    javax.swing.JLabel MeetingDate = new javax.swing.JLabel();
    javax.swing.JLabel NameL = new javax.swing.JLabel();
    javax.swing.JLabel NameD = new javax.swing.JLabel();
    javax.swing.JLabel LocationL = new javax.swing.JLabel();
    javax.swing.JLabel LocationD = new javax.swing.JLabel();
    javax.swing.JLabel SummaryL = new javax.swing.JLabel();
    javax.swing.JLabel SummaryD = new javax.swing.JLabel();
    javax.swing.JLabel SmartL = new javax.swing.JLabel();
    javax.swing.JTextField mName = new javax.swing.JTextField();
    javax.swing.JComboBox groups = new javax.swing.JComboBox(gList);
```

```
    javax.swing.JComboBox loc_list = new
javax.swing.JComboBox(Location.LIST_LOC);
    javax.swing.JTextArea summary = new javax.swing.JTextArea();
    javax.swing.JLabel NotificationL = new javax.swing.JLabel();
    javax.swing.JLabel NotificationD = new javax.swing.JLabel();
    javax.swing.JLabel TimeL = new javax.swing.JLabel();
    //javax.swing.JTextArea Time = new javax.swing.JTextArea();
    javax.swing.JLabel Time = new javax.swing.JLabel();
    javax.swing.JLabel SmartD = new javax.swing.JLabel();
    javax.swing.JButton CancelB = new javax.swing.JButton();
    javax.swing.JButton CreateB = new javax.swing.JButton();
    javax.swing.JButton ClearB = new javax.swing.JButton();
    javax.swing.JButton OkB = new javax.swing.JButton();
    javax.swing.JButton DeleteB = new javax.swing.JButton();
    javax.swing.JButton EditB = new javax.swing.JButton();
    javax.swing.JButton DoneB = new javax.swing.JButton();
    javax.swing.JComboBox not = new javax.swing.JComboBox(notList);
    javax.swing.JRadioButton smartY = new javax.swing.JRadioButton();
    javax.swing.JRadioButton smartN = new javax.swing.JRadioButton();



    //{{DECLARE_MENUS



    class SymMouse extends java.awt.event.MouseAdapter
    {
      public void mouseClicked(java.awt.event.MouseEvent event)
      {
          Object object = event.getSource();
          if (object == smartY)
            smartY_mouseClicked(event);
          else if (object == smartN)
            smartN_mouseClicked(event);
          else if (object == ClearB)
            ClearB_mouseClicked(event);
          else if (object == CreateB)
            CreateB_mouseClicked(event);
          else if (object == CancelB)
            CancelB_mouseClicked(event);
          else if (object == EditB)
            EditB_mouseClicked(event);
          else if (object == OkB)
            OkB_mouseClicked(event);
          else if (object == DeleteB)
            DeleteB_mouseClicked(event);
          else if (object == DoneB)
            DoneB_mouseClicked(event);
      }
    }


    class SymAction implements ActionListener
    {
      public void actionPerformed(ActionEvent e) {
          JComboBox cb = (JComboBox)e.getSource();
```

```
          int i = groups.getSelectedIndex();
        loc_list.setVisible(false);
          if (i == 0) {
            loc_list = new javax.swing.JComboBox(Location.LIST_LOC);
          loc_list.setBounds(80,116,140,18);
          loc_list.setFont(f);
          getContentPane().add(loc_list);
            System.out.println("All was selected");
        }
          else {
            i--;
            System.out.println("The index is: "+i);
          loc_list = new
javax.swing.JComboBox((Vector)Group.GROUPS.get(i));
          loc_list.setBounds(80,116,140,18);
          loc_list.setFont(f);
          getContentPane().add(loc_list);
          }
        loc_list.setVisible(true);
      }
    }

        void smartY_mouseClicked(java.awt.event.MouseEvent event) {
        smart = true;
        smartN.setSelected(false);
        smartY.setSelected(true);
      }

      void smartN_mouseClicked(java.awt.event.MouseEvent event)
      {
        smart = false;
        smartY.setSelected(false);
        smartN.setSelected(true);
      }

      void ClearB_mouseClicked(java.awt.event.MouseEvent event)
      {
        smart = false;
        not.setSelectedIndex(0);
        mName.setText("");
        summary.setText("");
        smartY.setSelected(false);
          smartN.setSelected(true);
      }

      void CreateB_mouseClicked(java.awt.event.MouseEvent event)
      {
          if ( mName.getText().equals("")) {
            System.out.println("Meeting name is Null");
            (new ErrorGUI("nameNull")).setVisible(true);
          }
          else if ( summary.getText().equals("")) {
            System.out.println("A meeting description is required");
            (new ErrorGUI("summaryNull")).setVisible(true);
          }
          else if ( Time.getText().equals("")) {
            System.out.println("A time is required");
```

```
            (new ErrorGUI("timeNull")).setVisible(true);
          }
          else {
            setVisible(false);
            meetingLabelTime = Time.getText();
            updateMeetingLabel(mName.getText(), meetingLabelTime);
            // in case the user has changed the time field
            //GregorianCalendar copy = new GregorianCalendar(
thisDay.get(thisDay.YEAR), thisDay.get(thisDay.MONTH),
thisDay.get(thisDay.DATE), meetingLabelTime,
thisDay.get(thisDay.MINUTE) );
            Meeting create = new Meeting(thisDay, mName.getText(),
(String)loc_list.getSelectedItem(), summary.getText(),
not.getSelectedIndex(), smart);
            MeetingList.insert(create);
          }

      }
      public void updateMeetingLabel(String name, String time)
      {
          if(time.equals("8:00")) {EightText.setText(name);}
          else if(time.equals("9:00")) {NineText.setText(name);}
          else if(time.equals("10:00")) {TenText.setText(name);}
          else if(time.equals("11:00")) {ElevenText.setText(name);}
          else if(time.equals("12:00")) {TwelveText.setText(name);}
          else if(time.equals("13:00")) {ThirteenText.setText(name);}
          else if(time.equals("14:00")) {FourteenText.setText(name);}
          else if(time.equals("15:00")) {FifteenText.setText(name);}
          else if(time.equals("16:00")) {SixteenText.setText(name);}
          else if(time.equals("17:00")) {SeventeenText.setText(name);}
          else if(time.equals("18:00")) {EighteenText.setText(name);}
          else if(time.equals("19:00")) {NineteenText.setText(name);}
          else if(time.equals("20:00")) {TwentyText.setText(name);}
          else if(time.equals("21:00")) {TwentyOneText.setText(name);}
      }

      void CancelB_mouseClicked(java.awt.event.MouseEvent event)
      {
          setVisible(false);
      }

      void OkB_mouseClicked(java.awt.event.MouseEvent event)
      {
          setVisible(false);
      }

      void EditB_mouseClicked(java.awt.event.MouseEvent event)
      {
          copyForDelete = display;//add code here to copy display
meeting
          setVisible(false);
          (new MeetingGUI(display, 0)).setVisible(true);
      }

      void DeleteB_mouseClicked(java.awt.event.MouseEvent event)
      {
          MeetingList.delete(display);
```

```
            setVisible(false);
            meetingLabelTime = Time.getText();
            updateMeetingLabel(mName.getText(), meetingLabelTime);
            //We should refresh the DayGUI
        }

        void DoneB_mouseClicked(java.awt.event.MouseEvent event)
        {
            if ( mName.getText().equals("")) {
              System.out.println("Meeting name is Null");
              (new ErrorGUI("nameNull")).setVisible(true);
            }
            else if ( summary.getText().equals("")) {
              System.out.println("A meeting description is required");
              (new ErrorGUI("summaryNull")).setVisible(true);
            }
            else {
                MeetingList.delete(copyForDelete); //should delete
instead of display a copy of the original
                setVisible(false);
                //Create new meeting
                Meeting create = new Meeting(thisDay, mName.getText(),
(String)loc_list.getSelectedItem(), summary.getText(),
not.getSelectedIndex(), smart);
                MeetingList.insert(create);
                System.out.println("Meeting has been created");
                meetingLabelTime = Time.getText();
                updateMeetingLabel(mName.getText(), meetingLabelTime);
              }
        }
    }
    void DayGUI_windowClosing(java.awt.event.WindowEvent event)
    {
      boolean savedAlarm = AlarmList.save();
      boolean savedMeeting = MeetingList.save();
        boolean savedLocationAlarmList = LocationAlarmList.save();
      System.out.println("Alarm state saved: "+savedAlarm+ " Meetings
state saved: "+savedMeeting+ " LocationAlarms state saved:
"+savedLocationAlarmList);
      this.setVisible(false);
    }

    class SymAction implements java.awt.event.ActionListener
    {
      public void actionPerformed(java.awt.event.ActionEvent event)
      {
          Object object = event.getSource();
          System.out.println("action performed");
      }
    }

    class SymMouse extends java.awt.event.MouseAdapter
    {
      public void mouseClicked(java.awt.event.MouseEvent event)
      {   //findmeforbuttons
          Object object = event.getSource();
          if (object == EighteenButton){
```

```
            EighteenButton_mouseClicked(event);
          }
          else if (object == ForwardButton) {
            ForwardButton_mouseClicked(event);
          }
          else if (object == BackButton) {
            BackButton_mouseClicked(event);
          }
          else if (object == EightButton){
            EightButton_mouseClicked(event);
          }
          else if (object == NineButton) {
            NineButton_mouseClicked(event);
          }
          else if (object == TenButton) {
            TenButton_mouseClicked(event);
          }
          else if (object == ElevenButton) {
            ElevenButton_mouseClicked(event);
          }
          else if (object == TwelveButton) {
            TwelveButton_mouseClicked(event);
          }
          else if (object == ThirteenButton) {
            ThirteenButton_mouseClicked(event);
          }
          else if (object == FourteenButton) {
            FourteenButton_mouseClicked(event);
          }
          else if (object == FifteenButton) {
            FifteenButton_mouseClicked(event);
          }
          else if (object == SixteenButton) {
            SixteenButton_mouseClicked(event);
          }
          else if (object == SeventeenButton) {
            SeventeenButton_mouseClicked(event);
         }
          else if (object == NineteenButton) {
            NineteenButton_mouseClicked(event);
          }
          else if (object == TwentyButtton) {
            TwentyButtton_mouseClicked(event);
          }
          else if (object == TwentyOneButton) {
            TwentyOneButton_mouseClicked(event);
          }
      }
    }
    public void paint() {
      DateDisplay.setDisabledTextColor(java.awt.Color.black);
      DateDisplay.setEditable(false);
      getContentPane().add(DateDisplay);
      DateDisplay.setText(day);
      DateDisplay.setBackground(new java.awt.Color(204,204,204));
      DateDisplay.setBounds(60,12,240,40);
      //$$ bevelBorder1.move(0,413);
```

```
EightText.setText(eight);
EightText.setDisabledTextColor(java.awt.Color.black);
EightText.setEditable(false);
getContentPane().add(EightText);
EightText.setBackground(new java.awt.Color(204,204,204));
EightText.setBounds(72,72,279,18);
//$$ compoundBorder1.move(24,413);
//$$ etchedBorder1.move(48,413);
//$$ lineBorder1.move(72,413);
//$$ softBevelBorder1.move(96,413);
//$$ titledBorder1.move(120,413);
NineText.setText(nine);
NineText.setDisabledTextColor(java.awt.Color.black);
NineText.setEditable(false);
getContentPane().add(NineText);
NineText.setBackground(new java.awt.Color(204,204,204));
NineText.setBounds(72,96,279,18);
TenText.setDisabledTextColor(java.awt.Color.black);
TenText.setEditable(false);
getContentPane().add(TenText);
TenText.setText(ten);
TenText.setBackground(new java.awt.Color(204,204,204));
TenText.setBounds(72,120,279,18);
ElevenText.setDisabledTextColor(java.awt.Color.black);
ElevenText.setEditable(false);
getContentPane().add(ElevenText);
ElevenText.setText(eleven);
ElevenText.setBackground(new java.awt.Color(204,204,204));
ElevenText.setBounds(72,144,279,18);
TwelveText.setDisabledTextColor(java.awt.Color.black);
TwelveText.setEditable(false);
getContentPane().add(TwelveText);
TwelveText.setText(twelve);
TwelveText.setBackground(new java.awt.Color(204,204,204));
TwelveText.setBounds(72,168,279,18);
ThirteenText.setDisabledTextColor(java.awt.Color.black);
ThirteenText.setEditable(false);
getContentPane().add(ThirteenText);
ThirteenText.setText(thirteen);
ThirteenText.setBackground(new java.awt.Color(204,204,204));
ThirteenText.setBounds(72,192,279,18);
FourteenText.setDisabledTextColor(java.awt.Color.black);
FourteenText.setEditable(false);
getContentPane().add(FourteenText);
FourteenText.setText(fourteen);
FourteenText.setBackground(new java.awt.Color(204,204,204));
FourteenText.setBounds(72,216,279,18);
FifteenText.setDisabledTextColor(java.awt.Color.black);
FifteenText.setEditable(false);
getContentPane().add(FifteenText);
FifteenText.setText(fifteen);
FifteenText.setBackground(new java.awt.Color(204,204,204));
FifteenText.setBounds(72,240,279,18);
SixteenText.setDisabledTextColor(java.awt.Color.black);
SixteenText.setEditable(false);
getContentPane().add(SixteenText);
SixteenText.setText(sixteen);
```

```
      SixteenText.setBackground(new java.awt.Color(204,204,204));
      SixteenText.setBounds(72,264,279,18);
      SeventeenText.setDisabledTextColor(java.awt.Color.black);
      SeventeenText.setEditable(false);
      getContentPane().add(SeventeenText);
      SeventeenText.setText(seventeen);
      SeventeenText.setBackground(new java.awt.Color(204,204,204));
      SeventeenText.setBounds(72,288,279,18);
      EighteenText.setDisabledTextColor(java.awt.Color.black);
      EighteenText.setEditable(false);
      getContentPane().add(EighteenText);
      EighteenText.setText(eighteen);
      EighteenText.setBackground(new java.awt.Color(204,204,204));
      EighteenText.setBounds(72,312,279,18);
      NineteenText.setDisabledTextColor(java.awt.Color.black);
      NineteenText.setEditable(false);
      getContentPane().add(NineteenText);
      NineteenText.setText("asdfasdfasdf");
      NineteenText.setBackground(new java.awt.Color(204,204,204));
      NineteenText.setBounds(72,336,279,18);
      TwentyText.setDisabledTextColor(java.awt.Color.black);
      TwentyText.setEditable(false);
      getContentPane().add(TwentyText);
      TwentyText.setText(twenty);
      TwentyText.setBackground(new java.awt.Color(204,204,204));
      TwentyText.setBounds(72,360,279,18);
      TwentyOneText.setDisabledTextColor(java.awt.Color.black);
      TwentyOneText.setEditable(false);
      getContentPane().add(TwentyOneText);
      TwentyOneText.setText(twentyone);
      TwentyOneText.setBackground(new java.awt.Color(204,204,204));
      TwentyOneText.setBounds(72,384,279,18);
      getContentPane().add(ForwardButton);
      ForwardButton.setBounds(324,12,12,40);
      getContentPane().add(BackButton);
      BackButton.setBounds(24,12,12,40);
    }
    /**
     * Returns a java GregorianCalendar object with an invalid meeting
time
     */
    public boolean isMeetingValid(Meeting m) {
      GregorianCalendar maybe = m.getMeetingDate();
      if(maybe.get(maybe.YEAR) == 1970 ) {
          return false;
      }
      else {
          return true;
      }
    }

    void EighteenButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 18;
```

```
      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          EighteenText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);
          if(isMeetingValid(m) ) {
            EighteenText.setText(m.getName());

