Expanding the Wireless
Communication Paradigm

by

Marc A. Bourget
B.S. Electrical Engineering and 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

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.
Expanding the Wireless Communication Paradigm

by

Marc A. Bourget
B.S. Electrical Engineering and 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

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

Abstract

With the rapid advancement of technology, handheld devices are becoming more and more powerful. Taking advantage of these improvements can dramatically enhance a user's experience. This project's intent is to expand the device's capabilities. This thesis explains the project, whose goal is to improve the functionality of an iPAQ that is equipped with a wireless card and a Cricket listener (a location retrieval device). The focus of the project is a context aware Smart Reminder application. Smart Reminder relies on the Cricket Location System and the IP Location System for location retrieval. The application's design and implementation is documented in this paper. The project shows that when given the proper resources, a handheld device can be a very useful tool; capable of replacing many of the items the everyday person carries.
# TABLE OF CONTENTS

1 INTRODUCTION .................................................................................................................. 6
  1.1 OVERVIEW .................................................................................................................. 6
  1.2 BACKGROUND .......................................................................................................... 7
    1.2.1 MIT Project Oxygen ......................................................................................... 7
    1.2.2 Handhelds.org ................................................................................................. 9
  1.3 COLLABORATION WITH CRAIG MUSIC ............................................................. 10

2 LOCATION TECHNOLOGIES ....................................................................................... 10
  2.1 CRICKET LOCATION SYSTEM ............................................................................... 10
  2.2 IP LOCATION SYSTEM ......................................................................................... 12
  2.3 GPS LOCATION SYSTEM ...................................................................................... 13

3 SYSTEM DESIGN .......................................................................................................... 14
  3.1 FUNCTIONALITY OF SYSTEM .............................................................................. 14
    3.1.1 Schedule Reminders ....................................................................................... 15
    3.1.2 Schedule Location Alarms ............................................................................. 16
    3.1.3 User Preferences ............................................................................................ 17
  3.2 MODULES AND THEIR INTERACTIONS (HIGH-LEVEL) ..................................... 18
  3.3 LOW-LEVEL DESCRIPTION ................................................................................... 19
    3.3.1 User Interface .................................................................................................. 20
    3.3.2 MeetingList .................................................................................................... 26
    3.3.3 AlarmList ....................................................................................................... 26
    3.3.4 LocationAlarmList ......................................................................................... 28
    3.3.5 Location Service ............................................................................................ 29
    3.3.6 Cricket Location Provider ............................................................................. 29
    3.3.7 IP Location Provider ...................................................................................... 30
    3.3.8 Manual Location Provider ............................................................................ 31
    3.3.9 Groups ........................................................................................................... 31
    3.3.10 Time-To-Travel Matrix ............................................................................... 32

4 EVALUATION OF SYSTEM .......................................................................................... 32
  4.1 DESIGN ALTERNATIVES ....................................................................................... 32
    4.1.1 Java VS C ........................................................................................................ 33
    4.1.2 Linux ............................................................................................................. 33
    4.1.3 Location Service ............................................................................................. 34
    4.1.4 List Sorting .................................................................................................... 35
    4.1.5 System Files ................................................................................................... 36
  4.2 TEST RESULTS ........................................................................................................ 38
  4.3 EXPERIENCES WITH LOCATION SYSTEM ....................................................... 40

5 RELATED WORK .......................................................................................................... 41
  5.1 NetGeo – THE INTERNET GEOGRAPHIC DATABASE ....................................... 41

6 CONCLUSIONS ............................................................................................................. 42
  6.1 EXPANSION ............................................................................................................ 42
  6.2 FINAL THOUGHTS .................................................................................................. 43

7 ACKNOWLEDGEMENT ................................................................................................. 44

8 RESOURCES .................................................................................................................. 44

9 APPENDIX ................................................................................................................... 45
  9.1 ALARMJAVA ......................................................................................................... 45

Masters Thesis June 2002
| 9.2 | ALARMLIST.JAVA | 47 |
| 9.3 | ALERTGUIJAVA | 53 |
| 9.4 | ALLLocGUIJAVA | 55 |
| 9.5 | CRicketLocProvider | 62 |
| 9.6 | CURRENTDATE | 69 |
| 9.7 |DAYGUIJAVA | 70 |
| 9.8 | DISTANCEToTRAVELJAVA | 97 |
| 9.9 | ERRORGUIJAVA | 98 |
| 9.10 | GROUPJAVA | 100 |
| 9.11 | IPlcProviderJAVA | 102 |
| 9.12 | LOCATIONJAVA | 108 |
| 9.13 | LOCATIONALARMJAVA | 108 |
| 9.14 | LOCATIONALARMGUIJAVA | 112 |
| 9.15 | LOCATIONALARMLISTJAVA | 117 |
| 9.16 | LOCATIONALARMVIEWERGUIJAVA | 121 |
| 9.17 | LOCATIONALERTGUIJAVA | 126 |
| 9.18 | LOCATIONGUIJAVA | 128 |
| 9.19 | LOCATIONSERVICEJAVA | 137 |
| 9.20 | MEETINGJAVA | 139 |
| 9.21 | MEETINGLISTJAVA | 141 |
| 9.22 | REMINDERUPDATERJAVA | 144 |
| 9.23 | SETTINGSGUIJAVA | 144 |
| 9.24 | SRJAVA | 149 |
| 9.25 | STARTGUIJAVA | 151 |
| 9.26 | UPDATELocGUIJAVA | 153 |
LIST OF FIGURES

1. Cricket Beacon
2. Cricket Listener
3. StartGUI
4. DayGUI
5. MeetingGUI
6. LocationAlarmViewerGUI
7. LocationAlarmGUI
8. SettingsGUI
9. High Level System Block Diagram
10. AlertGUI
11. AllLocGUI
12. ErrorGUI
13. LocationAlertGUI
14. LocationGUI
15. UpdateLocGUI
16. CricketTable File
17. IPTable File
18. Groups File
19. Linux running on iPAQ
1 Introduction

1.1 Overview

The purpose of the thesis is to explore new ways to empower personal digital assistants (PDA's) that have wireless connectivity and knowledge of their locations in order to expand an individual's options. More specifically, the goal is to develop a Smart Reminder application, which runs on Compaq's iPAQ H3600. The iPAQ is equipped with an 802.11b card (for wireless connectivity) and a Cricket Listener (a location retrieval service).

The Smart Reminder application takes a user's current location into account when reminding him/her about future responsibilities. For example, the application does not alert a user about a meeting he is already in. It also warns him when he must depart for a meeting in order to make it on time (depending on the travel time to the destination). In other words, the program is context aware and its behavior changes depending on the location of the user.

Through collaboration with ongoing research at the MIT Lab for Computer Science's Oxygen project and Compaq's Cambridge Research facility, the outcome of the project is Smart Reminder, a solution that can potentially change the paradigm with which people see the interaction of computers and human beings. The primary focus of the thesis is to "enable people to do more by doing less, that is, to accomplish more with less work." [4]

The thesis is intended 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]
1.2 Background

The project was inspired by the research being conducted at MIT’s Lab for Computer Science and Handhelds.org. In fact, the work done for the thesis was possible because of the vast amount of information provided by the Handhelds site and users. The Linux operating system that runs on the iPAQ was downloaded from the Handhelds servers. The following two sections will describe some of the related work that is going on in Project Oxygen and at Handhelds.

1.2.1 MIT Project Oxygen

MIT’s Project Oxygen is striving to make people’s lives easier by building a system that customizes itself to meet the needs of its users. The fact that the system is “accessible through natural perceptual interfaces” allows people to effortlessly do their everyday tasks. [4]

Oxygen’s current system is composed of a network of mobile and stationary devices. In order to facilitate the easy entry and exit of devices into the system, the network is self-configuring. The system accommodates the users’ needs by providing computation and communication at all times on multiple levels of software technology. To achieve the project’s goals, the system must possess certain characteristics. According to the Oxygen’s objectives, the system must be:

- **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. [4]
The system cannot be fully integrated into a user’s life until these requirements are fulfilled.

Handheld devices are used as mobile points of interaction with the Oxygen system. Users may communicate with them by using speech or visual input. The devices have the added ability to reconfigure (adapt) themselves if the need to support more communication protocols or execute more jobs arises. Such added services include functioning as a cellular phone, beeper, radio, television, geographical positioning system, and several more. As a result, they replace many dedicated devices that a user may carry, thus reducing the weight of items to be carted around. In addition, the handheld devices are capable of passing communication and computation intensive processes on to the more powerful stationary machines to conserve power.

When additional software is needed, the device will locate and download it from the closest stationary or mobile device. The software can determine the device’s location to decide which communication protocols exist in the current setting. Such “adaptation enables smooth vertical hand-offs to different communication protocols, not just horizontal hand-offs to different transmitters using the same protocol.” [4]
1.2.2 Handhelds.org

Handhelds.org plays a significant role in the Open Source Software movement, which is primarily focused on handheld and wearable devices. The group’s goal is “to encourage and facilitate the creation of open source software for use on handheld and wearable computers.” [2] The site and group provide support for users wishing to contribute to this goal. The users generally fall into one of three categories: system software wizards, application designers, and end users. System software wizards are individuals striving to port operating systems to handheld devices while application designers implement application software that runs on supported hardware/OS platforms. As one would expect, the end users are those who use the software developed by the system software wizards and application designers.

For operating systems, the group focuses on “Unix-like” systems, such as Linux and an assortment of editions of BSD Unix. Three types of resources are provided with the aim of assisting in the progress of open software for handheld computers. An abundant source of information for all users can be found on the handheld.org web pages. The second resource is communication, which is provided through email lists, email archives, and IRC chat facilities. Communication allows the users to collaborate and coordinate their efforts. The final resource is software archiving and distribution. Several reliable servers act as software repositories, where users can upload and download free software. This is important for project teams who can use the servers as a repository for their software. Compaq Computer Corporation is currently sponsoring the group’s endeavors.
1.3 Collaboration With Craig Music

Collaboration with Craig Music was essential in the definition and development of the project. He contributed in all aspects of the project including the identification of the project's modules, their interactions, and their implementation. The thesis supervisor, Jamey Hicks, provided an incredible amount of consultation and guided the project to a successful completion.

2 Location Technologies

2.1 Cricket Location System

The Cricket Location System is an initiative from MIT's Network and Mobile Systems Group. As part of Project Oxygen, the NMS group has been working on developing a system, Cricket, [6] that would allow the acquisition of a user's physical context information and would consequently enhance the possible computational experience. This project focuses 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 system is easily deployable, scalable, and requires 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 utmost importance. The Cricket System is capable of estimating an x-y-z position to "within a few centimeters and ... detect boundaries to within 2 feet". [6] The actual implementation of the Cricket Location Technology is based on "beacons" that are deployed throughout a designated area and a "listener" device that is able detect RF and ultrasound signals emitted by the beacons and estimate
its position. 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 and waits to obtain the corresponding ultrasound signal. The listeners use the time at which they receive each signal to calculate their distance from the respective beacons. 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. The system protects the privacy of the user since 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. 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.
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 the measurement of the areas and configuration files that contain this data. As a result, it requires more time and lower mobility. In theory, triangulation from multiple beacons' readings pinpoints the location, although the practical implementation has encountered some problems that are currently being dealt with. Although it was tested, this capability was not used in the 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.

2.2 IP Location System

Internet protocol is based on efficiently routing information traveling on the internet. In order for routing be done suitably, it must, in turn, be based on a logical organization of
the participating networks. Fortunately, at MIT, the campus networks are organized in a rough geographical manner. The MIT network is set up in such way that each IP subnet is assigned to one building. Members of the MIT community can access a database, which contains all of institute’s subnets and their corresponding locations (buildings, areas). Given the nature of this structure, knowledge of the device’s IP address makes it possible to determine which building (on the campus’ network) the user is in. As a result, IP was included in the list of location technologies. Although IP does not give as accurate information (about location) as the Cricket system, it is rather helpful for the situations where Cricket is not available. This specifically allows the application to function properly and expand its usefulness beyond the Laboratory for Computer Science, where the Cricket System is currently deployed.

2.3 GPS Location System

Even less accurate than the Cricket Location System is the Global Positioning System (GPS). However, GPS is more accurate than the IP Location System and has a significant advantage over the other two location technologies: it will work from every location including the outdoors. This is a major necessity since Cricket systems and IP networks are not deployed at all locations. GPS was implemented by the US military and consists of a network of 24 satellites and 5 monitoring ground stations positioned at various locations (Hawaii, Diego Garcia, Kwajalein, Colorado Springs, and Ascension Island) across the globe. The system allows a GPS receiver to acquire precise positioning information. With some military applications, the system can calculate position with a margin of error less than a meter. Such accurate calculations are obtainable because the system uses triangulation with four (of the 25) satellites and timing to determine their
distance from the target. Since its implementation, GPS has been perfected to achieve the desired precision. It is now very successful and used by many industries including commercial airline flights and weapon targeting.

3 System Design

Several technologies for wireless communication and location retrieval services were utilized to fulfill the project’s requirements. This section describes the functionality of the system as well as the software that takes advantage of these technologies and makes up the bulk of the project. The first part describes what a user can expect when using the application. The second part gives a high level explanation of the modules and how they interact to achieve the desired level of performance. Once the structure of the system has been laid out, the remainder of the section will describe the system’s components in detail.

3.1 Functionality of System

This section goes into detail about some key functions of the application and how the user can go about performing them. Pictures of the graphical user interfaces (GUIs) are included to guide the reader in the explanations. When the user starts the Smart Reminder application, the StartGUI appears. This is essentially the navigation screen that allows the user to interface with different parts of the application.
3.1.1 Schedule Reminders

Scheduling a reminder is a rather straightforward process. On the StartGUI, the user should click on the DayGUI button. This should instantiate the DayGUI (Figure 4).

Once the DayGUI has appeared, the user may browse to the day of the event. When on the schedule of the desired day, the user must click on the hour of the event (if there is not already an event scheduled for that hour). This action will instantiate the MeetingGUI (Figure 5). To schedule the reminder, the user must fill in the proper fields, choose the location, notification and if the alarm for the event is to be smart, and click Create. In Figure 5's example, the user has entered information for a Thesis Meeting, scheduled for 11:00 in Building N42.

Once a reminder has been created, the user may view the details of it by clicking on its time slot in the DayGUI. For example, in Figure 4, to view the details of the Thesis Meeting reminder, the user should click on the 11:00 button. While viewing the details, the user may edit or delete the reminder by clicking on the Edit or Delete buttons.
3.1.2 Schedule Location Alarms

Like scheduling reminders, creating location alarms is a simple task. On the StartGUI, the user should click on the LocationAlarmView button, which will instantiate the LocationAlarmViewerGUI (Figure 6). Once the GUI appears, click on the Add Button to bring up the LocationAlarmGUI. On the LocationAlarmGUI, enter the name of the alarm, a summary, and choose the location or group for which the alarm should trigger and click Add. In Figure 7, a location alarm is scheduled to trigger when the user is in LCS (MIT’s Laboratory for Computer Science). When the user is in this location, the alarm will remind him to “Pick up 6.033 handouts.”

If the need to edit or delete a location alarm arises, the user may do so by selecting the location alarm in the LocationAlarmViewerGUI and clicking the Edit or Remove button respectively.
3.1.3 User Preferences

The user may edit the system settings by clicking the Preferences button on the StartGUI to bring up the SettingsGUI. Several variables affecting the application’s functionality can be edited in this GUI. When the system has no way of determining the user’s whereabouts, it will resort to the default location setting. The user can also specify how often the system attempts to calculate the device’s location and the offset (in hours) from the Greenwich time zone. Once the user is content with the settings of all the variables, he must click Save for any changes to take effect. In Figure 8’s example, when no location providers are available, the system will assume the user is at LCS. The device’s location will be updated every 30 seconds and the system will check for new alarms every 60 seconds. The application’s clock will be adjusted to the Eastern Time Zone, which is 5 hours before the Greenwich Time Zone. The user can change the settings at any point while the Smart Reminder application is running.
3.2 Modules and Their Interactions (High-level)

The application is comprised of six major modules that are shown in Figure 9.

- User Interface
- MeetingList
- AlarmList
- LocationAlarmList
- Location Service
- Time-to-Travel Matrix

![Figure 9: High Level System Block Diagram]

The interaction of these modules represents the core of the SmartReminder application.

The state of the system is maintained in the three lists: MeetingList, AlarmList, and LocationAlarmList. The User Interface provides a means of directly changing the contents of these lists. Alternatively, the lists may be updated via the Location Service.