          }
          else {
            EighteenText.setText("");

          }
      }
    }


    public void d(String m) {
      if(debug == true) {
          System.out.println(m);
      }
    }


    void ForwardButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      today.add(today.DATE, 1);
      DateDisplay.setText(parseCalendar(today));
      updateNames(today);
    }

    void BackButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      today.add(today.DATE, -1);
      DateDisplay.setText(parseCalendar(today));
      updateNames(today);


    }

    void EightButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 8;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
```

```
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          EightText.setText(m.getName());
      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);
           }
    }

    void NineButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 9;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          NineText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);

      }
    }

    void TenButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 10;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          TenText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
```

```
            m = MeetingList.search(now);

        }
    }

    void ElevenButton_mouseClicked(java.awt.event.MouseEvent event)
    {
        int yr = today.get(today.YEAR);
        int mo = today.get(today.MONTH);
        int dt = today.get(today.DATE);
        int hr = 11;

        GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
        Meeting m = MeetingList.search(now);
        if(isMeetingValid(m) ) {
            d("result of isMeetingValid " + isMeetingValid(m));
            (new MeetingGUI(m)).setVisible(true);
            ElevenText.setText(m.getName());

        }
        else {
            (new MeetingGUI(now)).setVisible(true);
            d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
            m = MeetingList.search(now);

        }
    }

    void TwelveButton_mouseClicked(java.awt.event.MouseEvent event)
    {
        int yr = today.get(today.YEAR);
        int mo = today.get(today.MONTH);
        int dt = today.get(today.DATE);
        int hr = 12;

        GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
        Meeting m = MeetingList.search(now);
        if(isMeetingValid(m) ) {
            d("result of isMeetingValid " + isMeetingValid(m));
            (new MeetingGUI(m)).setVisible(true);
            TwelveText.setText(m.getName());

        }
        else {
            (new MeetingGUI(now)).setVisible(true);
            d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
            m = MeetingList.search(now);

        }
    }

    void ThirteenButton_mouseClicked(java.awt.event.MouseEvent event)
    {
        int yr = today.get(today.YEAR);
        int mo = today.get(today.MONTH);
```

```
      int dt = today.get(today.DATE);
      int hr = 13;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          ThirteenText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);

      }
    }

    void FourteenButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 14;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          FourteenText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);

      }
    }

    void FifteenButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 15;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          FifteenText.setText(m.getName());
```

```
      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);

      }
    }

    void SixteenButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 16;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          SixteenText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);

      }
    }

    void SeventeenButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 17;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          SeventeenText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);

      }
```

```
    }

    void NineteenButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 19;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          NineteenText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);
          }
    }

    void TwentyButtton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 20;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          TwentyText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);

      }
    }

    void TwentyOneButton_mouseClicked(java.awt.event.MouseEvent event)
    {
      int yr = today.get(today.YEAR);
      int mo = today.get(today.MONTH);
      int dt = today.get(today.DATE);
      int hr = 21;

      GregorianCalendar now = new GregorianCalendar(yr,mo,dt,hr,0);
```

```
      Meeting m = MeetingList.search(now);
      if(isMeetingValid(m) ) {
          d("result of isMeetingValid " + isMeetingValid(m));
          (new MeetingGUI(m)).setVisible(true);
          TwentyOneText.setText(m.getName());

      }
      else {
          (new MeetingGUI(now)).setVisible(true);
          d("this is the hour of the gregorian calendar we are givein
go the meeting ui " + now.get(now.HOUR_OF_DAY));
          m = MeetingList.search(now);

      }
    }
    }
```

IX.8 DistanceToTravel.java

```
/**
 * this is a matrix implemented as a Vector with all the Locations from
Location.LOC_LIST
 * this vector is used as an index into a long[][] which is where all
the estimates of
 * TimeToTravel are held in milliseconds
 *
 * @author Craig Music
 */


package SmartReminder;
import java.util.*;
import SmartReminder.Location.*;
import java.io.Serializable;



public class DistanceToTravel implements Serializable {
  private static Vector locs = new Vector();
  private static long[][] distances;
  private static long MAX_TIME = 10*60*1000;

  public DistanceToTravel() {
  }

  public static void main(String[] args) {
    System.out.println(MAX_TIME);
    Random randy = new Random();


  }

  /**
   * This method initializes the matrix and populates all slots with
random
```

```
   * values.
   */
  public static void init() {
      // get the vector from the Location class
      locs = SmartReminder.Location.Location.LIST_LOC;
      int length = locs.size();
      distances = new long[length][length];
      Random randy = new Random();
      for(int y = 0; y < length; y++) {
            for(int x = 0; x < length; x++) {
             double r = randy.nextDouble()*MAX_TIME;
             long t = (long)r;
            distances[y][x] = t;
            }
      }

  }

  /*
  * Method that given String representations for departure and arrival
  * locations will return an estimate of the time to travel from one to
the
  * other in milliseconds/
  */
  public static long getTimeToTravel(String departure, String arrival)
{
    // this is what the rest of the program will use
    int departIndex = locs.indexOf(departure);
    int arriveIndex = locs.indexOf(departure);
    long result = distances[departIndex][arriveIndex];
    return result;
  }

   public static long getTimeToTravel(int departure, int arrival) {
    // this is used for debug purposes
    long result = distances[departure][arrival];
    return result;
  }
}
```

IX.9 ErrorGUI.java

```
/**
 * ErrorGUI displays an error message
 *
 * @author Marc Bourget
 */

package SmartReminder;

import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.lang.*;

public class ErrorGUI extends javax.swing.JFrame {
```

```
  public ErrorGUI() {
    getContentPane().setLayout(null);
    setSize(200,100);
    setVisible(false);
    getContentPane().add(ErrorL);
    ErrorL.setFont(new Font("Dialog", Font.BOLD, 16));
    ErrorL.setBounds(70,12,80,20);
    MessageL.setText("Error Message");
    getContentPane().add(MessageL);
    MessageL.setBounds(15,35,170,20);
    OkB.setBounds(70,70,70,25);
    getContentPane().add(OkB);

    //{{REGISTER_LISTENERS
    SymMouse aSymMouse = new SymMouse();
    OkB.addMouseListener(aSymMouse);
    //}}
  }

  //Called by the DayGUI to display what field is blank
  public ErrorGUI(String message) {
    this();
    if (message.equals("nameNull"))
      MessageL.setText("Meeting name is empty.");
    else if (message.equals("summaryNull"))
      MessageL.setText("Summary is empty.");
    else if (message.equals("locationNull"))
      MessageL.setText("Location is empty.");
    else if (message.equals("timeNull"))
      MessageL.setText("Time is empty.");
  }

  public void setVisible(boolean b) {
    if (b)
      setLocation(50, 50);
    super.setVisible(b);
  }

  static public void main(String args[]) {
    (new ErrorGUI("This is a test")).setVisible(true);
    GregorianCalendar today = CurrentDate.getCurrentCalendar();
    System.out.println(today);
  }

  public void addNotify() {
    // Record the size of the window prior to calling parents
addNotify.
    Dimension size = getSize();
    super.addNotify();
    if (frameSizeAdjusted)
      return;
    frameSizeAdjusted = true;
    // Adjust size of frame according to the insets and menu bar
    Insets insets = getInsets();
    javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
    int menuBarHeight = 0;
    if (menuBar != null)
```

```
      menuBarHeight = menuBar.getPreferredSize().height;
    setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
  }

  // Used by addNotify
  boolean frameSizeAdjusted = false;

  //{{DECLARE_CONTROLS
  javax.swing.JLabel ErrorL = new javax.swing.JLabel("ERROR!");
  javax.swing.JLabel MessageL = new javax.swing.JLabel();
  javax.swing.JButton OkB = new javax.swing.JButton("OK");
  //{{DECLARE_MENUS

  class SymMouse extends java.awt.event.MouseAdapter {
    public void mouseClicked(java.awt.event.MouseEvent event) {
      Object object = event.getSource();
      if (object == OkB)
      setVisible(false);
    }
  }
}
```

IX.10 Group.java

```
/**
 * Group Class maintains information about the system groups
 *
 * author Marc Bourget
 */

package SmartReminder.Location;

import java.util.*;
import java.io.*;

public class Group {
    public static Vector NAMES = new Vector(); //Holds the name of all
the groups
    public static Vector GROUPS = new Vector(); // Holds the locations
for the groups

    public Group() {
    }

    /**
     * This method initializes the vectors according to the groups file
     */
    public static void init() {
      Vector temp;
      DataInputStream input;
      StringBuffer b = new StringBuffer();
      String data = "";
      int charint = 0;
      NAMES = new Vector();
      GROUPS = new Vector();
```

```
      try {
      input = new DataInputStream(new FileInputStream( "Groups" ) );
        while((charint=input.read())!=-1) {
        b.append((char)charint);
      }
      }
      catch (IOException e) {
      System.err.println( "Error opening or reading file\n" +
e.toString() );
      System.exit(1);
      }
      data = b.toString();
      StringTokenizer st = new StringTokenizer(data, "\n"); // separate
the data into individual lines
      StringTokenizer lt;
      String line,name,loc;

      while(st.hasMoreTokens()) {
      temp = new Vector();
        line = st.nextToken();
        lt = new StringTokenizer(line, ":");
        name = lt.nextToken();
        NAMES.add(name);

        while(lt.hasMoreTokens()) {
        loc = lt.nextToken();
        temp.add(loc);
        }
      GROUPS.add(temp);
      }
    }

    public static void remove(String loc) {
    /**
     * now we remove loc from all of the groups
     */
      for (int i=0; i<Group.NAMES.size(); i++) {
        if (((Vector)Group.GROUPS.get(i)).contains(loc)) {
          ((Vector)Group.GROUPS.get(i)).remove(loc);
        }
      }
      saveVectors();
    }

    /**
     * adds a new group
     */
    static public void addGroup(String g) {
      String data = "";
      try {
        FileWriter fw = new FileWriter( "Groups", true );
      data = "\n"+g;
      fw.write(data);
      fw.close();
      }
      catch (IOException e) {
```

```
      System.err.println( "Error opening or writing to file\n" +
e.toString() );
      System.exit(1);
      }

      /**
       * add the new group to the proper vectors
       */
      Group.NAMES.add(g);
      Vector v = new Vector();
      Group.GROUPS.add(v);
    }

    static public void removeGroup(int i) {
      Group.NAMES.remove(i);
      Group.GROUPS.remove(i);
      saveVectors();
    }

    /**
     * saves the state of the vectors to a file
     */
    static public void saveVectors() {
      String data = "";
      for (int i=0; i<Group.NAMES.size(); i++) {
        data = data+((String)Group.NAMES.elementAt(i));
        for (int k=0; k<((Vector)Group.GROUPS.get(i)).size(); k++) {
          data = data+":"+((Vector)Group.GROUPS.get(i)).elementAt(k);
        }
        data = data+"\n";
      }

      try {
        FileWriter fw = new FileWriter( "Groups",false );
        fw.write(data);
        fw.close();
      }
      catch (IOException e) {
        System.err.println( "Error opening or writing to file\n" +
e.toString() );
        System.exit(1);
      }
    }
}
```

IX.11 IPLocProvider.java

```
/**
 * IPLocProvider Class maintains information about the device's
location
 * and is responsible for updating the system of the device's
whereabouts
 *
```

```
 * author Marc Bourget
 */

package SmartReminder.Location;

import java.util.*;
import java.io.*;

public class IPLocProvider {
    static private String ip = "10.10.10.10";
    static private String mask = "255.255.255.255";

    public IPLocProvider(String name, int priority) {
    }

    static public void main(String[] args) {
    }

    /**
     * initializes the vectors according to the IPTable
     */
    static public void init() {
      DataInputStream input;
      StringBuffer b = new StringBuffer();
      String data = "";
      int charint = 0;
      Location.LIST_LOC = new Vector();
      Location.LOC_ALL = new Vector();
      Location.IP_ALL = new Vector();

      try {
      input = new DataInputStream(new FileInputStream( "IPtable" ) );
        while((charint=input.read())!=-1) {
        b.append((char)charint);
        }
      }
      catch (IOException e) {
      System.err.println( "Error opening or reading file\n" +
e.toString() );
      System.exit(1);
      }
      data = b.toString();
      StringTokenizer st = new StringTokenizer(data, "\n\r"); //
separate the data into individual lines
      StringTokenizer lt;
      String line,ip,loc;

      while(st.hasMoreTokens()) {
      line = st.nextToken();
        lt = new StringTokenizer(line, " \t");
        ip = lt.nextToken();
        loc = lt.nextToken();

        while(lt.hasMoreTokens()) {
        loc = loc + " " + lt.nextToken();
        }
```

```
        Location.LOC_ALL.add(loc);
        Location.IP_ALL.add(ip);


        /**
       * Don't allow duplicates in the location list
       */
        if (!Location.LIST_LOC.contains(loc)) {
        Location.LIST_LOC.add(loc);
        }
      }

      System.out.println("Test 18.2.0.0/21 maps to " +
getLoc("18.2.0.0/21"));
      System.out.println("Test 128.52.0.0/16 maps to " +
getLoc("128.52.0.0/16"));
    }

    /**
     * Finds the location for the ip(s) if it is known
     */
    static public String getLoc(String s) {
      if (Location.IP_ALL.contains(s)) {

return((String)Location.LOC_ALL.get(Location.IP_ALL.indexOf(s)));
      }
      else {
        return("unknown");
      }
    }