After starting the application, the user may choose from the following options:

1 – Schedule a Reminder

2 – Schedule a Location Alarm

If the user desires to create a reminder, he may do so through the user interface. Once the proper information (time, location, summary, etc) has been provided, a Meeting object,
which retains all of the information regarding the new reminder, is added to the MeetingList and an Alarm object is added to the AlarmList.

Location alarm can also be scheduled via the user interface. Upon entering the required information (name, location/group, summary), a Location Alarm object corresponding to the user’s entries is inserted into the LocationAlarmList.

Scheduling reminders and location alarms are the only ways the user can insert items into the any of the three lists.

**Location Retrieval**

The location service runs in the background and regularly collects information regarding the user’s location. Upon receiving new information about the user’s whereabouts, the location service notifies the two alarm lists of any changes. When presented with this news, the reminder alarms calculate new notification times via the time-to-travel matrix. Once the alarms have updated their notification times, the AlarmList resorts them to keep the chronological order. The time-to-travel matrix simply returns the estimated time to travel from one location to another. When the Location Alarm List learns of a location change, it checks all of the location alarms and triggers the appropriate ones (if any are present).

**3.3 Low-level Description**

This section describes the purpose of each module in detail. Along with the purpose is an explanation of the components that make up the modules and how they are used in conjunction with each other.
3.3.1 User Interface

The user interface is comprised of 12 GUIs that allow the user to interact with the program. Each GUI is listed below with a brief explanation of its functionality and how it ties into the application.

AlertGUI

The AlertGUI displays an alert message warning the user that an event is approaching. It is displayed when a reminder’s alarm has been triggered. The alert message lists the reminder’s (event) name, time, and location. The AlertGUI also plays an ‘alarm tune’ in the event that the user is not looking at the iPAQ screen.

AllLocGUI

The AllLocGUI allows the user to view what information is known about a selected location. This information is either an IP address or a Cricket beacon and is gathered from both the IPTable and CricketTable. (See sections 3.3.6 and 3.3.7 for an explanation of the IPTable and CricketTable) The AllLocGUI gives a user the
ability to directly access the data tables. The user can choose to edit a location’s name or information, as it exists in the data tables. In addition to this functionality, locations can be added or deleted from the tables.

**DayGUI**

The DayGUI serves several purposes in the Smart Reminder application. With the DayGUI, the user can view the schedule for any given day. A schedule displays the day, month, and year of a given day with the added capability of advancing forward to the next day or going backwards to the previous day’s schedule. Also visible for each day are the time slots (10:00, 11:00, etc), which are either labeled with a reminder name if an event has been scheduled for that hour or blank if the slot is available.

If the user clicks on a slot that is available (no event has been scheduled), a MeetingGUI is instantiated where the user can create a reminder for that selected time slot. On the other hand, if slot labeled with an event’s name is clicked, a MeetingGUI, displaying the reminder’s information (name, summary, location, etc.), is instantiated. In this form of the GUI, the user can edit or delete the reminder. See Figure 4 for a screen shot of the DayGUI.

**ErrorGUI**

The ErrorGUI notifies the user that a reminder can not be created because a required field (title or summary) of the MeetingGUI has been left blank. This GUI has no functionality and does not affect the state of the system.

![Figure 12: ErrorGUI](image)
MeetingGUI

The user can schedule a reminder in the MeetingGUI. The user must specify the event’s name, location, and summary, as well as the notification time and if the alarm should be smart or not. When selecting the location, the user may choose from all of the known locations or narrow down the choices by choosing a group. (Section 3.3.9 goes into more detail on Groups.) If the user tries to create a reminder with either the name or summary field left blank, the ErrorGUI will be instantiated. When all fields have been filled out properly, a meeting object and alarm object are created for the reminder and inserted into the MeetingList and AlarmList respectively.

The details of an existing reminder can be viewed in the MeetingGUI; all of the information about the event is displayed. The user can edit these details when the Edit button is clicked. If a reminder’s information has been changed, the corresponding meeting object is removed from the MeetingList and a new meeting with the updated information is created and added. Clicking on the Delete button will delete the meeting object and its related alarm objects from the proper lists, as one would expect.

The MeetingGUI represents the user’s only means of directly editing the contents of the MeetingList and AlarmList. See Figure 5 for a screenshot of the MeetingGUI.

LocationAlarmGUI

The user may create a location alarm in the LocationAlarmGUI. In order to add a location alarm to the system, the user must specify a name, location or group, and a summary for the alarm. A location alarm can be triggered when the user is either at a specific location or at one of several locations in a group. For example, the user may want to create a location alarm reminding him to print out his resume when he is in the vicinity of a
printer. Obviously, there are multiple locations that have a printer so he can create a
group named ‘rooms-printer’ and be reminded of his task when he is in any of the
locations belonging to this group. It is therefore necessary to give a user the option of
selecting a location or group when making a location alarm. When a location alarm has
been created, it is added to the LocationAlarmList.

The LocationAlarmGUI is also used to edit the information of an existing alarm. See
Figure 7 for a screen shot of the LocationAlarmGUI.

LocationAlarmViewerGUI
The LocationAlarmViewerGUI allows the user to view all of the location alarms that
exist in the system. These alarms are presented just as they reside in the
LocationAlarmList. Each location alarm’s name, location/group, and creation date is
listed in a table for the user. A location alarm can be deleted if it is highlighted when the
Remove button is clicked. Clicking the Add or Edit button will bring up the
LocationAlarmGUI where the user can either create a new location alarm or edit the
details of a location alarm. See Figure 6 for a screen shot of the
LocationAlarmViewerGUI.

LocationAlertGUI
The LocationAlertGUI displays an alert message
warning the user of a location alarm. It is
displayed when a location alarm has been
triggered, which occurs when the user is in a
location/group for which a location alarm exists.

![Figure 13: LocationAlertGUI](image)
The alert message lists the location alarm’s name, summary, and location or group. The LocationAlertGUI also plays an ‘alarm tune’ in the event that the user is not looking at the iPAQ screen.

LocationGUI

The LocationGUI displays information about the user’s current location according to the various location providers. The user can choose to view the data currently being collected by the IP Location Provider or the Cricket Location Provider. If the current information data does not exist in the system, the user may add it. (See Figure 14) If the user opts to view the IP Location Provider’s output, the current IP address, subnet mask, and the IPTable key and location (if an entry exists for this location) are displayed given that the iPAQ is communicating on some network. The Cricket Location Provider’s output is presented in the form of a Beacon ID, which is the closest beacon to the iPAQ. The user can manually set the location of the iPAQ in this GUI. Manually setting the location will override the output of the other location providers. This is useful in many instances, which are described in section on the Manual Location Provider. The Location Providers are discussed in further detail in sections 3.3.6-3.3.8.
SettingsGUI

The SettingsGUI allows the user to set some of the system variables to a desired value. These variables include the default location (in the event that none of the location providers are providing information), the frequency the system checks the user’s location, and the offset of the system time (in the event that the user has changed time zones). These variables are stored in a Settings file so that they can be restored when the user has exited and reopened the application. See Figure 8 for a screen shot of the SettingsGUI.

StartGUI

The StartGUI is the first thing that user sees when the Smart Reminder has been opened. It is a navigation or menu GUI that allows the user to access different areas of functionality of the system. From the StartGUI, the user can go to the DayGUI, LocationGUI, LocationAlarmViewerGUI, and the SettingsGUI by clicking on their respective buttons. See Figure 3 for a screen shot of the StartGUI.

UpdateLocGUI

The UpdateLocGUI allows the user to view the groups (collection of locations) in the system and the locations that belong to each one. The user has the added capability of editing the contents of a group by adding or removing locations. Once the user has finished editing the groups, he must click on the Save button in order for the changes to take effect. Clicking the Save button saves the

Figure 15: UpdateLocGUI
groups and their contents to the Groups file. The user can add or delete a group by clicking on the Edit Groups button. The groups will be discussed further in section 3.3.9.

### 3.3.2 MeetingList

The MeetingList contains all of the meetings that exist in the system. Every time a reminder is created, a meeting object is created for it and inserted into the list. A meeting object is a means of encapsulating all of the information about a reminder. Such information includes the event’s name, date, location, summary, the notification time, and if the reminder’s alarm is smart.

The MeetingList is sorted chronologically by the dates of the events. When a reminder is edited, its corresponding meeting is deleted and a new meeting object with the updated information is added to the list. As expected, when a reminder is deleted, its related meeting object is removed from the list. The MeetingList provides the means of browsing through all of the meetings in chronological order. It fulfills the requirements of the DayGUI by allowing the user to view the events scheduled for a given day.