    /**
     * Makes a system call to obtain the ip address and mask
     */
    static public String[] getIP() {
      ip = "Not Available";
      mask = "Not Available";
      Runtime rt = Runtime.getRuntime();
      String input = "";
      try {
      Process p = rt.exec("pump --status");
      InputStream is = p.getInputStream();
      StringBuffer b = new StringBuffer();
      int charint = 0;
      while((charint=is.read())!=-1) {
        b.append((char)charint);
      }
      input = b.toString();
      }
      catch(IOException e) {
      System.out.println("Can't obtain ip info");
      }

      StringTokenizer st = new StringTokenizer(input);
      while(st.hasMoreTokens()) {

      String token = st.nextToken();
```

```
      if(token.equals("IP:")) {
        ip = st.nextToken();
      }
      if(token.equals("Netmask:")) {
        mask = st.nextToken();
      }
      }
      String[] ipInfo = {ip, mask};
      return( ipInfo );
    }

    /**
     * Returns the location name for the current ip address (if it is
known)
     */
    static public String getCurrentLoc() {
      String[] ipInfo = getIP();
      String name = "unknown";
      if (!ipInfo[0].equals("Not Available")) {
      ipInfo = getLocName();
      System.out.println("The location (according to iplocProvider is
"+ipInfo[1]);
      name = ipInfo[1];
      }
      return(name);
    }


    static public String[] getLocName() {
      /**
       * Find out the number of significant bits
       */
      StringTokenizer st = new StringTokenizer(mask, ".");
      String byte1 = st.nextToken();
      String byte2 = st.nextToken();
      String byte3 = st.nextToken();
      int end;
      if (byte1.equals("255")) {
      if (byte2.equals("255")) {
        if (byte3.equals("255")) {
          end = 24;
        }
        else  end = 16;
      }
      else end = 0;
      }
      else end = 0;

      /**
       * Calculate the key
       */
      st = new StringTokenizer(ip, ".");
      byte1 = st.nextToken();
      byte2 = st.nextToken();
      byte3 = st.nextToken();
      String key = "";
      String temp[] = { "unknown", "unknown"};
```

```
      while (end > 8) {
      if (end > 16) {
        key = byte1 + "." + byte2 + "." + byte3 + "." + "0/" + end;
      }
      else key = byte1 + "." + byte2 + ".0.0/" + end;
        String locName = (String)getLoc(key);
      if ( !locName.equals("unknown") ) {
        temp[0] = key;
        temp[1] = locName;
        break;
      }
      end--;
      }
      return(temp);
    }

    /**
     * removes a location from the vectors and file (IPTable)
     */
    static public void removeLocation(String key) {
      /**
       * remove the pair from the vectors
       */
      if (Location.IP_ALL.contains(key)) {
        int m = Location.IP_ALL.indexOf(key);
        Location.IP_ALL.removeElementAt(m);
        Location.LOC_ALL.removeElementAt(m);
        /**
       * remove from the file by saving the new vectors
       */
        String data = "";
        for (int i=0; i<Location.IP_ALL.size(); i++) {
          data = data + "\n" + Location.IP_ALL.elementAt(i) + "\t" +
Location.LOC_ALL.elementAt(i);
        }
        try {

          FileWriter fw = new FileWriter( "IPtable", false );
          fw.write(data);
          fw.close();
          //after saving the new file, we should reinitialize
          init();
        }
        catch (IOException e) {
        System.err.println( "Error opening or writing to file\n" +
e.toString() );
        System.exit(1);
        }
      }
      else {
      System.err.println("Key: "+key+" not found");
      }
    }

    static public void editLocation(String key, String loc) {
      /**
```

```
       * if neither the key nor loc exist in the two vectors, treat
them as a new location entry
       */
      if ( !(Location.IP_ALL.contains(key)) &&
!(Location.LOC_ALL.contains(loc)) ) {
        addLocation(key,loc);
      }
      else {
        int m;
        if (Location.IP_ALL.contains(key)) {
          m = Location.IP_ALL.indexOf(key);
          Location.LOC_ALL.set(m, loc);
          Location.IP_ALL.set(m, key);
        }
        else if (Location.LOC_ALL.contains(loc)) {
          m = Location.LOC_ALL.indexOf(loc);
          Location.LOC_ALL.set(m, loc);
          Location.IP_ALL.set(m, key);
        }
        else {
          System.exit(1);
        }

        String data = "";
        for (int i=0; i<Location.IP_ALL.size(); i++) {
          data = data + "\n" + Location.IP_ALL.elementAt(i) + "\t" +
Location.LOC_ALL.elementAt(i);
        }
        try {
          FileWriter fw = new FileWriter( "IPtable", false );
          fw.write(data);
          fw.close();
          /**
           * after saving the new file, we should reinitialize
           */
          init();
        }
        catch (IOException e) {
          System.err.println( "Error opening or writing to file\n" +
e.toString() );
          System.exit(1);
        }
      }
    }

    static public void addLocation(String key, String location) {
      String data = "";
      try {
        FileWriter fw = new FileWriter( "IPtable", true );
      data = "\n"+key+"\t"+location;
      fw.write(data);
      fw.close();
      }
      catch (IOException e) {
      System.err.println( "Error opening or writing to file\n" +
e.toString() );
      System.exit(1);
```

```
        }
        /**
         * add location pair to the vectors
         */
        Location.LOC_ALL.add(location);
        Location.IP_ALL.add(key);

        if (!Location.LIST_LOC.contains(location)) {
        Location.LIST_LOC.add(location);
        }
    }
}
```

IX.12 Location.java

```
/**
 * Location Class holds information about the current location and the
 * current vectors from the IPTable and CricketTable
 *
 * author Marc Bourget
 */

package SmartReminder.Location;

import java.util.*;

public class Location {
    public static Vector LIST_LOC = new Vector(); // List of location
names (without duplicate entries)

    /**
     * IP Vectors
     */
    public static Vector IP_ALL = new Vector();
    public static Vector LOC_ALL = new Vector();

    /**
     * Cricket Vectors
     */
    public static Vector CRICKET_BEACON_IDS = new Vector();
    public static Vector CRICKET_LOCS = new Vector();

    public Location() {
    }
}
```

IX.13 LocationAlarm.java

```
/**
 * Class that represent a LocationAlarm object. A meeting or activity
that
 * only knows the location where it is supposed to happen and will
trigger if
 * its internal locatio matches that location that the system is
currently at.
```

```
 *
 * @author: Craig Music
 */

package SmartReminder;
import SmartReminder.Location.*;
import java.io.Serializable;
import java.util.Date;
import java.util.*;

public class LocationAlarm implements Serializable{
  private String alarmName;
  private String alarmLocation;
  private String message;
  private long creationTime;
  private boolean locationIsGroup;

  /**
   * Constructor
   */
  public LocationAlarm(String name, String location, String mess, long
time, boolean locationIsGroup) {
       alarmName = name;
       alarmLocation = location;
       message = mess;
       creationTime = time;
       this.locationIsGroup = locationIsGroup;
  }

  public static void main(String[] args) {
    LocationAlarm la = new LocationAlarm("TestLocationAlarm", "Office",
"message this", 465464, false);
    la.alert();
  }

   /**
   * Used to notify this object that the system's location has changed.
   * If the object's internal location matches the currentLocation an
alarm
   * will be triggered. Used when only one location provider is
functioning
   */
  public void locationChanged(String currentLocation) {
   boolean contains = false;
   if(locationIsGroup) {
    int index =
SmartReminder.Location.Group.NAMES.indexOf(this.alarmLocation);
    Vector groupLocs =
(Vector)SmartReminder.Location.Group.GROUPS.elementAt(index);
    contains = groupLocs.contains(currentLocation);
    System.out.println("Current Location: "+ currentLocation+ " match
with alarmLocation: "+this.alarmLocation);
   }
   if(!locationIsGroup) {
    if(currentLocation.equals(this.alarmLocation)) {
      contains = true;
    }
```

```
    }
    if(contains) {
     alert();
    }
   }

   /**
    * Used to notify this object that the system's location has changed.
    * If the object's internal location matches the currentLocation an
alarm
    * will be triggered. Used when two location providers are
functioning,
    * hence the two current location, one cricket and one IP
    */
   public void locationChanged(String currentLocation1, String
currentLocation2) {
    boolean contains1 = false;
    boolean contains2 = false;
    if(locationIsGroup) {
     int index =
SmartReminder.Location.Group.NAMES.indexOf(this.alarmLocation);
     Vector groupLocs =
(Vector)SmartReminder.Location.Group.GROUPS.elementAt(index);
     contains1 = groupLocs.contains(currentLocation1);
     contains2 = groupLocs.contains(currentLocation2);
     System.out.println("Current Locations: "+ currentLocation1+ " or "
+ currentLocation2+" match with alarmLocation: "+this.alarmLocation);
    }
    if(!locationIsGroup) {
     if(currentLocation1.equals(this.alarmLocation) ||
currentLocation2.equals(this.alarmLocation)) {
       contains1 = true;
       contains2 = true;
     }
    }
    if(contains1 || contains2) {
     alert();
    }
   }

    /**
     *  this will trigger the alarm and display a GUI
     */
   public void alert() {

     (new LocationAlertGUI(this.alarmName, this.message,
this.alarmLocation)).setVisible(true);
     System.out.println("This is a LocationAlarm alarm. ALERT ALERT");
   }

   /**
    * Used to querry the system whether its location is a group of
locations or
    * a single one.
    */
   public boolean isLocationGroup() {
    return this.locationIsGroup;
```

```
  }

  /**
   * Used to notify the object if its internal location is actually a
   * group of locations instead of a single one.
   */
  public void setLocationIsGroup(boolean t) {
    locationIsGroup = t;
  }

  /**
   * Returns the String name of this LocationAlarm.
   */
  public String getName() {
       return alarmName;
  }

   /**
    * Method returns the internal location where this object should
happen
    */
  public String getLocation() {
       return alarmLocation;
  }

  /**
   * Method returns the description inputed by the user about this
LocationAlarm
   */
  public String getMessage() {
       return message;
  }

  /**
   * Method returns the creation time of this LocationAlarm
   */
  public String getTime() {
       Date dtime = new Date(creationTime);
       return dtime.toString();
  }

  /**
   * Method used to set the creation time of this object
   */
  public void setTime(long time) {
       creationTime = time;
  }

  /**
   * Method used to set the description of this object
   */
  public void setMessage(String mess) {
       message = mess;
  }

  /**
   * Method used to set the name of this object
```

```
   */
  public void setName(String name) {
       alarmName = name;
  }

  /**
   * Method used to set the Location of this object
   */
  public void setLocation(String location) {
       alarmLocation = location;
  }
}
```

IX.14 LocationAlarmGUI.java

```
/**
 * LocationAlarmGUI allows the user to add a location to the list
 *
 * @author Craig Music
 */

package SmartReminder;
import SmartReminder.Location.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.*;

public class LocationAlarmGUI extends javax.swing.JFrame
{
    private Font f = new Font("dialog",Font.PLAIN,10);
    private LocationAlarmViewerGUI lavg;
    private boolean editor = false;
    private boolean locationIsGroup;


    public LocationAlarmGUI(SmartReminder.LocationAlarmViewerGUI
parent, LocationAlarm la)
    {
      this(parent);
      name.setText(la.getName());
      message.setText(la.getMessage());
      editor = true;
      if(la.isLocationGroup())
      {
      loc1.setSelected(true);
      loc2.setSelected(false);
      int index = Group.NAMES.indexOf((la.getLocation()));
      group_list.setSelectedIndex(index);
      }
      if(!la.isLocationGroup()) {
      loc2.setSelected(true);
      loc1.setSelected(false);
      int index = Location.LIST_LOC.indexOf((la.getLocation()));
      loc_list.setSelectedIndex(index);
```

```
    }
    }

    public LocationAlarmGUI(SmartReminder.LocationAlarmViewerGUI
parent)
    {
      this.lavg = parent;
      setTitle("LocationAlarm");
      getContentPane().setLayout(null);
      setSize(240,300);
      setVisible(true);

      header.setText("Input Alarm based solely on Location");
      getContentPane().add(header);
      header.setBounds(15,10,240,15);
      header.setFont(f);

      name_L.setText("Name:");
      getContentPane().add(name_L);
      name_L.setBounds(15,45,50,15);
      name_L.setFont(f);

      name.setBounds(85,45,130,15);
      getContentPane().add(name);
      name.setFont(f);
      name.setEditable(true);
      name.setVisible(true);

      loc_List.setText("Location:");
      getContentPane().add(loc_List);
      loc_List.setBounds(15,85,50,15);
      loc_List.setFont(f);
      loc_List.setVisible(true);

      loc_list.setBounds(85,85,130,15);
      getContentPane().add(loc_list);
      loc_list.setFont(f);

      group_List.setText("Group:");
      getContentPane().add(group_List);
      group_List.setBounds(15,65,130,15);
      group_List.setFont(f);
      group_List.setVisible(true);

      group_list.setBounds(85,65,130,15);
      getContentPane().add(group_list);
      group_list.setFont(f);

      loc0.setBounds(70,45,12,12);
      getContentPane().add(loc0);
      loc0.setVisible(false);

      loc1.setSelected(false);
      loc1.setBounds(70,65,12,12);
      getContentPane().add(loc1);

      loc2.setSelected(true);
```

```
      loc2.setBounds(70, 85,12,12);
      getContentPane().add(loc2);

      messageLabel.setText("Enter Message:");
      messageLabel.setBounds(15,115,100,10);
      getContentPane().add(messageLabel);
      messageLabel.setFont(f);

      message.setBounds(15,130,200,45);
      getContentPane().add(message);
      message.setFont(f);
      message.setEditable(true);
      message.setForeground(Color.white);
      message.setBackground(Color.black);

      Add_Btn.setText("Add");
      Add_Btn.setBounds(15,190,75,15);
      Add_Btn.setFont(f);
      getContentPane().add(Add_Btn);

      Close_Btn.setText("Close");
      getContentPane().add(Close_Btn);
      Close_Btn.setBounds(125,190,90,15);
      Close_Btn.setFont(f);

      //{{REGISTER_LISTENERS
      SymMouse aSymMouse = new SymMouse();
      SymAction aSymAction = new SymAction();
      Add_Btn.addMouseListener(aSymMouse);
      Close_Btn.addMouseListener(aSymMouse);
      loc0.addMouseListener(aSymMouse);
      loc1.addMouseListener(aSymMouse);
      loc2.addMouseListener(aSymMouse);
      group_list.addActionListener(aSymAction);
      loc_list.addActionListener(aSymAction);
                  //}}

    }


    public void updateDisplay()
    {
    System.out.println("updateDisplay called");
      // get wheter loc1 or loc 2 is slected
      if(loc1.isSelected())
      {
      loc_List.setText("Group locs");
      int i = group_list.getSelectedIndex();
      System.out.println("The selected index of group_list is: "+i);

      if(i >= 0)
      {
            getContentPane().remove(loc_list);
            loc_list = new
javax.swing.JComboBox((Vector)Group.GROUPS.get(i));
            loc_list.setBounds(85,85,130,15);
            getContentPane().add(loc_list);
```