As reminders are created, an insertion sort algorithm is used to add meetings to the MeetingList. This decision’s impact on the system is discussed in section 4.1.4.

### 3.3.3 AlarmList

The AlarmList contains the alarms for all of the meetings that exist in the system. When a reminder is created, an alarm for that event is created and inserted into the list. Each alarm object belongs to a specific reminder and is responsible for notifying the user of the event at its notification time. When the application’s time equals an alarm’s notification time, the alarm is triggered. This notification time may vary depending on if the alarm is
smart or not. If the alarm is not smart, it's notification time will be the user-selected time at the creation of the reminder: 10 min, 20 min, ..., 1 day before the time of the event for which the reminder was created.

A smart alarm's notification time is calculated by the system and may change several times before the alarm is triggered. When a smart alarm is created, the system computes the time required to travel from the user's current location to the location of the reminder via the time-to-travel matrix. This travel time is offset from the event time to determine the notification time. Every time the user's location has changed, the AlarmList informs each alarm (in the list) and they recalculate their notification times. This method of updating the state of the system is required for several reasons. For example, if the user moves further away from an event's location, the system should move the notification time of the alarm (for this event) earlier or else it could potentially alert the user too late. A late alarm would not give the user sufficient time to travel to the event and this is not a desired trait of the Smart Reminder.

In another example, suppose a user arrives at an event's location early (before the alarm has triggered). In this occasion, the notification time is recalculated and would be equal to the event time since it takes no time to travel to a location where the user is already present. Given that the user is already at the location of the event, there is no need to alert him of the event.

Like the meeting list, the alarm list is ordered chronologically. The alarms are sorted by their notification dates. When a reminder is edited, its corresponding alarm is deleted and a new alarm object with the updated information is added to the AlarmList. Similar to the MeetingList, when a reminder is deleted, its related alarm is removed from the list.
When an alarm is triggered, an AlertGUI is instantiated with the event’s information. This action displays an alert message to the user and plays an ‘alert sound’ as well. Similar to the MeetingList, an insertion sort algorithm is used to add alarms to the AlarmList as alarms are created. This decision’s impact on the system is discussed in section 4.1.4.

### 3.3.4 LocationAlarmList

The LocationAlarmList contains all of the location alarms that exist in the system. When a location alarm is created, it is inserted into the list. In order to schedule a location alarm, the user must specify the name of the location alarm, a summary, and the location or group where the alarm should trigger. Each location alarm is responsible for reminding the user of something when he is at a certain location or place belonging to a group of locations.

The LocationAlarmList is sorted alphabetically by the names of the location alarms. When an alarm is edited, it is deleted and a new location alarm object with the updated information is added to the list. As expected, when a location alarm is deleted, it is removed from the list. The LocationAlarmList provides the means of listing all of the location alarms in alphabetical order. It fulfills the requirements of the LocationAlarmViewerGUI by allowing the user to view all of the alarms. When the user’s location has changed, the LocationAlarmList checks all of the alarms and triggers the appropriate ones if any exist. As location alarms are created, an insertion sort algorithm is used to add them to the LocationAlarmList. This decision’s impact on the system is discussed in section 4.1.4.
### 3.3.5 Location Service

The location service is a critical component of the system. It is responsible for keeping track of the user’s location. Without it, the system would not adjust properly as the user moves from one location to another. The location service consistently gathers information from all of the location providers and reports the most accurate data to the system. Upon learning of a user’s movement, the service informs the AlarmList and LocationAlarmList of the new location and they process the information accordingly.

The Location Service consists of a thread that checks the Manual Location Provider, Cricket Location Provider, and IP Location Provider for new information about the user’s location. The frequency the thread runs at is determined by the user-defined setting specified in the SettingsGUI.

### 3.3.6 Cricket Location Provider

The Cricket Location Provider (Clp) determines the location of the device by interfacing with the Cricket Location System. As one would expect, the device must be equipped with a Cricket listener and be near a beacon in order for the Clp to actually provide the user’s location. This module interacts with Allen Miu’s code, which gathers data from the listener and calculates the closest Cricket beacon. Once the closest beacon id is known, the Clp looks up the location of the beacon in the CricketTable. The CricketTable contains a list of all the known Cricket beacons and their location.

With the knowledge of the closest beacon’s id, the Clp can determine the device’s location. The user

---

**Figure 16: CricketTable File**

- LR02-32 6FloorLounge1
- LR01-32 6FloorLounge2
- LR6-32 6FloorLounge3
- 600LL-33 6FloorLounge4
- 600LR-32 6FloorLounge5
- 617B-32 6FloorLounge6
- 617-32 6FloorLounge7
- 616-32 6FloorLounge8
- 615-32 6FloorLounge9
can add, delete, and edit the CricketTable as he sees fit through the AllLocGUI. The location service relies heavily on the Clp for obtaining critical location information.

3.3.7 IP Location Provider

The IP Location Provider (Iplp) performs several important tasks to determine the location of the user. Naturally, the provider does not function properly unless the user is communicating on a network. The MIT network is set up in such a way that each IP subnet is assigned to one building. Given the nature of this structure, knowledge of the device’s IP address makes it possible to determine which building (on the MIT campus) the user is in. The Iplp obtains the iPAQ’s IP address by making a system call. Once the IP address is known, there must be a method of determining which location it is mapped to. To facilitate this need, there is a file that contains a mapping of all known IP subnets to their respective locations. The file, named IPTable, allows the system to do a quick table lookup whenever the user’s current location is required. The user can extend the range of the Iplp beyond MIT’s campus by adding more IP subnets and locations to the IPTable. The file can also be edited (via the AllLocGUI) in the event that a location name or IP subnet needs to be changed. The Iplp is an essential component of the application and works hand in hand with the location service.

### Figure 17: IPTable File

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.52.0.0/16</td>
<td>AI-Lab</td>
</tr>
<tr>
<td>128.128.0.0/16</td>
<td>WoodsHole</td>
</tr>
<tr>
<td>192.52.0.0/16</td>
<td>NEARnet</td>
</tr>
<tr>
<td>192.52.64.0/24</td>
<td>Haystack observatory</td>
</tr>
<tr>
<td>18.2.0.0/21</td>
<td>Building N42</td>
</tr>
<tr>
<td>18.2.8.0/21</td>
<td>Building E52</td>
</tr>
<tr>
<td>18.7.0.0/27</td>
<td>Akamai</td>
</tr>
<tr>
<td>18.7.7.0/24</td>
<td>Building 24</td>
</tr>
<tr>
<td>18.10.0.0/16</td>
<td>LCS</td>
</tr>
<tr>
<td>18.11.0.0/16</td>
<td>Building W11</td>
</tr>
<tr>
<td>18.35.0.0/16</td>
<td>SloanTF 56</td>
</tr>
</tbody>
</table>

Masters Thesis June 2002
3.3.8 Manual Location Provider

The manual location provider helps the application continue to function normally in the absence of the other two location providers. There are instances when the system will not be able to determine the user’s location from the IP or Cricket Location Providers; when there is no access to a cricket system or a network. To deal with these situations, the user can manually set the location of the device. Manually setting the location causes the location service to overlook the other location data. In other words, the manual location provider will override the other providers. The system will continue to accept the manual location setting until the user has cleared it. The manual location provider was very helpful in the testing phase of the project.

3.3.9 Groups

A group is simply a collection of locations. Groups serve two purposes in the SmartReminder application. The first is to help make the user’s experience more pleasant. When selecting a location (while scheduling a reminder), rather than scrolling through the long list, the user can narrow down or filter the choices by selecting a group. For example, there may be a group labeled Work, which contains a list of locations related to the user’s work. The second purpose groups serve is related to location alarms. When creating a location alarm, there may be more than one location where the user wants the alarm to trigger. Rather than creating identical alarms for each of these locations, the user can specify a group of locations where the user should be reminded. The user can create, delete, and
edit the locations in a group in the UpdateLocGUI. The groups and the locations in them are saved to the Groups file so that they are not lost when the application is closed or the iPAQ is off.

3.3.10 Time-To-Travel Matrix

The Time-To-Travel Matrix is a key module in the application. It is required in order for the smart alarms to trigger at the appropriate time. Whenever the user’s location has changed, the notification time of a smart alarm must be updated. The new notification time depends on how much time is needed to travel from the device’s current location to the location of the event. The matrix provides the walking travel time between any two locations that the system knows about. Unfortunately, the travel times (of the matrix) used in the project were randomly generated. Though these times were not correct, they were needed to demonstrate the functionality of the Smart Reminder. Implementing an accurate Time-To-Travel Matrix is beyond the scope of the project, but would be a very useful extension.

4 Evaluation of System

4.1 Design Alternatives

The design and implementation of the Smart Reminder application was finalized after many modifications to the initial layout. In many cases, problems were overcome by choosing either a more straightforward method or one that was not considered in the design phase of the project. This section describes some of the situations where there was more than one solution/implementation.
4.1.1 Java VS C

The first question had to be answered before any design or implementation could take place. It was not apparent what language the application should be developed in. There were many options that were eventually narrowed down to Java and C. Although both languages have their pros and cons, they were preferred because they were supported under the OS running on the iPAQ. After some evaluation, Java was chosen as the language to develop the Smart Reminder application. Java was more appealing because it would be easier to interface with Allen Miu's Cricket Location System code, which was also written in Java. In addition, the developers of the application were more proficient in Java than C. All development was compliant with JDK 1.3, which was the latest version that could be used with the iPAQ's StrongARM processor. Developing in C would have resulted in a faster application that is less taxing on the device's processor. However, a lot more time and effort would be required to interact with the Cricket Location System's code.

4.1.2 Linux

A significant yet straightforward decision was to use Linux as the operating system on the iPAQ. There were several reasons why this choice was made. Using an open source platform made it possible to modify the operating system (if the need arose) and make all research available to other developers. It also allowed the thesis to tie in nicely to the progress being made in MIT's Project Oxygen and at Compaq's Cambridge Research Labs. The project used the handheld group's Familiar distribution, which is based on the Debian version of Linux. This distribution possessed many features that were required for the successful completion of the project. Some of these features include a variety of
networking options (IPv6, Mobile IP, etc), a JVM, X-windows, and the ability to use multiple file systems.

Since the handhelds group supported the Familiar version, the site (handhelds.org) as well as the handhelds users were very supportive and proved to be valuable resources.

4.1.3 Location Service

The entire application was based on the location discovery and its implementation. As expected, there were a number a ways that the location service could interact with the location providers. On one hand, the location providers could push data to the location service. In this implementation, each provider would have its own thread that runs, collects information about the device's location, and updates the location service when the location has changed. The location service would consistently be listening for information from the providers. This implementation was quickly discarded due to its inefficiency. There were simply too many threads in this design and this would be an arduous task for the processor since the threads are collecting and comparing data simultaneously.

The preferred implementation involved the location service pulling information from the location providers. In this design, the location service has a thread, which collects (pulls) location data from the providers. The location service gathers the data and determines if
the user has moved or not. This option proved to be more resourceful because the location service processes all of the information rather than each of the providers. It would also be easier to add more location providers in this centralized system. As mentioned in section 3.3.5, the user specifies (in the SettingsGUI) the frequency with which the location service collects information from the providers.

4.1.4 List Sorting

Sections 3.3.2-3.3.4 described the three lists, which hold meetings, alarms, and location alarms. The lists are sorted either chronologically (by meeting date or alarm notification date) or alphabetically (by location alarm name). As a result, some form of sorting algorithm has to be used to ensure that items in a list remain in the correct order. Rather than add an item to some place and resort the list, the item is inserted into its proper place in the list so that the right ordering is maintained. In other words, an insertion sort algorithm is used to sort the items in the lists. When an item's details are edited (edit a reminder date, etc), the item is deleted from the list and the updated one is then inserted. However, if multiple items need to be edited such as when the alarms recalculate their notification times when the device changes locations, the items of the list are inserted into a newly constructed list. Though this decision seems to waste resources, it proves to be not as detrimental as one would assume. In its worst case, insertion sort, runs for $\theta(n^2)$ time. Since the application only deals with small data sets, the algorithm runs pretty quickly. Though this implementation is sufficient, if the system is expanded, a more optimal algorithm will be required as the size of the lists grow. One alternative design is a merge sort which will maintain the lists' order in $\theta(n \log n)$ time.
4.1.5 System Files

A desired feature of Smart Reminder is the ability to save state when the application is closed (or the device is off) so that the user can resume using the application from where he left off. Taking advantage of Java’s serializable interface was necessary for saving the MeetingList, AlarmList, and LocationAlarmList to disk whenever the user exited the application. Upon opening Smart Reminder, the system read the files (AlarmList.dat, MeetingList.dat and LocationAlarmList.dat) to bring the program to the last known stable state. The alternative method considered to achieve this goal was to save the lists as strings and store them in a file. In this design, the files would be parsed and reloaded when the user opened the application. This design was not used because it would be inefficient when the length of the lists grew to a large number. It would simply utilize too many system resources to parse through a lengthy file. Using the first method proved to be the most appropriate implementation of saving the lists to maintain the application’s state.

Saving the application’s state is not limited to the three lists. It is also imperative that the user-defined settings are maintained when the device has been turned off. This requirement is fulfilled with the Settings file, in which each system variable value is listed as a string row by row. Since there are only four variables, parsing this file and loading the settings at startup is a rather simple task. Using the serializable interface was not as beneficial in this case for two reasons. As mentioned earlier, the data set is small and can easily be extracted from a file. The second reason is that there is no class that encapsulates the user preferences and this renders the serializable interface useless for this implementation.
As described in sections 3.3.6 and 3.3.7, the location providers are useless unless they are able to map the data they obtain to a location name. The CricketTable and IPTable contain the mappings from Cricket Beacon ID and IP subnet key to a location. The data in these files are stored as strings in two columns. One column has the Beacon ID (in the CricketTable) or IP subnet key (in the IPTable) and the other column has the corresponding locations that the first column entries map to (see Figures 16 & 17). Due to the structure of the file, each Beacon ID or IP subnet key is mapped to at most one location. The user has the added capability of editing the information in either column via the AllLocGUI. Once again, there is no need for the serializable interface since there is no class that encapsulates the mappings from Beacon ID to location and IP subnet key to location.

The groups, which are described in section 3.3.9, are stored in the Groups file. Groups are basically a collection of locations. Each group has a unique user-defined name and a number of locations. The Groups file is organized in a manner such that each line has a Group followed by the locations that are a member of that group delimited by colons (See Figure 18). Each group and its locations take up no more than one line in the file to make the job of parsing the file easier. The user may edit the members of a group in the UpdateLocGUI (see Figure 15). There was an alternative design considered for capturing the groups and the locations in them. Rather than saving all of the information to one file, it is possible to have a Group directory with each file in it named according to the group that it represents. Each file would contain a list of the locations that are members of the group. Though the structure of this implementation appears more organized, it unfortunately makes the groups harder to maintain. For example, if the user deletes a
location from the system, each file in the Group directory has to be opened in order to
delete the location if it exists. The overhead included in making such minor changes is
just too much, which makes this implementation less appealing than the other one.

4.2 Test Results

All of the application’s functionality was thoroughly tested. This section is intended to
highlight the major problems encountered during the test/debug phase as well as the
alterations were made to overcome them.

The location providers are the core of the application and were, therefore, tested first.
Fortunately, the IP location provider worked after the initial implementation and a few
minor adjustments. It was tested in various buildings on the MIT campus and provided
accurate location information as supplied by the IPTable. Due to previous experiences
with the cricket location system (explained in the next section), the implementation and
testing of the cricket location provider went smoothly. Since the cricket location system
is deployed in MIT’s LCS, this was the setting for the functionality testing.

Once the location providers’ proper operation was verified, the next module to test was
the location service. Interfacing the location service with the location providers required
minimal effort, but would require a minor adjustment when additional components were
introduced to the system. This will be discussed later in the section.

Testing the ability to scheduling reminders was a two-step process. The first and simple
step was to make sure the user could schedule an event with a regular alarm. The second,
more involved, step was to schedule reminders with smart alarms. A minor change to the
MeetingGUI was made soon after step one commenced. In order to give the user more
options and reduce the waiting time during the testing phase, more options were added to
the notification time. Before the modification, the options were 15 minutes, 30 minutes, 1 hour, and 1 day before the event time. No other changes were required because the alarms of the reminders triggered appropriately. Step two required much more effort and patience. Testing the smart alarms required interaction with the time-to-travel matrix, which fortunately worked as expected. The major revision resulting from this phase was made to the thread, which checks the alarms and triggers them when their notification times arrive. At first, this thread woke up periodically to see if an alarm needed to be triggered. After the revision, the thread now wakes up at the time of the next alarm. This modification reduces the unnecessary use of system resources.