```
        loc_list.setFont(f);
        loc_list.setVisible(true);
    }
   }
   if(loc2.isSelected())
   {
        loc_List.setText("Location:");
        getContentPane().remove(loc_list);
        loc_list = new javax.swing.JComboBox(Location.LIST_LOC);
        loc_list.setBounds(85,85,130,15);
        getContentPane().add(loc_list);
        loc_list.setFont(f);
        loc_list.setVisible(true);
   }
  }

  public void setVisible(boolean b)
  {
    if (b)
        setLocation(50, 50);
    super.setVisible(b);
  }


  static public void main(String args[])
  {
    //(new LocationAlarmGUI()).setVisible(true);
  }

  public String getKey(String ip, String m) {

    //Figure out the number of significant bits
    StringTokenizer st = new StringTokenizer(m, ".");
    String byte1 = st.nextToken();
    String byte2 = st.nextToken();
    String byte3 = st.nextToken();
    int end;
    if (byte1.equals("255")) {
        if (byte2.equals("255")) {
          if (byte3.equals("255")) {
              end = 24;
          }
          else  end = 16;
        }
        else end = 0;
    }
    else end =0;

    //Figure out the key
    st = new StringTokenizer(ip, ".");
    String segment = st.nextToken() + "." + st.nextToken();
    String key;
    if (end == 16) {
        key = segment + ".0.0/16";
    }
    else if (end == 24) {
        String next = st.nextToken();
```

```
            key = segment + "." + next + ".0/24";
        }
        else
            key = "error";

        return(key);
    }

    public void addNotify()
    {
        // Record the size of the window prior to calling parents
addNotify.
        Dimension size = getSize();

        super.addNotify();

        if (frameSizeAdjusted)
            return;
        frameSizeAdjusted = true;

        // Adjust size of frame according to the insets and menu bar
        Insets insets = getInsets();
        javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
        int menuBarHeight = 0;
        if (menuBar != null)
            menuBarHeight = menuBar.getPreferredSize().height;
        setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
    }

    // Used by addNotify
    boolean frameSizeAdjusted = false;
    //{{DECLARE_CONTROLS
    javax.swing.JComboBox loc_list = new
javax.swing.JComboBox(Location.LIST_LOC);
    javax.swing.JComboBox group_list = new
javax.swing.JComboBox(Group.NAMES);
    javax.swing.JLabel loc_List = new javax.swing.JLabel();
    javax.swing.JLabel group_List = new javax.swing.JLabel();
    javax.swing.JLabel name_L = new javax.swing.JLabel();
    javax.swing.JLabel header = new javax.swing.JLabel();
    javax.swing.JTextArea message = new javax.swing.JTextArea();
    javax.swing.JLabel messageLabel = new javax.swing.JLabel();
     javax.swing.JTextArea name = new javax.swing.JTextArea();
    javax.swing.JRadioButton loc0 = new javax.swing.JRadioButton();
    javax.swing.JRadioButton loc1 = new javax.swing.JRadioButton();
    javax.swing.JRadioButton loc2 = new javax.swing.JRadioButton();
    javax.swing.JLabel MessageL = new javax.swing.JLabel();
    javax.swing.JButton Add_Btn = new javax.swing.JButton();
    javax.swing.JButton Close_Btn = new javax.swing.JButton();



    class SymMouse extends java.awt.event.MouseAdapter
    {
        public void mouseClicked(java.awt.event.MouseEvent event)
        {
```

```
        Object object = event.getSource();

        if (object == Add_Btn)
          Add_Btn_mouseClicked(event);
        if (object == Close_Btn)
          Close_Btn_mouseClicked(event);
        if (object == loc1) {
          loc0.setSelected(false);
          loc1.setSelected(true);
          loc2.setSelected(false);
          updateDisplay();

        }
         if (object == loc2) {
          loc1.setSelected(false);
          loc2.setSelected(true);
          updateDisplay();

        }

      }
    }

    class SymAction implements ActionListener
    {
      public void actionPerformed(ActionEvent e)
      {
      JComboBox cb = (JComboBox)e.getSource();
      updateDisplay();
      //System.out.println("called update display from action
performed");
      }

    }

    void Add_Btn_mouseClicked(java.awt.event.MouseEvent event)
    {
      String location = "default";
      if(loc1.isSelected()) {
      //think about making a class group and just having a vector of
groups here...
      location = (String)group_list.getSelectedItem();
      this.locationIsGroup = true;
      }
      if(loc2.isSelected()) {
       location = (String)loc_list.getSelectedItem();
       this.locationIsGroup = false;
      }
      LocationAlarm locAlarm = new LocationAlarm(name.getText(),
location, message.getText(),
CurrentDate.getCurrentTimeMillis(),locationIsGroup);
      LocationAlarmList.addLocationAlarm(locAlarm);
      lavg.updateDisplay();
      this.setVisible(false);
     }
```

```
    void Close_Btn_mouseClicked(java.awt.event.MouseEvent event) {
       this.setVisible(false);
    }



}
```

IX.15 LocationAlarmList.java

```
/**
 * Class is a repository data structure that gives add, delete, modify
access
 * to LocationAlarm elements that reside inside.
 *
 * @author: Craig Music
 */

package SmartReminder;
import java.util.*;
import java.io.*;


 /* Two vectors keep the state of this list. One vector is a vector of
Location alarms whereass
 the second vecond holds only the locations for each of the location
alarms. This is so such that
 the hasElement method of Vector can be used to quickly verify the
existance of an alarm
 */
public class LocationAlarmList {
       private static Vector locationAlarms = new Vector();
       private static Vector locationAlarmLocations = new Vector();
       private static Vector locationAlarmNames = new Vector();

 /**
  * Deletes all elements in the data structure.
  */
  public static void renew() {
       locationAlarms = new Vector();
       locationAlarmLocations = new Vector();
       locationAlarmNames = new Vector();
  }

  /**
   * Method will notify every element inside the data structure that
the
   * system's current location has changed. Notifies of a single
current
   * location because only one location provider is working.
   */
  public static void locationChanged(String currentLocation) {
    int size = locationAlarms.size();
    int index = 0;
    for(index = 0; index < size; index++) {
```

```
      LocationAlarm la =
(LocationAlarm)locationAlarms.elementAt(index);
      la.locationChanged(currentLocation);
      locationAlarms.set(index, la);
     }
    }

  /**
   * Method will notify every element inside the data structure that
the
   * system's current location has changed. Notifies of two current
   * locations because two location providers are working.
   */
    public static void locationChanged(String[] currentLocation, int
active) {
    int size = locationAlarms.size();
    if(active == 1) {
      int index = 0;
      for(index = 0; index < size; index++) {
       LocationAlarm la =
(LocationAlarm)locationAlarms.elementAt(index);
       la.locationChanged(currentLocation[0]);
       locationAlarms.set(index, la);
      }
    }
    if(active ==2) {
      int index = 0;
      for(index = 0; index < size; index++) {
       LocationAlarm la =
(LocationAlarm)locationAlarms.elementAt(index);
       la.locationChanged(currentLocation[0], currentLocation[1]);
       locationAlarms.set(index, la);
      }
    }
  }

/**
 * empty constructor
 */
  public LocationAlarmList() {
  }

  /**
   * used to add elements into the repository
   */
  public static boolean addLocationAlarm(LocationAlarm la) {
       String location = la.getLocation();
       String name = la.getName();
       locationAlarms.addElement(la);
       locationAlarmLocations.addElement(location);
       locationAlarmNames.addElement(name);
       String test =
((LocationAlarm)locationAlarms.lastElement()).getLocation();
       if(test.equals(location)) {
            return true;
       }
       else {
```

```
            return false;
        }
    }

    /**
     * used to remove elements from the repository
     */
    public static void removeLocationAlarm(String name) {
      int index = locationAlarmNames.indexOf(name);
      for(int y = 0;y < locationAlarmNames.size(); y++)
      {
        System.out.println(locationAlarmNames.elementAt(y));
      }


      System.out.println("removing "+name);
      System.out.println("index" + index);
      if(index!= -1) {
        locationAlarms.removeElementAt(index);
        locationAlarmLocations.removeElementAt(index);
        locationAlarmNames.removeElementAt(index);
      }
    }


    /**
     * Method will return all LocationAlarm objects whose location
matches
     * the argument
     */
    public static Vector exists(String location) {
        // must check several times in case ther are multiple alarms
with the same location
        Vector alarms = new Vector();
        int index = 0;
        int lastFound = 0;
        while (index != -1) {
            index = locationAlarmLocations.indexOf(location,
lastFound);
            lastFound = index;
            if(index != -1) {
                alarms.addElement(locationAlarms.elementAt(index));
            }
        }
        return alarms;
    }

    /**
     * Method used to read a serialized version of the object from file
and
     * create a new object
     */
    static public boolean read()
      {
        locationAlarms = new Vector();
        locationAlarmLocations = new Vector();
        boolean result = false;
```

```
        // unserialize the vector
        System.out.println("unserializing LocationAlarmList");
        try {
       FileInputStream fin = new
FileInputStream("LocationAlarmList.dat");
       ObjectInputStream ois = new ObjectInputStream(fin);
       locationAlarms = (Vector) ois.readObject();
       ois.close();
       result = true;
       int size = locationAlarms.size();
       int index = 0;
       while(index < size) {
                  String location =
((LocationAlarm)locationAlarms.elementAt(index)).getLocation();
                  String name =
((LocationAlarm)locationAlarms.elementAt(index)).getName();
                  locationAlarmLocations.insertElementAt(location,
index);
                  locationAlarmNames.insertElementAt(name,index);
                  index++;
       }
        }
        catch (Exception e)
       {
         e.printStackTrace();
         result = false;
         return result;
       }
        return result;
        }

        /**
         * Method utilizes java Serializale to write a copy of this
method
         * to disk
         */
        static public boolean save()
    {
      boolean result = false;
      System.out.println("serializing locationAlarms Vector");
      try {
      FileOutputStream fout = new
FileOutputStream("LocationAlarmList.dat");
      ObjectOutputStream oos = new ObjectOutputStream(fout);
      oos.writeObject(locationAlarms);
      oos.close();
      result = true;
      }
      catch (Exception e)
      {
         result = false;
         e.printStackTrace();
         return result;
      }
      return result;
    }
```

```
      static public Vector getLocationAlarms() {
           return locationAlarms;
      }
}
```

IX.16 LocationAlarmViewerGUI.java

```
/**
 * LocationAlarmGUI allows the user to view all the LocationAlarm
 * objects inside the system as well as allowing the access to the
 * repository data structure LocationAlarmList
 *
 * @author Craig Music
 */

package SmartReminder;
import SmartReminder.Location.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.*;

public class LocationAlarmViewerGUI extends javax.swing.JFrame
{
    private Font f = new Font("dialog",Font.PLAIN,10);
    private Vector headers = createHeaders();
    private Vector locationAlarms =
LocationAlarmList.getLocationAlarms();
    private Vector data = createData();

    private javax.swing.JTable table;
    private javax.swing.JScrollPane jt;

    /**
     * Spawns a new GUI
     */
    public LocationAlarmViewerGUI() {

      setTitle("LocationAlarmViewer");
      getContentPane().setLayout(null);
      setSize(240,300);
      setVisible(true);

       locationAlarms = LocationAlarmList.getLocationAlarms();
       data = createData();
       table  = new javax.swing.JTable(data, headers);
       table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

       table.setFont(f);
       jt = new javax.swing.JScrollPane(table);
       getContentPane().add(jt);
       jt.setBounds(10,30,225,150);
       jt.setFont(f);

      header.setText("Existing LocationAlarms");
```

```
      getContentPane().add(header);
      header.setBounds(15,10,240,15);
      header.setFont(f);

      //Other
      Add_Btn.setText("Add");
      Add_Btn.setBounds(15,190,75,15);
      Add_Btn.setFont(f);
      getContentPane().add(Add_Btn);
      Add_Btn.setVisible(true);

      Remove_Btn.setText("Remove");
      Remove_Btn.setBounds(125,190,100,15);
      Remove_Btn.setFont(f);
      getContentPane().add(Remove_Btn);
      Remove_Btn.setVisible(true);

      Close_Btn.setText("Close");
      getContentPane().add(Close_Btn);
      Close_Btn.setBounds(15,215,75,15);
      Close_Btn.setFont(f);
      Close_Btn.setVisible(true);

      Edit_Btn.setText("Edit");
      getContentPane().add(Edit_Btn);
      Edit_Btn.setBounds(125,215,100,15);
      Edit_Btn.setFont(f);
      Edit_Btn.setVisible(true);

      SymMouse aSymMouse = new SymMouse();
      SymAction aSymAction = new SymAction();
      Add_Btn.addMouseListener(aSymMouse);
      Remove_Btn.addMouseListener(aSymMouse);
      Edit_Btn.addMouseListener(aSymMouse);
      Close_Btn.addMouseListener(aSymMouse);
      Update_Btn.addMouseListener(aSymMouse);
      //}}

    }

    /**
     * Obtains data to be displayed from the LocationAlarmList
repository
     * data structure
     */
    public Vector createData() {
         // name location creationtime
         data = new Vector();
         locationAlarms = LocationAlarmList.getLocationAlarms();
         int size = locationAlarms.size();
         if(size == 0) {
          return data;
          } // in case no locatoin alarms
         for(int i = 0; i < size; i++)
         {
            Vector temp = new Vector(); // subvector of name, location,
creationTime
```

```
            LocationAlarm la =
(LocationAlarm)locationAlarms.elementAt(i);
            temp.add(la.getName());
            temp.add(la.getLocation());
            temp.add(la.getTime());
            data.add(temp);
        }
        return data;
    }
    /**
     * Formats that data obtained from LocationAlarmList to be
displayed
     */
    public Vector createHeaders() {
        Vector temp = new Vector();
        temp.add("Names");
        temp.add("Locations");
        temp.add("Created");
        return temp;
    }

     public void setVisible(boolean b)
    {
      if (b)
          setLocation(50, 50);
      super.setVisible(b);
    }


    static public void main(String args[])
    {
      (new LocationAlarmViewerGUI()).setVisible(true);
    }


    public void addNotify()
    {
      // Record the size of the window prior to calling parents
addNotify.
      Dimension size = getSize();

      super.addNotify();

      if (frameSizeAdjusted)
          return;
      frameSizeAdjusted = true;

      // Adjust size of frame according to the insets and menu bar
      Insets insets = getInsets();
      javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
      int menuBarHeight = 0;
      if (menuBar != null)
          menuBarHeight = menuBar.getPreferredSize().height;
      setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
    }
```

```
// Used by addNotify
boolean frameSizeAdjusted = false;
//{{DECLARE_CONTROLS


javax.swing.JLabel header = new javax.swing.JLabel();
javax.swing.JTextArea message = new javax.swing.JTextArea();

javax.swing.JButton Add_Btn = new javax.swing.JButton();
javax.swing.JButton Edit_Btn = new javax.swing.JButton();
javax.swing.JButton Remove_Btn = new javax.swing.JButton();
javax.swing.JButton Close_Btn = new javax.swing.JButton();
javax.swing.JButton Update_Btn = new javax.swing.JButton();


class SymMouse extends java.awt.event.MouseAdapter
{
  public void mouseClicked(java.awt.event.MouseEvent event)
  {

      Object object = event.getSource();

      if (object == Add_Btn)
        Add_Btn_mouseClicked(event);
      if (object == Remove_Btn)
        Remove_Btn_mouseClicked(event);
      if (object == Edit_Btn)
        Edit_Btn_mouseClicked(event);
      if (object == Close_Btn)
        Close_Btn_mouseClicked(event);
      if (object == Update_Btn)
        Update_Btn_mouseClicked(event);


  }
}

class SymAction implements ActionListener
{
  public void actionPerformed(ActionEvent e)
  {

  }

 }

void Add_Btn_mouseClicked(java.awt.event.MouseEvent event)
{
   (new LocationAlarmGUI(this)).setVisible(true);
}

 void Remove_Btn_mouseClicked(java.awt.event.MouseEvent event)
{
  //todo
  int row = table.getSelectedRow();
  int column =0;
```

```
      String temp = (String)table.getValueAt(row, column);
      LocationAlarmList.removeLocationAlarm(temp);
      updateDisplay();
    }

    public void updateDisplay() {
       Vector temp = new Vector();
       temp = createData();
       table  = new javax.swing.JTable(temp, headers);
       table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

       getContentPane().remove(jt);
       table.setFont(f);
       jt = new javax.swing.JScrollPane(table);
       getContentPane().add(jt,0);
       jt.setBounds(10,30,225,150);
       jt.setFont(f);

    }


    void Edit_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      int row = table.getSelectedRow();
      int column =0;
      Vector locationAlarms = LocationAlarmList.getLocationAlarms();
      LocationAlarm la = (LocationAlarm)locationAlarms.elementAt(row);
      //locationalarmviewregui, isEditor, name, message, time
      (new LocationAlarmGUI(this, la)).setVisible(true);
    }

    void Close_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      LocationAlarmList.save();
      this.setVisible(false);
    }

     class SymWindow extends java.awt.event.WindowAdapter
    {
      public void windowClosing(java.awt.event.WindowEvent event)
      {
          Object object = event.getSource();
          if (object == LocationAlarmViewerGUI.this)
            LocationAlarmViewerGUI_windowClosing(event);
      }
    }

    void
LocationAlarmViewerGUI_windowClosing(java.awt.event.WindowEvent event)
    {
      LocationAlarmList.save();
      System.out.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&LocationAlarmList
state saved");
      this.setVisible(false);
    }

    void Update_Btn_mouseClicked(java.awt.event.MouseEvent event) {

    }
```

```
}
```

IX.17 LocationAlertGUI.java

```
/**
 * Creates a GUI showing graphically the location, name and summary of
a
 * LocationAlarm object.
 */
package SmartReminder;

import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.lang.*;

public class LocationAlertGUI extends javax.swing.JFrame
{

        /**
         * construcor
         */
      public LocationAlertGUI()
      {
            getContentPane().setLayout(null);
            setSize(240,320);
            setVisible(false);
            AlertL.setText("Meeting Name");
            getContentPane().add(AlertL);
            AlertL.setFont(new Font("Dialog", Font.BOLD, 16));
            AlertL.setBounds(15,12,120,30);
            MessageL.setEditable(false);
            getContentPane().add(MessageL);
            MessageL.setBounds(15,45,210,80);
            OkB.setText("OK");
            OkB.setActionCommand("OK");
            OkB.setBounds(85,150,70,25);
            getContentPane().add(OkB);
            setTitle("REMINDER");
            SymMouse aSymMouse = new SymMouse();
            OkB.addMouseListener(aSymMouse);
      }


              /**
               * Constructor that allows the definition of name,
message and location
               */
           public LocationAlertGUI(String name, String message, String
location)
          {
            this();
            AlertL.setText(name);
            MessageL.setText("This appointment is in " + location + "
Please remember to " + message);
```

```
        }


    public void setVisible(boolean b)
    {
        if (b)
            setLocation(50, 50);
        super.setVisible(b);
    }

    static public void main(String args[])
    {
         GregorianCalendar today = new GregorianCalendar(2001, 4, 20,
7, 30);
        (new LocationAlertGUI("test name", "remember to blah", "kens
office")).setVisible(true);
    }

    public void addNotify()
    {
        // Record the size of the window prior to calling parents
addNotify.
        Dimension size = getSize();

        super.addNotify();

        if (frameSizeAdjusted)
            return;
        frameSizeAdjusted = true;

        // Adjust size of frame according to the insets and menu
bar
        Insets insets = getInsets();
        javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
        int menuBarHeight = 0;
        if (menuBar != null)
            menuBarHeight = menuBar.getPreferredSize().height;
        setSize(insets.left + insets.right + size.width, insets.top
+ insets.bottom + size.height + menuBarHeight);
    }

    // Used by addNotify
    boolean frameSizeAdjusted = false;

    javax.swing.JLabel AlertL = new javax.swing.JLabel();
    javax.swing.JTextPane MessageL = new javax.swing.JTextPane();
    javax.swing.JButton OkB = new javax.swing.JButton();

    class SymMouse extends java.awt.event.MouseAdapter
    {
        public void mouseClicked(java.awt.event.MouseEvent event)
        {
            Object object = event.getSource();
            if (object == OkB)
                OkB_mouseClicked(event);

        }
```

```
    }

    void OkB_mouseClicked(java.awt.event.MouseEvent event)
    {
        setVisible(false);
    }

}
```

IX.18 LocationGUI.java

```
/**
 * LocationGUI allows the user to view the data currently being
 * collected from the various location providers and
 * add a location to the list if it doesn't exist yet
 *
 * author Marc Bourget
 */

package SmartReminder.Location;

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.*;

public class LocationGUI extends javax.swing.JFrame {
  private Font f = new Font("dialog",Font.PLAIN,10);
  private boolean name = true;
  public CricketLocProvider clp;

  public LocationGUI() {
    setTitle("LocationGUI");
    getContentPane().setLayout(null);
    setSize(240,320);
    setVisible(true);

    header.setText("Here is your location info");
    getContentPane().add(header);
    header.setBounds(15,10,145,15);
    header.setFont(f);

    locP.setBounds(160,10,70,15);
    getContentPane().add(locP);
    locP.setFont(f);

    loc_L.setText("Location");
    getContentPane().add(loc_L);
    loc_L.setBounds(15,115,50,20);
    loc_L.setFont(f);

    loc.setBounds(85,115,130,15);
    getContentPane().add(loc);
    loc.setFont(f);
```

```
loc.setEditable(false);

message.setBounds(15,185,200,45);
getContentPane().add(message);
message.setFont(f);
message.setEditable(false);
message.setForeground(Color.white);
message.setBackground(Color.black);

/**
 * IP Fields
 */
getContentPane().add(ip_L);
ip_L.setBounds(15,35,50,15);
ip_L.setFont(f);
ip.setBounds(85,35,130,15);
getContentPane().add(ip);
ip.setFont(f);
getContentPane().add(mask_L);
mask_L.setBounds(15,60,50,15);
mask_L.setFont(f);
mask.setBounds(85,60,130,15);
getContentPane().add(mask);
mask.setFont(f);
getContentPane().add(key_L);
key_L.setBounds(15,85,50,15);
key_L.setFont(f);
key.setBounds(85,85,130,15);
getContentPane().add(key);
key.setFont(f);

/**
 * Cricket Fields
 */
getContentPane().add(beaconID_L);
beaconID_L.setBounds(15,35,50,15);
beaconID_L.setFont(f);
beaconID_L.setVisible(false);
beaconID.setBounds(85,35,130,15);
getContentPane().add(beaconID);
beaconID.setFont(f);
beaconID.setVisible(false);

Add_Btn.setText("Add");
Add_Btn.setBounds(15,235,75,15);
Add_Btn.setFont(f);
getContentPane().add(Add_Btn);
Add_Btn.setVisible(false);

loc0.setSelected(true);
loc0.setBounds(70,115,12,12);
getContentPane().add(loc0);
loc0.setVisible(false);

loc1.setBounds(70,140,12,12);
getContentPane().add(loc1);
loc1.setVisible(false);
```

```
    View_Btn.setText("Edit Groups");
    getContentPane().add(View_Btn);
    View_Btn.setBounds(15,265,100,15);
    View_Btn.setFont(f);

    getContentPane().add(Clear_Btn);
    Clear_Btn.setBounds(155,165,70,15);
    Clear_Btn.setFont(f);
    Clear_Btn.setVisible(false);

    getContentPane().add(Set_Btn);
    Set_Btn.setBounds(15,165,70,15);
    Set_Btn.setFont(f);
    Set_Btn.setVisible(false);

    Loc_Btn.setText("View Locations");
    getContentPane().add(Loc_Btn);
    Loc_Btn.setBounds(15,235,115,15);
    Loc_Btn.setFont(f);

    Update_Btn.setText("Update");
    getContentPane().add(Update_Btn);
    Update_Btn.setBounds(150,235,75,15);
    Update_Btn.setFont(f);

    Close_Btn.setText("Close");
    getContentPane().add(Close_Btn);
    Close_Btn.setBounds(150,265,75,15);
    Close_Btn.setFont(f);

    ip.setEditable(false);
    mask.setEditable(false);
    key.setEditable(false);
    crkt.setEditable(false);
    beaconID.setEditable(false);

    SymMouse aSymMouse = new SymMouse();
    SymAction aSymAction = new SymAction();
    Add_Btn.addMouseListener(aSymMouse);
    View_Btn.addMouseListener(aSymMouse);
    Close_Btn.addMouseListener(aSymMouse);
    Update_Btn.addMouseListener(aSymMouse);
    Clear_Btn.addMouseListener(aSymMouse);
    Set_Btn.addMouseListener(aSymMouse);
    Loc_Btn.addMouseListener(aSymMouse);
    loc0.addMouseListener(aSymMouse);
    loc1.addMouseListener(aSymMouse);
    locP.addActionListener(aSymAction);

    updateDisplay("IP");
  }

  /**
   * reacquires and displays the location info for the selected tab
   */
  public void updateDisplay(String locType) {
```

```
    if (locType.equals("IP")) {
      String[] ipInfo = IPLocProvider.getIP();
      ip.setText(ipInfo[0]);
      mask.setText(ipInfo[1]);
      if (!ipInfo[0].equals("Not Available")) {
      key.setText(getKey(ipInfo[0], ipInfo[1]));
      ipInfo = IPLocProvider.getLocName();
      loc.setText(ipInfo[1]);

      if (ipInfo[1].equals("unknown")) {
        loc_list = new javax.swing.JComboBox(Location.LIST_LOC);
        loc_list.setBounds(85,140,130,15);
        loc_list.setFont(f);
        getContentPane().add(loc_list);
        Add_Btn.setVisible(true);
        loc0.setVisible(true);
        loc1.setVisible(true);
        loc_list.setVisible(true);
        Loc_Btn.setVisible(false);
        message.setText("Enter a name or select a location \nfrom the
drop-down list and \nclick Add to add this location");
        loc.setEditable(true);
      }
      else {
        message.setText("Your location = \n "+ipInfo[1]);
        loc0.setVisible(false);
        loc1.setVisible(false);
        loc_list.setVisible(false);
        Add_Btn.setVisible(false);
        Loc_Btn.setVisible(true);
      }
      }
      else {
      loc0.setVisible(false);
      loc1.setVisible(false);
      loc_list.setVisible(false);
      Add_Btn.setVisible(false);
      Loc_Btn.setVisible(true);
      loc.setText("Not Available");
      message.setText("No IP information is available at this time");
      }
    }
    else if (locType.equals("Cricket")) {
      String tempBeaconID = clp.getLocation();
      beaconID.setText(tempBeaconID);
      if(!tempBeaconID.equals("Not Available")) {
      String fromProvider = CricketLocProvider.getLoc(tempBeaconID);
      loc.setText(fromProvider);
      if((loc.getText()).equals("unknown")) {
        loc_list = new javax.swing.JComboBox(Location.LIST_LOC);
        loc_list.setBounds(85,140,130,15);
        loc_list.setFont(f);
        getContentPane().add(loc_list);
        Add_Btn.setVisible(true);
        loc0.setVisible(true);
        loc1.setVisible(true);
        loc_list.setVisible(true);
```

```
        Loc_Btn.setVisible(false);
        message.setText("Enter a name or select a location \nfrom the
drop-down list and \nclick Add to add this location");
        loc.setEditable(true);
      }
      else {
        message.setText("Your location = \n "+loc.getText() );
        loc0.setVisible(false);
        loc1.setVisible(false);
        loc_list.setVisible(false);
        Add_Btn.setVisible(false);
        Loc_Btn.setVisible(true);
      }
      }
      if(tempBeaconID.equals("Not Available")) {
      message.setText("No Cricket information available");
      loc0.setVisible(false);
      loc1.setVisible(false);
      loc_list.setVisible(false);
      Add_Btn.setVisible(false);
      Loc_Btn.setVisible(true);
      loc.setText("Not Available");
      }
    }
    else if (locType.equals("Manual")) {
      String s = LocationService.getManualLocation();
      loc.setText(s);
      if (s.equals("Location Not Set")) {
      message.setText("Choose your location from the list \nand click
Set or clear the location\nby clicking on Clear");
      }
      else {
      message.setText("Manual location set to\n"+s);
      }
      loc0.setVisible(false);
      loc1.setVisible(false);
      loc_list = new javax.swing.JComboBox(Location.LIST_LOC);
      loc_list.setBounds(85,140,130,20);
      loc_list.setFont(f);
      getContentPane().add(loc_list);
      loc_list.setVisible(true);
      Add_Btn.setVisible(false);
      Loc_Btn.setVisible(true);
      Clear_Btn.setVisible(true);
      Set_Btn.setVisible(true);
    }
  }

  public void setVisible(boolean b) {
    if (b)
    setLocation(50, 50);
    super.setVisible(b);
  }

  static public void main(String args[]) {
    (new LocationGUI()).setVisible(true);
  }
```