The final step was to test the scheduling of location alarms. A significant change had to be made to the location service to fix a major bug concerning location alarm. It was quickly noticed that not all of the location alarms were triggering correctly. This problem was occurring because of a bug in the location service. As the location service gathered information from the providers, it only notified the system of the most accurate data. This implementation led to problems because it was possible for a less accurate location alarm to exist. For example, suppose the user creates two location alarms with one for a room in LCS and one for LCS itself. When the user is in the room specified by the first alarm and the location service queries the providers, the room would be returned by the Cricket location provider and LCS would be returned by the IP provider. Since the location service only pays attention to the Cricket provider in this case, the second alarm does not trigger. This bug was fixed by making the location service report all of the data (it gathers from the providers) to the LocationAlarmList.
After the test phase was completed, the system fulfilled all of the requirements and met all of the expectations.

4.3 Experiences With Location System

An opportunity to experiment with the Cricket Location Technology arose in a project for a Pervasive Computing class. The project's objective was to consistently track the position of 4 to 8 users as they participated in a game. The users played the game on the same device (iPAQ) that the Smart Reminder is intended to run on. The Cricket Location System was used to track location. This decision proved to be a great way to become familiar with the system before including it in the thesis.

At first, the project, named Capture the Flag (CTF), took advantage of Allen Miu's interface to the Cricket System code. Unfortunately, obtaining accurate latitude/longitude/altitude values for a user was difficult with the system's implementation. The device's undesirable response time suggested that the system was computationally intensive. In addition, the game required more precise location values than the system could provide. Both altering the code and explaining the issues to Allen dealt with these problems. Allen helped out by providing another version of the system, which was more precise, but unfortunately still too intensive.

It was later determined that the hardware was the source of many problems. Testing the project revealed unpredictable and unreliable behavior by the hardware.

Working on the CTF project shed light on several system limitations, which presented themselves when precise latitude/longitude/altitude were required.
5 Related Work

5.1 NetGeo – The Internet Geographic Database

NetGeo is an endeavor of CAIDA, the Cooperative Association for Internet Data Analysis. CAIDA’s goal is to support the engineering and maintenance of the Internet by making resources such as tools and analyses available. NetGeo uses a database and an assortment of Perl scripts to map domain names, IP addresses, and Autonomous System numbers to their geographical locations. The main motivation for the group’s effort is to enhance network visualization tools.

The system’s functionality is simple and straightforward. NetGeo’s database is composed of tables, which have fields for location names, city, state, and country or US zip codes. Each table corresponds to a domain name and a set of latitude/longitude values. When the system searches for info about a domain name, it first consults its database. If the table is present, the latitude/longitude values are returned immediately. On the other hand, if the domain name’s table doesn’t exist, NetGeo will perform a series of whois lookups, parse the location information from their return, and store all of the relevant data into a table in the database. The proper table will now exist the next time location information is requested for the domain name in question. As the system’s use increases, the database will grow and become populated with more location information. This is very similar to the IP location system/provider used in the project. The sponsors for the Internet geographic database include APNIC, ARIN, GETTY, NSFNSI, and RIPE NCC.
6 Conclusions

6.1 Expansion

Although the project met all of the specifications, there are several areas where enhancements/expansions to the system can be made. This section will discuss some enhancements and expansions that could improve the functionality of the system as well as the user's experience. Some of the expansions were in the original design, but were not implemented due to timing constraints.

Making the system compatible with more devices would make the application appealing to a bigger audience. More specifically, it would be helpful if the Smart Reminder worked under the Windows platform. This enhancement would allow the application to work on a wider range of devices. Porting the system to the Windows OS would require some changes to the current code. The biggest change would be made to the IPLocationProvider. The application would need to know what platform is running on the device so it can execute the proper system command to obtain the IP address and subnet mask. The code would also need to be compiled appropriately for the specific operating system running on the device. The debugging and testing phase of the project would have been a lot easier if the system was compatible with the Windows OS.

Naturally, adding more location providers would result in a more robust application. In the original design of the system, the application was also supposed to interface with GPS. In the implementation, however, the GPS interface was excluded for several factors that will not be discussed. To incorporate GPS, code would need to be added to the Location Service and a few lines would need to be added to other various places. Adding GPS would make the location retrieval mechanism a lot more powerful. The biggest
advantage of using GPS is that the system would be able to work outdoors. The addition of this feature would result in an application that works at any location regardless of what resources are available.

As one may notice, the system is not completely precise in its calculations. This is due to the absence of an accurate time-to-travel matrix. A much-needed enhancement to the system is an interface that would allow the user to view and edit the times in the matrix. Such an interface would result in a system where all of the smart alarms trigger at the exact time the user should depart for an event. Allowing for multiple methods of transportation is a useful expansion to the matrix. Since the user may not be traveling by foot, it is necessary to have travel times for other modes of transportation including by car, by bike, by bus, etc.

Many people who have used the applications expressed an interest in interacting with other users. Some added functionality includes finding the user’s buddies, finding all buddies within a certain range from the user, finding all users located in a certain group of locations, chat capability, etc. This enhancement would require the user to compromise his privacy when the system is centralized. To implement these features, the devices would have to regularly update a server with their location information. Adding this feature would make the application more appealing while taking away the user’s privacy, which is very important to many potential users.

6.2 Final Thoughts

Many implementations of the Smart Reminder were reviewed and compared in the designing phase of the project. In the end, a user-friendly implementation that minimized system overhead was chosen. The overall project achieves all of the goals of the system
and meets all of the requirements. The system makes use of the available resources, the
two most important being the wireless network (set up on the MIT campus) and the
Cricket Location System (deployed at MIT’s LCS). The result is a useful application that
makes the user’s life easier. The fact that its behavior can change depending on the
device’s location makes it attractive to people who maintain a busy schedule. As
mentioned earlier, the application may be combined with GPS so that its range of
functionality can expand to the outdoors. All things considered, the project has succeeded
in showing that pervasive computing will be prominent in the future. It is just one
example of how the interaction of computers and human beings can be improved.
Working on this project has been a very educational and eye-opening experience.
Hopefully, the Smart Reminder will be useful to many and contribute to both the goals of
Project Oxygen and the Handhelds group.

7 Acknowledgement

The author wishes to express his sincere thanks to Anant Agarwal, John Ankcorn, Steve
Garland, Jamey Hicks, Allen Miu, Craig Music, Larry Rudolph, Ken Steele, and Gita
Sukthankar for valuable discussions, encouragement and comments

8 Resources

[1] Cormen; Thomas, Leiserson; Charles, Rivest; Ronald. Introduction to Algorithms.


http://handhelds.org/Compaq/iPAQH3600/iPAQ_H3600.html (March 02, 2002).


9 Appendix

9.1 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");
    }
}

9.2 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.
 *
 */
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.updateState called");
AlarmList.resort();
SR.updateTimeToNextAlarm();
}
/*
 * Used internally to resort elements within the data structure
 */
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 = CurrentDateGetCurrentCalendar();
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 something");
insert(create);
e("\n\njust inserted alarm, the same 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 = Calendar.getInstance();
    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 purposes.

```java
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

```java
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.

```java
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(alarmCalendar.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);
    }
}
}

9.3 AlertGUI.java

/**
 * AlertGUI 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) + 
        + 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) +\n        +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);
    }
}

9.4 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);
```java
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.CRICKET_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.LISTLOC);
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 == loc) {
            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);
        locO.setVisible(true);
        loc1.setVisible(true);
        loc0.setVisible(true);
        data0.setVisible(true);
        data1.setVisible(true);
        beaconID_L.setVisible(true);
        key_L.setVisible(true);
        key_L.setVisible(false);
        beaconID_L.setVisible(false);
    }
}
loc.setVisible(true);
loc.setVisible(true);
data0.setVisible(true);
data1.setVisible(true);
SaveBtn.setVisible(true);
CancelBtn.setVisible(true);
RemoveBtn.setVisible(false);
EditBtn.setVisible(false);
NewBtn.setVisible(false);
}

void NewBtn_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);
    SaveBtn.setVisible(true);
    CancelBtn.setVisible(true);
    RemoveBtn.setVisible(false);
    EditBtn.setVisible(false);
    NewBtn.setVisible(false);
}

void RemoveBtn_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 SaveBtn_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);
}

9.5 CricketLocProvider

/**
 * 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.*;

Masters Thesis June 2002
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;
}
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
    ********************
    
    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())!=-l)
        { 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 inputted 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_BEACON_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);
    }
  }
}

9.6 CurrentDate

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

9.7 DayGUI.java

/**
 * Class used as one of the GUI's of the system. It displays
 * a schedule 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 & Marc Bourget
 */
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 TEXTHEIGHT = 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 = "";

    Masters Thesis June 2002
    Masters Thesis June 2002

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()
{
    today = CurrentDate.getCurrentCalendar();// set the current date
    this.day = parseCalendar(today); //get back here

    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(0,90,73,18);
    TenButton.setText("10:00");
    TenButton.setActionCommand("10:00");
    getContentPane().add(TenButton);
    TenButton.setBounds(0,151,73,18);
    ElevenButton.setText("11:00");
    ElevenButton.setActionCommand("11:00");
    getContentPane().add(ElevenButton);
    ElevenButton.setBounds(0,212,73,18);
    TwelveButton.setText("12:00");
    TwelveButton.setActionCommand("12:00");
    getContentPane().add(TwelveButton);
    TwelveButton.setBounds(0,273,73,18);
    ThirteenButton.setText("1:00");
    ThirteenButton.setActionCommand("1:00");
    getContentPane().add(ThirteenButton);
    ThirteenButton.setBounds(0,334,73,18);
    FourteenButton.setText("2:00");
    FourteenButton.setActionCommand("2:00");
    getContentPane().add(FourteenButton);
    FourteenButton.setBounds(0,395,73,18);
    FifteenButton.setText("3:00");
    FifteenButton.setActionCommand("3:00");
    getContentPane().add(FifteenButton);
    FifteenButton.setBounds(0,456,73,18);
    SixteenButton.setText("4:00");
    SixteenButton.setActionCommand("4:00");
    getContentPane().add(SixteenButton);
    SixteenButton.setBounds(0,517,73,18);
    SeventeenButton.setText("5:00");
    SeventeenButton.setActionCommand("5:00");
    getContentPane().add(SeventeenButton);
    SeventeenButton.setBounds(0,578,73,18);
    EighteenButton.setText("6:00");
    EighteenButton.setActionCommand("6:00");
    getContentPane().add(EighteenButton);
    EighteenButton.setBounds(0,639,73,18);
    NineteenButton.setText("7:00");
    NineteenButton.setActionCommand("7:00");
    getContentPane().add(NineteenButton);
    NineteenButton.setBounds(0,700,73,18);
    TwentyButton.setText("8:00");
    TwentyButton.setActionCommand("8:00");
    getContentPane().add(TwentyButton);
    TwentyButton.setBounds(0,761,73,18);
    TwentyOneButton.setText("9:00");
    TwentyOneButton.setActionCommand("9:00");
    getContentPane().add(TwentyOneButton);
    TwentyOneButton.setBounds(0,822,73,18);
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.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.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("18:00");
getContentPane().add(EighteenButton);
EighteenButton.setBounds(0,219,73,18);
EighteenButton.setFont(ff);
NineteenButton.setText("19:00");
NineteenButton.setActionCommand("19:00");
getContentPane().add(NineteenButton);
NineteenButton.setBounds(0,237,73,18);
NineteenButton.setFont(ff);
TwentyButton.setText("20:00");
TwentyButton.setActionCommand("20:00");
getContentPane().add(TwentyButton);
TwentyButton.setBounds(0,255,73,18);
TwentyButton.setFont(ff);
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(TEXTHORSTART,TEXTVERSTART+0*TEXTHEIGHT,TEXTWIDTH,TEXTHEIGHT);
NineText.setText(nine);
NineText.setDisabledTextColor(java.awt.Color.black);
NineText.setEditable(false);
getContentPane().add(NineText);
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.setBounds(TEXT_HORSTART,TEXTVERSTART+2*TEXTHEIGHT,TEXTWIDTH,TEXTHEIGHT);
ElevenText.setDisabledTextColor(java.awt.Color.black);
ElevenText.setEditable(false);
getContentPane().add(ElevenText);
ElevenText.setText(eleven);
ElevenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 2 * TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
TwelveText.setDisabledTextColor(java.awt.Color.black);
TwelveText.setEditable(false);
getContentPane().add(TwelveText);
TwelveText.setBounds(TEXT_HORSTART,TEXT_VER_START + 3 * TEXT_HEIGHT,TEXTWIDTH,TEXTHEIGHT);
ThirteenText.setDisabledTextColor(java.awt.Color.black);
ThirteenText.setEditable(false);
getContentPane().add(ThirteenText);
ThirteenText.setText(thirteen);
ThirteenText.setBounds(TEXT_HOR_START,TEXT_VER_START + 4 * TEXT_HEIGHT,TEXT_WIDTH,TEXT_HEIGHT);
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));

FifteenText.setDisabledTextColor(java.awt.Color.black);
FifteenText.setEditable(false);
getContentPane().add(FifteenText);
FifteenText.setText(fifteen);
FifteenText.setBackground(new java.awt.Color(204,204,204));

SixteenText.setDisabledTextColor(java.awt.Color.black);
SixteenText.setEditable(false);
getContentPane().add(SixteenText);
SixteenText.setText(sixteen);
SixteenText.setBackground(new java.awt.Color(204,204,204));

SeventeenText.setDisabledTextColor(java.awt.Color.black);
SeventeenText.setEditable(false);
getContentPane().add(SeventeenText);
SeventeenText.setText(seventeen);
SeventeenText.setBackground(new java.awt.Color(204,204,204));

EighteenText.setDisabledTextColor(java.awt.Color.black);
EighteenText.setEditable(false);
getContentPane().add(EighteenText);
EighteenText.setText(eighteen);
EighteenText.setBackground(new java.awt.Color(204,204,204));

NineteenText.setDisabledTextColor(java.awt.Color.black);
NineteenText.setEditable(false);
getContentPane().add(NineteenText);
NineteenText.setText(nineteen);
NineteenText.setBackground(new java.awt.Color(204,204,204));

TwentyText.setDisabledTextColor(java.awt.Color.black);
TwentyText.setEditable(false);
getContentPane().add(TwentyText);
TwentyText.setText(twenty);
TwentyText.setBackground(new java.awt.Color(204,204,204));

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);
TwentyButton.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 JFrame and makes it visible.
 */
static public void main(String[] args)
{
  try {
    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()
{
  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();
boolean frameSizeAdjusted = false;

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();

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

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);

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 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(locJList);
    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(loclist);
    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());
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) );

} else 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");
return( "December" );
else return( "error" );
}

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

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[] 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.LISTLOC);
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.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();

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 */
        Meeting create = new Meeting(thisDay, mName.getText(),
            (String) loc_list.getSelectedItem(), summary.getText(),
            not.getSelectedItem(), 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
    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);
}

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(copypDelete); // should delete
        instead of display a copy of the original
        Meeting create = new Meeting(thisDay, mName.getText(),
        (String)loc_list.getSelectedItem(), summary.getText(),
        not.getSelectedItem(), smart); // Create new meeting
        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)
    {
        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 == TwentyButton) {  
    TwentyButton.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));  
    EightText.setText(eight);  
    EightText.setDisabledTextColor(java.awt.Color.black);  
    EightText.setEditable(false);  
    getContentPane().add(EightText);  
    EightText.setBackground(new java.awt.Color(204,204,204));  
    NineText.setText(nine);  
    NineText.setDisabledTextColor(java.awt.Color.black);  
    NineText.setEditable(false);  
    getContentPane().add(NineText);  
    NineText.setBackground(new java.awt.Color(204,204,204));  
    TenText.setText(ten);  
    TenText.setDisabledTextColor(java.awt.Color.black);  
    TenText.setEditable(false);  
    getContentPane().add(TenText);  
    TenText.setBackground(new java.awt.Color(204,204,204));  
    ElevenText.setText(eleven);  
    ElevenText.setDisabledTextColor(java.awt.Color.black);  
    ElevenText.setEditable(false);  
    getContentPane().add(ElevenText);  
    ElevenText.setBackground(new java.awt.Color(204,204,204));  
    TwelveText.setText(twelve);  
    TwelveText.setDisabledTextColor(java.awt.Color.black);  
    TwelveText.setEditable(false);  
    getContentPane().add(TwelveText);  
    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));

FourteenText.setDisabledTextColor(java.awt.Color.black);
FourteenText.setEditable(false);
getContentPane().add(FourteenText);
FourteenText.setText(fourteen);
FourteenText.setBackground(new java.awt.Color(204,204,204));