```
  /**
   * This method acquirs the ip subnet key
   */
  public String getKey(String ip, String m) {
    /**
     * Figure out the number of significant bits
     */
    StringTokenizer st = new StringTokenizer(m, ".");
    String byte1 = st.nextToken();
    String byte2 = st.nextToken();
    String byte3 = st.nextToken();
    int end;
    if (byte1.equals("255")) {
      if (byte2.equals("255")) {
      if (byte3.equals("255")) {
        end = 24;
      }
      else  end = 16;
      }
      else end = 0;
    }
    else end =0;

    /**
     * Figure out the key
     */
    st = new StringTokenizer(ip, ".");
    String segment = st.nextToken() + "." + st.nextToken();
    String key;
    if (end == 16) {
      key = segment + ".0.0/16";
    }
    else if (end == 24) {
      String next = st.nextToken();
      key = segment + "." + next + ".0/24";
    }
    else
      key = "error";

    return(key);
  }

  public void addNotify() {
    Dimension size = getSize();
    super.addNotify();
    if (frameSizeAdjusted)
      return;
    frameSizeAdjusted = true;
    Insets insets = getInsets();
    javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
    int menuBarHeight = 0;
    if (menuBar != null)
      menuBarHeight = menuBar.getPreferredSize().height;
    setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
  }
```

```
    boolean frameSizeAdjusted = false;
    String[] data = {"IP","Cricket","Manual","Wireless"};
    javax.swing.JLabel beaconID_L = new javax.swing.JLabel("BeaconID");
    javax.swing.JLabel beaconLoc_L = new javax.swing.JLabel();
    javax.swing.JTextArea beaconID = new javax.swing.JTextArea();
    javax.swing.JLabel ip_L = new javax.swing.JLabel("IP");
    javax.swing.JLabel mask_L = new javax.swing.JLabel("Mask");
    javax.swing.JLabel key_L = new javax.swing.JLabel("Key");
    javax.swing.JLabel crkt_L = new javax.swing.JLabel("Cricket");
    javax.swing.JLabel loc_L = new javax.swing.JLabel();
    javax.swing.JLabel header = new javax.swing.JLabel();
    javax.swing.JTextArea message = new javax.swing.JTextArea();
    javax.swing.JComboBox locP = new javax.swing.JComboBox(data);
    javax.swing.JTextArea ip = new javax.swing.JTextArea();
    javax.swing.JTextArea mask = new javax.swing.JTextArea();
    javax.swing.JTextArea key = new javax.swing.JTextArea();
    javax.swing.JTextArea crkt = new javax.swing.JTextArea();
    javax.swing.JTextArea loc = new javax.swing.JTextArea();
    javax.swing.JComboBox loc_list = new
javax.swing.JComboBox(Location.LIST_LOC);
    javax.swing.JRadioButton loc0 = new javax.swing.JRadioButton();
    javax.swing.JRadioButton loc1 = new javax.swing.JRadioButton();
    javax.swing.JLabel MessageL = new javax.swing.JLabel();
    javax.swing.JButton Add_Btn = new javax.swing.JButton();
    javax.swing.JButton Set_Btn = new javax.swing.JButton("Set");
    javax.swing.JButton Clear_Btn = new javax.swing.JButton("Clear");
    javax.swing.JButton View_Btn = new javax.swing.JButton();
    javax.swing.JButton Loc_Btn = new javax.swing.JButton();
    javax.swing.JButton Close_Btn = new javax.swing.JButton();
    javax.swing.JButton Update_Btn = new javax.swing.JButton();


    class SymMouse extends java.awt.event.MouseAdapter {
      public void mouseClicked(java.awt.event.MouseEvent event) {
      Object object = event.getSource();
      if (object == Add_Btn)
        Add_Btn_mouseClicked(event);
      if (object == View_Btn)
        View_Btn_mouseClicked(event);
      if (object == Close_Btn)
        Close_Btn_mouseClicked(event);
      if (object == Update_Btn)
        Update_Btn_mouseClicked(event);
      if (object == Set_Btn) {
        String l = (String)loc_list.getSelectedItem();
        loc.setText(l);
        LocationService.setManualLocation(l);
        message.setText("Manual location set to\n"+l+"!");
      }
      if (object == Clear_Btn) {
        LocationService.clearManualLocation();
        loc.setText("Location Not Set");
        message.setText("Manual location cleared!");
      }
      if (object == loc0) {
        name = true;
```

```
      loc0.setSelected(true);
      loc1.setSelected(false);
    }
    if (object == loc1) {
      name = false;
      loc0.setSelected(false);
      loc1.setSelected(true);
    }
    if (object == Loc_Btn) {
      (new AllLocGUI()).setVisible(true);
    }
    }
  }

  class SymAction implements ActionListener {
    public void actionPerformed(ActionEvent e) {
    JComboBox cb = (JComboBox)e.getSource();
    if (((String)cb.getSelectedItem()).equals("IP")) {
      ip.setVisible(true);
      mask.setVisible(true);
      key.setVisible(true);
      ip_L.setVisible(true);
      mask_L.setVisible(true);
      key_L.setVisible(true);
      beaconID_L.setVisible(false);
      beaconID.setVisible(false);
      Clear_Btn.setVisible(false);
      Set_Btn.setVisible(false);
      updateDisplay("IP");
    }
    else if (((String)cb.getSelectedItem()).equals("Cricket")) {
      ip.setVisible(false);
      mask.setVisible(false);
      key.setVisible(false);
      ip_L.setVisible(false);
      mask_L.setVisible(false);
      key_L.setVisible(false);
      beaconID_L.setVisible(true);
      beaconID.setVisible(true);
      Clear_Btn.setVisible(false);
      Set_Btn.setVisible(false);
      updateDisplay("Cricket");
    }
    else if (((String)cb.getSelectedItem()).equals("Manual")) {
      ip.setVisible(false);
      mask.setVisible(false);
      key.setVisible(false);
      ip_L.setVisible(false);
      mask_L.setVisible(false);
      key_L.setVisible(false);
      beaconID_L.setVisible(false);
      beaconID.setVisible(false);
      updateDisplay("Manual");
    }
    else if (((String)cb.getSelectedItem()).equals("Wireless")) {
      ip.setVisible(false);
      mask.setVisible(false);
```

```
        key.setVisible(false);
        ip_L.setVisible(false);
        mask_L.setVisible(false);
        key_L.setVisible(false);
        beaconID_L.setVisible(false);
        beaconID.setVisible(false);
        Clear_Btn.setVisible(false);
        Set_Btn.setVisible(false);
        updateDisplay("Wireless");
      }
      }
    }

    void Add_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      String locType = (String)locP.getSelectedItem();
      if(locType.equals("IP")) {
      if ( name && (loc.getText().equals("") ||
loc.getText().equals("unknown")) ) {
        message.setText("Location name is not valid");
        System.out.println("Location name is not valid");
      }
      else {
        if (name) {
          IPLocProvider.addLocation(key.getText(), loc.getText());
        }
        else {
          IPLocProvider.addLocation(key.getText(),
(String)loc_list.getSelectedItem());
          loc.setText((String)loc_list.getSelectedItem());
        }
        message.setText("Location added successfully!");
        Add_Btn.setVisible(false);
        loc0.setVisible(false);
        loc1.setVisible(false);
        loc_list.setVisible(false);
        Loc_Btn.setVisible(true);
        loc.setEditable(false);
      }
      }
      if(locType.equals("Cricket")) {
      if ( name && (loc.getText().equals("") ||
loc.getText().equals("unknown")) ) {
        message.setText("Location name is not valid");
        System.out.println("Location name is not valid");
      }
      else {
        if (name) {
          CricketLocProvider.addLocation(beaconID.getText(),
loc.getText());
        }
        else {
          CricketLocProvider.addLocation(beaconID.getText(),
(String)loc_list.getSelectedItem());
          loc.setText((String)loc_list.getSelectedItem());
        }
        message.setText("Location added successfully!");
        Add_Btn.setVisible(false);
```

```
        loc0.setVisible(false);
        loc1.setVisible(false);
        loc_list.setVisible(false);
        Loc_Btn.setVisible(true);
        loc.setEditable(false);
      }
      }
    }

    void View_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      (new UpdateLocGUI()).setVisible(true);
    }

    void Close_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      this.setVisible(false);
    }

    void Update_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      updateDisplay((String)locP.getSelectedItem());
    }
}
```

IX.19 LocationService.java

```
/**
 * LocationService Class provides interface to get location data from
the
 * providers
 *
 * author Marc Bourget
 */

package SmartReminder.Location;
import SmartReminder.*;
import java.util.*;

public class LocationService implements Runnable{

    static private String loc = "";
    static private String manualLoc = "Location Not Set";
    static private boolean manualSet = false;
    private String oldIPloc;

    public LocationService() {
    }

    static public String getLocation() {
      return(loc);
    }

    static public String getManualLocation() {
      return(manualLoc);
    }

    static public void setManualLocation(String s) {
      manualSet = true;
      manualLoc = s;
```

```
    }

    static public void clearManualLocation() {
      manualSet = false;
      manualLoc = "Location Not Set";
    }

    public void run() {
      boolean locFound = false;
      try{
        while(true) {
        String[] locations = {"unknown","unknown"};
        int count = 0;
        System.out.println("Checking Location**************");
        /**
         * Check if location has been set manually
         */
        if (manualSet) {
            oldIPloc="unknown";
          if (!manualLoc.equals(loc)) {
            loc = manualLoc;
            System.out.println("Location Change**************Manual
Location is [["+loc+"]]");
            count = 1;
            locations[0] = loc;
            AlarmList.locationChanged(loc);
            System.out.println("AlarmList.locationChanged() called with
" + loc);
          }
        }
        else {
          String temp = "unknown";
          temp = CricketLocProvider.getCurrentLoc();
          if (!temp.equals("unknown")) {
            locFound = true;
            /**
             * check to see if location has changed
             */
            if (!temp.equals(loc)) {
            loc = temp;
            locations[0] = temp;
              count = 2;
            AlarmList.locationChanged(loc);
            System.out.println("AlarmList.locationChanged() called with
" + loc);
            }
          }
          temp = IPLocProvider.getCurrentLoc();
          if (!temp.equals("unknown")) {
            /**
             * if ip location has changed, inform the location based
alarms
             */
            if (!temp.equals(oldIPloc)) {
            locations[1] = temp;
            count = 2;
            oldIPloc = temp;
```

```
            }
            if (!locFound) {
            locFound = true;
            if (!temp.equals(loc)) {
              loc = temp;
              System.out.println("LocationAlarmList.locationChanged()
called with " + loc);
              AlarmList.locationChanged(loc);
                System.out.println("AlarmList.locationChanged() called
with " + loc);
            }
            }
          }
          if (!locFound) {
            /**
             *  If no providers are working, set the location to the
user defined default location
             */
            loc = SR.defaultLoc;
          }
        }
        if (count > 0) {
          LocationAlarmList.locationChanged(locations,count);
          if(count == 1) {
            System.out.println("LocationAlarmList.locationChanged()
called with array " + locations[0] + " and counter " + count);
          }
          if(count == 2) {
            System.out.println("LocationAlarmList.locationChanged()
called with array " + locations[0] + " and " + locations[1] + " and
counter " + count);
          }

        }
        Thread.sleep(SR.defaultUpdate);
        }
      }
      catch(InterruptedException e) {
      }
    }
}
```

IX.20 Meeting.java

```
/**
 * MeetingList is a repository style data structure that
 * holds Meeting
 * elements and allows adding, deleting and editing of
 * elements
 *
 * @author Craig Music
 */

package SmartReminder;

import java.util.*;
```

```
import java.io.Serializable;

public class Meeting extends GregorianCalendar implements Serializable
{
    private GregorianCalendar alarmDate = null, meetingDate = null;
    private String location = null, summary = null, meetingName = null;
    private boolean smart = false;
    private int notification;

    /**
     * Constructor, also creates object of type Alarm
     */
    public Meeting(GregorianCalendar mDate, String mName, String loc,
String sum, int notif, boolean sm) {
      meetingDate = mDate;
      meetingName = mName;
      location = loc;
      summary = sum;
      notification = notif;
      smart = sm;

      // Create alarm(s) for meeting and add it to AlarmList
      GregorianCalendar alert = copyDate( meetingDate );
      if ( smart ) {
      GregorianCalendar update = copyDate( meetingDate );
      update.add(update.HOUR_OF_DAY, -5);
      Alarm initial = new Alarm(update, location, sm, meetingDate,
meetingName);
      AlarmList.insert(initial);
      }

      if ( notification == 0 ) { //Corresponds to 15 minutes
      alert.add(alert.MINUTE, -10);
      }
      else if ( notification == 1 ) { //Corresponds to 20 minutes
      alert.add(alert.MINUTE, -20);
      }
      else if ( notification == 2 ) { //Corresponds to 30 minutes
      alert.add(alert.MINUTE, -30);
      }
      else if ( notification == 3 ) { //Corresponds to 40 minutes
      alert.add(alert.MINUTE, -40);
      }
      else if ( notification == 4 ) { //Corresponds to 50 minutes
      alert.add(alert.MINUTE, -50);
      }
      else if ( notification == 5 ) { //Corresponds to 1 hour
      alert.add(alert.HOUR_OF_DAY, -1);
      }
      else if ( notification == 6 ) { //Corresponds to 1 day
      alert.add(alert.DATE, -1);
      }
      Alarm notice = new Alarm(alert, location, sm, meetingDate,
meetingName);
      AlarmList.insert(notice);
    }
```

```
    /**
     * Returns if the current Meeting object is smart
     */
    public boolean isSmart() {
      return( smart );
    }

    /**
     * Returns the representation of how much before the actual
     * meeting should the system notify
     */
    public int getNotification() {
      return( notification );
    }

    /**
     * Returns the string representation of the Meetings description
     */
    public String getSummary() {
      return( summary );
    }

    /**
     * Returns the inputted location of this Meeting object
     */
    public String getLocation() {
      return( location );
    }

    /**
     * Returns the name of this meetings
     */
    public String getName() {
      return( meetingName );
    }

    /**
     * Returns the date of this Meeting object
     */
    public GregorianCalendar getMeetingDate() {
      return( meetingDate );
    }

    public GregorianCalendar copyDate( GregorianCalendar date ) {
      GregorianCalendar copy = new GregorianCalendar(
date.get(date.YEAR), date.get(date.MONTH), date.get(date.DATE),
date.get(date.HOUR_OF_DAY), date.get(date.MINUTE) );
      return( copy );
    }
}
```

IX.21 MeetingList.java

```
package SmartReminder;
import java.util.*;
import java.io.*;
```

```
public class MeetingList {

    static private Vector meetings = new Vector();
    static private boolean debug = false;
    static private boolean debug2 = false;

    static public void main(String[] args) {

    }

    static public void renew() {
      meetings = new Vector();
    }

    static public Meeting invalid() {
      GregorianCalendar iv = new GregorianCalendar(1970,1,1,1,1,1);
      Meeting ivv = new Meeting(iv, "invalid", "", "invalid", 1,
false);
      return ivv;
    }

    static public void insert(Meeting m) {
      GregorianCalendar meetingCalendar = m.getMeetingDate();
      int length = meetings.size()  ;
      int index = 0;
      while(index <= length) {
          if(length == length) {
            d("Meeting list empty or we reached the end, just add
meeting");
            //System.out.println("inserted meeting with date "
+m.getMeetingDate());
            meetings.insertElementAt(m, index);
            return;
          }
          GregorianCalendar tempCalendar =
((Meeting)meetings.elementAt(index)).getMeetingDate();
          if(index == length) {
            d("reached end of meeting list, just add");
            //System.out.println("inserted meeting with date "
+m.getMeetingDate());
            meetings.insertElementAt(m, index);
            return;
          }
          if(meetingCalendar.after(tempCalendar) ) {
            d("meeting is after the current one, move on");
            index++;
          }
          else {
            d("Meeting is before the current one, add it");
            meetings.insertElementAt(m, index);
            //System.out.println("inserted meeting with date "
+m.getMeetingDate());
            return;
          }
      }
    }
```

```
    static public void delete(Meeting m) {
      boolean happened = meetings.removeElement(m);
      d("the meeting was found and deleted " + happened);
      AlarmList.delete(m);
      /*
        int length = meetings.size()  ;
        int index = 0;
        while(index <= length) {
        Meeting temp = (Meeting)meetings.elementAt(index);
        if(m.equals(temp)) {
        meetings.removeElementAt(index);
        return;
        }
        }
      */
    }
    static public void e(String n) {
      if(debug2 ==true) {
          System.out.println(n);
      }
    }

    static public boolean areCalendarsEqual(GregorianCalendar a,
GregorianCalendar b) {
      int year1 = a.get(a.YEAR);
      int year2 = b.get(b.YEAR);
      int mo1 = a.get(a.MONTH);
      int mo2 = b.get(b.MONTH);
      int dt1 = a.get(b.DATE);
      int dt2 = b.get(b.DATE);
      int hr1 = a.get(b.HOUR_OF_DAY);
      int hr2  = b.get(b.HOUR_OF_DAY);
      e("\n\n comparison " + year1 + year2 + " month " + mo1 + mo2 + "
date " +dt1 + dt2 + " hour " +hr1 + hr2);
      if(year1==year2 &&mo1==mo2 && dt1==dt2 &&hr1==hr2) {
          return true;
      }
      else {
          return false;
      }
    }

    static public Meeting search(GregorianCalendar c) {
      int length = meetings.size() ;
      if(length == 0) {
          Meeting in = invalid();
          // System.out.println("this is the returned meeting datefrom
search meeting list " + in.getMeetingDate());
          return in;
      }

      int index;
      for(index = 0; index < length ; index++) {
          Meeting m = (Meeting) (meetings.elementAt(index));
          GregorianCalendar listCalendar = m.getMeetingDate();
          //  d("this is the inputed calendar create: "+ c.toString());
```

```
          // d("this is the calendar from the list " +
tempCalendar.toString());
          //if the year, month,date, hour match then you're set
          if(areCalendarsEqual(listCalendar,c )) {
            d("found meeting that matches date, return it");

            Meeting t = (Meeting)meetings.elementAt(index);
            // System.out.println("this is the returned meeting
datefrom search meeting list " + t.getMeetingDate());

                return t;
          }
      }
      d("did not find meeing that matches date");
      Meeting in2 = invalid();
      // System.out.println("this is the returned meeting datefrom
search meeting list " + in2.getMeetingDate());
      return in2;

    }

    static public boolean save()
    {
      boolean result = false;
      System.out.println("serializing Meetings Vector");
      try {
      FileOutputStream fout = new FileOutputStream("MeetingList.dat");
      ObjectOutputStream oos = new ObjectOutputStream(fout);
      oos.writeObject(meetings);
      oos.close();
      result = true;
      }
      catch (Exception e)
      {
         result = false;
         e.printStackTrace();
         return result;
      }
      return result;
    }

    static public boolean read()
    {
       meetings = new Vector();
       boolean result = false;
       // unserialize the vector
       System.out.println("unserializing MeetingList");
       try {
      FileInputStream fin = new FileInputStream("MeetingList.dat");
      ObjectInputStream ois = new ObjectInputStream(fin);
      meetings = (Vector) ois.readObject();
      ois.close();
      result = true;
      }
       catch (Exception e)
      {
        e.printStackTrace();
```

```
        result = false;
        return result;
      }
       return result;
       }

    static public void d(String m) {
      if (debug == true) {
          System.out.println(m);
      }
    }
}
```

IX.22 ReminderUpdater.java

```
/**
 * Used as threading mechanism to schedule tasks according to the next
Alarms
 * in queue.
 *
 * @author: Craig Music
 */

package SmartReminder;
import java.util.TimerTask;

 public class ReminderUpdater extends TimerTask {
      public void run() {
             System.out.println("ReminderUpdater.run called");
             Alarm temp = AlarmList.next();
             if (SR.isValid(temp)) {
                temp.alert();
                SR.updateTimeToNextAlarm();
            }
            else {
                SR.updateTimeToNextAlarm();
            }
       }
     }
```

IX.23 SettingsGUI.java

```
/**
 * SettingsGUI allows the user to customize some features of the app
 *
 * author Marc Bourget
 */

package SmartReminder;

import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.*;
```

```
import SmartReminder.Location.*;

public class SettingsGUI extends javax.swing.JFrame {
  private Font f = new Font("dialog",Font.PLAIN,10);
  private boolean name = true;

  public SettingsGUI() {
    setTitle("SettingsGUI");
    getContentPane().setLayout(null);
    setSize(240,320);
    setVisible(true);

    header.setText("Edit program settings");
    getContentPane().add(header);
    header.setBounds(15,10,145,15);
    header.setFont(f);

    getContentPane().add(defLoc_L);
    defLoc_L.setBounds(15,35,75,15);
    defLoc_L.setFont(f);

    loc_list.setBounds(95,35,120,15);
    getContentPane().add(loc_list);
    loc_list.setFont(f);
    loc_list.setSelectedItem(SR.defaultLoc);

    getContentPane().add(updateTime_L);
    updateTime_L.setBounds(15,60,160,15);
    updateTime_L.setFont(f);

    updateTime.setBounds(155,60,50,15);
    getContentPane().add(updateTime);
    updateTime.setFont(f);
    updateTime.setSelectedItem(Long.toString(SR.defaultUpdate/1000));

    getContentPane().add(sec_L);
    sec_L.setBounds(205,60,30,15);
    sec_L.setFont(f);

    getContentPane().add(waitTime_L);
    waitTime_L.setBounds(15,85,95,15);
    waitTime_L.setFont(f);

    waitTime.setBounds(115,85,50,15);
    getContentPane().add(waitTime);
    waitTime.setFont(f);
    waitTime.setSelectedItem(Long.toString(SR.defaultWait/1000));

    getContentPane().add(sec1_L);
    sec1_L.setBounds(170,85,30,15);
    sec1_L.setFont(f);

    offsetTime_L.setText("Offset Date (in hours) by");
    getContentPane().add(offsetTime_L);
    offsetTime_L.setBounds(15,115,120,20);
    offsetTime_L.setFont(f);
```

```
    offsetTime.setBounds(145,115,50,15);
    getContentPane().add(offsetTime);
    offsetTime.setFont(f);
    offsetTime.setSelectedItem(Long.toString(SR.defaultOffset));

    message.setBounds(15,185,200,45);
    getContentPane().add(message);
    message.setFont(f);
    message.setEditable(false);
    message.setForeground(Color.white);
    message.setBackground(Color.black);

    Save_Btn.setBounds(15,265,75,15);
    Save_Btn.setFont(f);
    getContentPane().add(Save_Btn);

    loc0.setSelected(true);
    loc0.setBounds(70,115,12,12);
    getContentPane().add(loc0);
    loc0.setVisible(false);

    loc1.setBounds(70,140,12,12);
    getContentPane().add(loc1);
    loc1.setVisible(false);

    getContentPane().add(Restore_Btn);
    Restore_Btn.setBounds(15,235,180,15);
    Restore_Btn.setFont(f);

    getContentPane().add(Close_Btn);
    Close_Btn.setBounds(150,265,75,15);
    Close_Btn.setFont(f);

    SymMouse aSymMouse = new SymMouse();
    Save_Btn.addMouseListener(aSymMouse);
    Close_Btn.addMouseListener(aSymMouse);
    Restore_Btn.addMouseListener(aSymMouse);

    loc0.addMouseListener(aSymMouse);
    loc1.addMouseListener(aSymMouse);
  }

  public void setVisible(boolean b) {
    if (b)
    setLocation(50, 50);
    super.setVisible(b);
  }

  static public void main(String args[]) {
    (new SettingsGUI()).setVisible(true);
  }

  /**
   * initializes the settings according to the settings file
   */
  public static void init() {
    DataInputStream input;
```

```
    StringBuffer b = new StringBuffer();
    String data = "";
    int charint = 0;

    try {
      input = new DataInputStream(new FileInputStream( "Settings" ) );
      while((charint=input.read())!=-1) {
      b.append((char)charint);
      }
    }
    catch (IOException e) {
      System.err.println( "Error opening or reading file\n" +
e.toString() );
      System.exit(1);
    }
    data = b.toString();
    StringTokenizer st = new StringTokenizer(data, "\n"); // first
separate the data into individual lines
    String line,name,loc;
    Vector temp = new Vector();
    while(st.hasMoreTokens()) {
      temp.add(st.nextToken());
    }
    SR.defaultLoc = (String)temp.elementAt(0);
    SR.defaultUpdate =
Long.valueOf((String)temp.elementAt(1)).longValue()*1000;
    SR.defaultWait =
Long.valueOf((String)temp.elementAt(2)).longValue()*1000;
    SR.defaultOffset =
Integer.valueOf((String)temp.elementAt(3)).intValue();
  }

  public void save() {
    SR.defaultLoc = (String)loc_list.getSelectedItem();
    SR.defaultUpdate =
Long.valueOf((String)updateTime.getSelectedItem()).longValue()*1000;
    SR.defaultWait =
Long.valueOf((String)waitTime.getSelectedItem()).longValue()*1000;
    SR.defaultOffset =
Integer.valueOf((String)offsetTime.getSelectedItem()).intValue();

    String data = SR.defaultLoc+"\n"+
              (String)updateTime.getSelectedItem()+"\n"+
              (String)waitTime.getSelectedItem()+"\n"+
              SR.defaultOffset;

    try {
      FileWriter fw = new FileWriter( "Settings",false );
      fw.write(data);
      fw.close();
    }
    catch (IOException e) {
      System.err.println( "Error opening or writing to file\n" +
e.toString() );
      System.exit(1);
    }
  }
```

```
   public void addNotify() {
    Dimension size = getSize();
    super.addNotify();
    if (frameSizeAdjusted)
      return;
    frameSizeAdjusted = true;
    Insets insets = getInsets();
    javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
    int menuBarHeight = 0;
    if (menuBar != null)
      menuBarHeight = menuBar.getPreferredSize().height;
    setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
  }


    boolean frameSizeAdjusted = false;
    String[] updateTimes = {"2","5","10","20","30","60","120","300"};
    String[] waitTimes = {"2","5","10","30","60","300"};
    String[] offsetTimes = {"-5","-4","-3","-2","-
1","0","1","2","3","4","5"};

    javax.swing.JLabel defLoc_L = new javax.swing.JLabel("Def.
Location");
    javax.swing.JLabel updateTime_L = new javax.swing.JLabel("Update
location info every");
    javax.swing.JLabel sec_L = new javax.swing.JLabel("sec");
    javax.swing.JLabel sec1_L = new javax.swing.JLabel("sec");
    javax.swing.JLabel waitTime_L = new javax.swing.JLabel("Def. Check
Time");
    javax.swing.JLabel offsetTime_L = new javax.swing.JLabel();
    javax.swing.JLabel crkt_L = new javax.swing.JLabel("Cricket");
    javax.swing.JLabel header = new javax.swing.JLabel();
    javax.swing.JTextArea message = new javax.swing.JTextArea();
    javax.swing.JComboBox loc_list = new
javax.swing.JComboBox(Location.LIST_LOC);
    javax.swing.JComboBox updateTime = new
javax.swing.JComboBox(updateTimes);
    javax.swing.JComboBox waitTime = new
javax.swing.JComboBox(waitTimes);
    javax.swing.JComboBox offsetTime = new
javax.swing.JComboBox(offsetTimes);
    javax.swing.JTextArea crkt = new javax.swing.JTextArea();
    javax.swing.JRadioButton loc0 = new javax.swing.JRadioButton();
    javax.swing.JRadioButton loc1 = new javax.swing.JRadioButton();
    javax.swing.JLabel MessageL = new javax.swing.JLabel();
    javax.swing.JButton Save_Btn = new javax.swing.JButton("Save");
    javax.swing.JButton Close_Btn = new javax.swing.JButton("Close");
    javax.swing.JButton Restore_Btn = new javax.swing.JButton("Restore
Default Settings");

    class SymMouse extends java.awt.event.MouseAdapter {
      public void mouseClicked(java.awt.event.MouseEvent event) {
      Object object = event.getSource();
      if (object == Save_Btn)
        Save_Btn_mouseClicked(event);
```

```
      if (object == Close_Btn)
        Close_Btn_mouseClicked(event);
      if (object == Restore_Btn)
        Restore_Btn_mouseClicked(event);
      }
    }


    void Save_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      save();
      message.setText("Settings saved");
    }

    void Close_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      this.setVisible(false);
    }

    void Restore_Btn_mouseClicked(java.awt.event.MouseEvent event) {
      loc_list.setSelectedItem("Akamai");
      updateTime.setSelectedItem("30");
      waitTime.setSelectedItem("300");
      offsetTime.setSelectedItem("-5");
      save();
      message.setText("Settings reset to their defaults");
    }
}
```

IX.24 SR.java

```
/**
 * Main class of the system, initialises the location providers and
starts the
 * main GUI.
 *
 * @author: Marc Bourget and Craig Music
 */

package SmartReminder;

import java.io.*;
import java.util.*;
import SmartReminder.Location.*;
import java.net.*;
import java.awt.*;

public class SR  {

  static public long timeToNextAlarm = 0;
  static public long defaultWait = 2000;
  static public String defaultLoc;
  static public long defaultUpdate = 30000;
  static public int defaultOffset = 24;

  public static java.util.Timer timer;
  public static ReminderUpdater remUp;

    static public void main(String[] args) {
```

```
        SettingsGUI.init();  //obtains initial settings
        IPLocProvider.init(); //initializes location provider
        CricketLocProvider.init(); //initializes location provider
        String[] cricketargs;
        try { //should use IP as commandline argument but just in case
check for null pointer
        String ip = "default";
        ip = args[1];
        cricketargs = new String[] {"-g"+ip,"-S"}; //needed arguments to
start cricket tracking
        }
        catch(Exception e) {
        cricketargs = new String[] {"-g","-S"};
        };
        CricketLocProvider clp = new CricketLocProvider(cricketargs);
        clp.d = false; // debug variable in cricketlocprovider that
generates location output
        Group.init();
        DistanceToTravel.init(); //initializes TimeToTravel matrix
        timer = new java.util.Timer(true);
        SR.remUp = new ReminderUpdater(); //starts Reminder Updaer

      try
      {
        if(args[0].equals("disk")) //tries to read state from disk
        {
          AlarmList.read();
          MeetingList.read();
          LocationAlarmList.read();
        }
        else
        {
          AlarmList.renew();
          MeetingList.renew();
          LocationAlarmList.renew();
        }
      }
      catch(ArrayIndexOutOfBoundsException e)
      {
          AlarmList.renew();
          MeetingList.renew();
          LocationAlarmList.renew();
      }
      (new StartGUI()).setVisible(true);
      timer.schedule(remUp, 0);

      // create LocationService thread
      Thread locServ = new Thread(new LocationService());
      locServ.start();

    }


    /**
     * This method queries the AlarmList and sets the system to sleep
     * until the next Alarm object inside AlarmList should trigger.
     */
```

```
    static public long updateTimeToNextAlarm() {
         SR.timeToNextAlarm = AlarmList.timeToNextAlarm();
         SR.remUp.cancel();
         SR.timer.cancel();
         remUp = null;
         remUp = new ReminderUpdater();
         timer = null;
         timer = new Timer(true);
         timer.schedule(remUp, SR.timeToNextAlarm+1000);

         System.out.println("updateTimeToNextAlarm in SR called, next
schedule execute in = "+SR.timeToNextAlarm);
         // call timer.cancel
         // make new timer with the new estimate of the next alarm
happening time
          return SR.timeToNextAlarm;
    }



    public SR() {


    }


    /**
     * This procedure assumes that if the date of the alarm is
     * 1970 (the biggining of time) then the date is invalid and we
     * don't do anything. Else the alarm is assumed to be valid and
     * it is triggered
     */
    public static boolean isValid( Alarm a ) {
      GregorianCalendar gc = a.getAlarmDate();
      int yr = gc.get(gc.YEAR);
      if ( yr==1970 )
          return( false );
      else return( true );

    }
}
```

IX.25 StartGUI.java

```
/**
 * StartGUI is the navigation/menu GUI for the application
 *
 * author Marc Bourget
 */

package SmartReminder;

import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.lang.*;
```

```
import SmartReminder.Location.*;

public class StartGUI extends javax.swing.JFrame
{
	private Font f = new Font("Dialog", Font.BOLD, 10);
	public StartGUI()
	{
		getContentPane().setLayout(null);
		setTitle("SMART REMINDER");
		setSize(240,320);
		setVisible(false);
		display.setText("Smart Reminder");
		getContentPane().add(display);
		display.setFont(new Font("Dialog", Font.BOLD, 14));
		display.setBounds(45,12,150,20);
		LocG.setBounds(45,100,150,25);
		getContentPane().add(LocG);
		DayG.setBounds(45,50,150,25);
		getContentPane().add(DayG);
		LocAG.setBounds(45,150,150,25);
		getContentPane().add(LocAG);
		Pref.setBounds(45,200,150,25);
		getContentPane().add(Pref);

		SymMouse aSymMouse = new SymMouse();
		LocG.addMouseListener(aSymMouse);
		DayG.addMouseListener(aSymMouse);
		LocAG.addMouseListener(aSymMouse);
		Pref.addMouseListener(aSymMouse);
	}

	public void setVisible(boolean b) {
		if (b)
			setLocation(50, 50);
		super.setVisible(b);
	}

	static public void main(String args[])
	{
	    (new StartGUI()).setVisible(true);
	}

	public void addNotify()
	{
		Dimension size = getSize();
		super.addNotify();
		if (frameSizeAdjusted)
			return;
		frameSizeAdjusted = true;
		Insets insets = getInsets();
		javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
		int menuBarHeight = 0;
		if (menuBar != null)
			menuBarHeight = menuBar.getPreferredSize().height;
		setSize(insets.left + insets.right + size.width, insets.top
+ insets.bottom + size.height + menuBarHeight);
	}
```

```
      boolean frameSizeAdjusted = false;
      javax.swing.JLabel display = new javax.swing.JLabel();
      javax.swing.JButton LocG = new
javax.swing.JButton("LocationGUI");
      javax.swing.JButton DayG = new javax.swing.JButton("DayGUI");
      javax.swing.JButton LocAG = new
javax.swing.JButton("LocationAlarmView");
      javax.swing.JButton Pref = new
javax.swing.JButton("Preferences");

      class SymMouse extends java.awt.event.MouseAdapter
      {
        public void mouseClicked(java.awt.event.MouseEvent event) {
          Object object = event.getSource();
          if (object == LocG) {
            (new LocationGUI()).setVisible(true);
          }
          if (object == DayG) {
            (new DayGUI()).setVisible(true);
          }
          if (object == LocAG) {
            (new LocationAlarmViewerGUI()).setVisible(true);
          }
          if (object == Pref) {
            (new SettingsGUI()).setVisible(true);
          }
        }
      }
}
```

IX.26 UpdateLocGUI.java

```
/**
 * UpdateLocGUI Allows user to add and remove the locations in a group
 * as well as add or remove groups
 *
 * author Marc Bourget
 */

package SmartReminder.Location;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.lang.*;
import java.awt.event.*;
import java.io.*;

public class UpdateLocGUI extends javax.swing.JFrame
{
    private boolean g = false;
    private Font f = new Font("dialog",Font.PLAIN,10);
    public UpdateLocGUI()
    {
      setTitle("Edit Groups");
```

```
        getContentPane().setLayout(null);
        setSize(240,320);
        setVisible(true);

        All_L.setText("All Locations");
        getContentPane().add(All_L);
        All_L.setBounds(10,10,100,15);
        All_L.setFont(f);

        getContentPane().add(groups);
        groups.setBounds(10,125,100,15);
        groups.setFont(f);

          getContentPane().add(all_loc);
          getContentPane().add(all_scrollPane);
          all_loc.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
          all_scrollPane.setBounds(10,25,130,85);
        all_loc.setFont(f);
          all_scrollPane.getViewport().setView(all_loc);

          getContentPane().add(group_loc);
          getContentPane().add(group_scrollPane);

group_loc.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        group_scrollPane.setBounds(10,145,130,85);
        group_loc.setFont(f);
          group_scrollPane.getViewport().setView(group_loc);

        getContentPane().add(gNameL);
        gNameL.setBounds(10,195,60,15);
        gNameL.setFont(f);
        gNameL.setVisible(false);

        getContentPane().add(gName);
        gName.setBounds(75,195,140,15);
        gName.setFont(f);
        gName.setVisible(false);

        getContentPane().add(Add_Btn);
        Add_Btn.setBounds(150,85,70,15);
        Add_Btn.setFont(f);

        getContentPane().add(Remove_Btn);
        Remove_Btn.setBounds(145,145,80,15);
        Remove_Btn.setFont(f);

        getContentPane().add(Save_Btn);
        Save_Btn.setBounds(10,250,75,15);
        Save_Btn.setFont(f);

        getContentPane().add(Edit_Btn);
        Edit_Btn.setBounds(130,250,100,15);
        Edit_Btn.setFont(f);

        getContentPane().add(Cancel_Btn);
        Cancel_Btn.setBounds(155,250,75,15);
        Cancel_Btn.setFont(f);
```

```
      Cancel_Btn.setVisible(false);

      getContentPane().add(Close_Btn);
      Close_Btn.setBounds(155,270,75,15);
      Close_Btn.setFont(f);

      SymMouse aSymMouse = new SymMouse();
        SymAction aSymAction = new SymAction();
      Add_Btn.addMouseListener(aSymMouse);
      Remove_Btn.addMouseListener(aSymMouse);
        Close_Btn.addMouseListener(aSymMouse);
      Edit_Btn.addMouseListener(aSymMouse);
      Cancel_Btn.addMouseListener(aSymMouse);
        Save_Btn.addMouseListener(aSymMouse);
        groups.addActionListener(aSymAction);
   }


    public void setVisible(boolean b)
    {
      if (b)
          setLocation(50, 50);
      super.setVisible(b);
    }

    static public void main(String args[])
    {
      (new UpdateLocGUI()).setVisible(true);
    }

    public void addNotify()
    {
      Dimension size = getSize();

      super.addNotify();

      if (frameSizeAdjusted)
          return;
      frameSizeAdjusted = true;

      Insets insets = getInsets();
      javax.swing.JMenuBar menuBar = getRootPane().getJMenuBar();
      int menuBarHeight = 0;
      if (menuBar != null)
          menuBarHeight = menuBar.getPreferredSize().height;
      setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
    }

    boolean frameSizeAdjusted = false;

    javax.swing.JLabel All_L = new javax.swing.JLabel();
    javax.swing.JComboBox groups = new
javax.swing.JComboBox(Group.NAMES);
    javax.swing.JList all_loc = new
javax.swing.JList(Location.LIST_LOC);
```

```
    javax.swing.JScrollPane all_scrollPane = new
javax.swing.JScrollPane();
    javax.swing.JList group_loc = new
javax.swing.JList(((Vector)Group.GROUPS.get(0)));
    javax.swing.JScrollPane group_scrollPane = new
javax.swing.JScrollPane();
    javax.swing.JLabel gNameL = new javax.swing.JLabel("Group Name");
    javax.swing.JTextField gName = new javax.swing.JTextField();
    javax.swing.JButton Add_Btn = new javax.swing.JButton("Add");
    javax.swing.JButton Remove_Btn = new javax.swing.JButton("Remove");
    javax.swing.JButton Save_Btn = new javax.swing.JButton("Save");
    javax.swing.JButton Close_Btn = new javax.swing.JButton("Close");
    javax.swing.JButton Cancel_Btn = new javax.swing.JButton("Cancel");
    javax.swing.JButton Edit_Btn = new javax.swing.JButton("Edit
Groups");

    class SymMouse extends java.awt.event.MouseAdapter
    {
      public void mouseClicked(java.awt.event.MouseEvent event)
      {
          Object object = event.getSource();

          if (object == Add_Btn) {
            int i = groups.getSelectedIndex();
                String l = (String)all_loc.getSelectedValue();
                if ( !(((Vector)Group.GROUPS.get(i)).contains(l)) ) {
                  ((Vector)Group.GROUPS.get(i)).add(l);
                  group_loc.setListData( ((Vector)Group.GROUPS.get(i))
);
                }
            }
          else if (object == Remove_Btn) {
              int i = groups.getSelectedIndex();
              if (g) {
            Group.removeGroup(i);
            groups.setSelectedIndex(0);
              }
            else {
              String l = (String)group_loc.getSelectedValue();
                System.out.println("Index is: "+i+"\nand loc is: "+l);
                ((Vector)Group.GROUPS.get(i)).remove(l);
                group_loc.setListData( ((Vector)Group.GROUPS.get(i)) );
            }
            }
            else if (object == Close_Btn) {
            Close_Btn_mouseClicked(event);
            }
          else if (object == Edit_Btn) {
            Edit_Btn_mouseClicked(event);
            }
          else if (object == Cancel_Btn) {
            Cancel_Btn_mouseClicked(event);
            }
            else if (object == Save_Btn) {
                Save_Btn_mouseClicked(event);
            }
      }
```

```
}

class SymAction implements ActionListener
{
    public void actionPerformed(ActionEvent e) {
        JComboBox cb = (JComboBox)e.getSource();
        int i = groups.getSelectedIndex();
        String l = (String)all_loc.getSelectedValue();
        group_loc.setListData( ((Vector)Group.GROUPS.get(i)) );
   }
}

  void Close_Btn_mouseClicked(java.awt.event.MouseEvent event) {
    this.setVisible(false);
  }

  void Edit_Btn_mouseClicked(java.awt.event.MouseEvent event) {
    g = true;
    Edit_Btn.setVisible(false);
    Cancel_Btn.setVisible(true);
    gNameL.setVisible(true);
    gName.setVisible(true);
    group_loc.setVisible(false);
    groups.setBounds(10,145,100,15);
    Add_Btn.setVisible(false);
    Remove_Btn.setVisible(true);
    Save_Btn.setText("Add");
    Save_Btn.setBounds(85,225,70,15);
    Cancel_Btn.setBounds(10,270,75,15);
    groups.setVisible(true);
    group_scrollPane.setVisible(false);
  }

  void Cancel_Btn_mouseClicked(java.awt.event.MouseEvent event) {
    g = false;
    Edit_Btn.setVisible(true);
    Cancel_Btn.setVisible(false);
    gNameL.setVisible(false);
    gName.setVisible(false);
    group_loc.setVisible(true);
    Add_Btn.setVisible(true);
    Save_Btn.setText("Save");
    Save_Btn.setBounds(10,250,75,15);
    Cancel_Btn.setBounds(155,250,75,15);
    Remove_Btn.setVisible(true);
    groups.setVisible(true);
    groups.setBounds(10,125,100,15);
    group_scrollPane.setVisible(true);
  }

  void Save_Btn_mouseClicked(java.awt.event.MouseEvent event) {
    if (g) {
    String n = gName.getText();
    if (n.equals("")) {
      gName.setText("Invalid name");
      }
    else if (n.equals("Group Added")) {
```

```
        gName.setText("Invalid name");
        }
      else if (n.equals("Invalid name")) {
        gName.setText("Invalid name");
        }
      else if (n.equals("Group already exists")) {
        gName.setText("Invalid name");
        }
      else if (n.equals("Same name as a location")) {
        gName.setText("Invalid name");
        }
        else if ( Group.NAMES.contains(n)) {
        gName.setText("Group already exists");
        }
      else if ( Location.LIST_LOC.contains(n) ) {
        gName.setText("Same name as a location");
        }
      else {
        Group.addGroup(n);
        gName.setText("Group Added");
      }
     }
     else {
      Group.saveVectors(); //save to groups file
     }
    }
}
```