FifteenText.setDisabledTextColor(java.awt.Color.black);
FifteenText.setEditable(false);
getContentPane().add(FifteenText);
FifteenText.setText(fifteen);
FifteenText.setBackground(new java.awt.Color(204,204,204));

SixteenText.setDisabledTextColor(java.awt.Color.black);
SixteenText.setEditable(false);
getContentPane().add(SixteenText);
SixteenText.setText(sixteen);
SixteenText.setBackground(new java.awt.Color(204,204,204));

SeventeenText.setDisabledTextColor(java.awt.Color.black);
SeventeenText.setEditable(false);
getContentPane().add(SeventeenText);
SeventeenText.setText(seventeen);
SeventeenText.setBackground(new java.awt.Color(204,204,204));

EighteenText.setDisabledTextColor(java.awt.Color.black);
EighteenText.setEditable(false);
getContentPane().add(EighteenText);
EighteenText.setText(eighteen);
EighteenText.setBackground(new java.awt.Color(204,204,204));

NineteenText.setDisabledTextColor(java.awt.Color.black);
NineteenText.setEditable(false);
getContentPane().add(NineteenText);
NineteenText.setText("asdfasdfasdf");
NineteenText.setBackground(new java.awt.Color(204,204,204));

TwentyText.setDisabledTextColor(java.awt.Color.black);
TwentyText.setEditable(false);
getContentPane().add(TwentyText);
TwentyText.setText(twenty);
TwentyText.setBackground(new java.awt.Color(204,204,204));

TwentyOneText.setDisabledTextColor(java.awt.Color.black);
TwentyOneText.setEditable(false);
getContentPane().add(TwentyOneText);
TwentyOneText.setText(twentyone);
TwentyOneText.setBackground(new java.awt.Color(204,204,204));
** 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 giving 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 giving 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 giving 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 giving 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)) {

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 giving 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 giving the meeting ui " + now.get(now.HOUR_OF_DAY));
    }
}
```java
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 giving 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 giving 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 giving 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 giving 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 giving 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);
        System.out.print("this is the hour of the gregorian calendar we are giving the meeting ui ");
        m = MeetingList.search(now);
    }
}

9.8 DistanceToTravel.java

/**
 * this is a matrix implemented as a Vector with all the Locations
 * from Location.LOCLIST
 * 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();
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;
        }
    }
}

public static long getTimeToTravel(String departure, String arrival) {
    int departIndex = locs.indexOf(departure);
    int arriveIndex = locs.indexOf(arrival);
    long result = distances[departIndex][arriveIndex];
    return result;
}

public static long getTimeToTravel(int departure, int arrival) {
    long result = distances[departure][arrival];
    return result;
}

9.9 ErrorGUI.java

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

package SmartReminder;

import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.xml.*;
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);

        SymMouse aSymMouse = new SymMouse();
        OkB.addMouseMotionListener(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() {
        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 ErrorL = new javax.swing.JLabel("ERROR!");
javax.swing.JLabel MessageL = new javax.swing.JLabel();
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);
    }
}

9.10 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);
    }
}

9.11 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.*;

Masters Thesis June 2002 102
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 = "i"l;
        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, " ");
            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.LOCALL.get(Location.IPALL.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};
** 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 bytel = st.nextToken();
    String byte2 = st.nextToken();
    String byte3 = st.nextToken();
    int end;
    if (bytel.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, ".");
    bytel = st.nextToken();
    byte2 = st.nextToken();
    byte3 = st.nextToken();
    String key = "";
    String temp[] = { "unknown", "unknown"};
    while (end > 8) {
        if (end > 16) {
            key = bytel + "." + byte2 + "." + byte3 + "." + "0/* + end;
        } else key = bytel + "." + byte2 + ".0.0/* + end;
        String locName = (String)getLoc(key);
        if (locName.equals("unknown") ) {
            temp[0] = key;
        }
temp[1] = locName;
break;
}
ed--;
}
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();
        init(); //after saving the new file, we should reinitialize
    } 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
" +
            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
" +
                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);
    }
9.12 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() {
    }
}

9.13 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 query 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;
}
9.14 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);
nam_setwEditable(true);
nam.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);

groupList.setText("Group:");
getContentPane().add(groupList);
groupl.setooeaks(15, 65, 130, 15);
groupl.setFont(f);
groupList.setVisible(true);

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

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

loc1.setSelected(false);
loc1.setBounds(70, 85, 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);

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 whether loc1 or loc 2 is selected
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[])
{
}

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;

    // 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); 
}

boolean frameSizeAdjusted = false;
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 groupList = new javax.swing.JLabel();
javax.swing.JLabel nameL = new javax.swing.JLabel();
javax.swing.JLabel header = new javax.swing.JLabel();
javax.swing.JTextField message = new javax.swing.JTextField();
javax.swing.JLabel messageLabel = new javax.swing.JLabel();
javax.swing.JTextField name = new javax.swing.JTextField();
javax.swing.JRadioButton locO = new javax.swing.JRadioButton();
javax.swing.JRadioButton locl = new javax.swing.JRadioButton();
javax.swing.JRadioButton loc2 = new javax.swing.JRadioButton();
javax.swing.JLabel MessageL = new javax.swing.JLabel();
javax.swing.JButton AddBtn = 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)
    {

```java
JComboBox cb = (JComboBox)e.getSource();
updateDisplay();
}
}

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);
}
}

9.15 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 whereas the second vector 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
 * existence 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 there 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));
        }
static public boolean read()
{
    locationAlarms = new Vector();
    locationAlarmLocations = new Vector();
    boolean result = false;
    // unserializing 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;
}

/**
 * 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);
        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;
}
**LocationAlarmViewerGUI.java**

```java
/**
 * LocationAlarmViewerGUI 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);
    }

    private Vector createHeaders() {
        return null;
    }

    private Vector createData() {
        return null;
    }

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

---

9.16 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

```java
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);
    }

    private Vector createHeaders() {
        return null;
    }

    private Vector createData() {
        return null;
    }

    static public Vector getLocationAlarms() {
        return locationAlarms;
    }
}
```
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.addMouseListerner(aSymMouse);
Remove_Btn.addMouseListerner(aSymMouse);
Edit_Btn.addMouseListerner(aSymMouse);
Close_Btn.addMouseListerner(aSymMouse);
Update_Btn.addMouseListerner(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 location 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;

Masters Thesis June 2002
123
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 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)

    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);
    (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) {
}
9.17 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, inset.top + inset.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);
9.18 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);
    }
}

June 2002

Masters Thesis
message.setBackground(Color.black);

/**
 * IP Fields
 */
getContentPane().add(ip_L);
ip_L.setBounds(15,35,50,15);
ip_L.setFont(f);
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);
mask.setFont(f);
getContentPane().add(key_L);
key_L.setBounds(15,85,50,15);
key_L.setFont(f);
key.setBounds(85,85,130,15);
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);
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.addMouseListner(aSymMouse);
View_Btn.addMouseListner(aSymMouse);
Close_Btn.addMouseListner(aSymMouse);
Update_Btn.addMouseListner(aSymMouse);
Clear_Btn.addMouseListner(aSymMouse);
Set_Btn.addMouseListner(aSymMouse);
Loc_Btn.addMouseListner(aSymMouse);
loc0.addMouseListner(aSymMouse);
loc1.addMouseListner(aSymMouse);
locP.addActionListner(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 \n from 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 \n from the drop-down list and \nclick Add to add this location");
            loc.setEditable(true);
        } else {
            message.setText("Your location = \n" + loc.getText() );
        }
    }
} 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 and click Set or clear the location by clicking on Clear");
  } else {
    message.setText("Manual location set to
    "+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 acquires 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
 */
StringTokenizer 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.JTextField beaconID = new javax.swing.JTextField();
javax.swing.JTextField beaconLoc = new javax.swing.JTextField();
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 locO = new javax.swing.JRadioButton();
javax.swing JRadioButton locI = new javax.swing.JRadioButton();
javax.swing JLabel MessageL = new javax.swing.JLabel();
javax.swing JButton Add Btn = new javax.swing.JButton("Add");
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 BtnMouseClicked(event);
        if (object == View Btn)
            View BtnMouseClicked(event);
        if (object == Close Btn)
            Close BtnMouseClicked(event);
        if (object == Update Btn)
            Update BtnMouseClicked(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);
        }
    }
}
SetBtn.setVisible(false);
updateDisplay("Wireless");
}
}

void AddBtnMouseClicked(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)locjlist.getSelectedItem());
    message.setText("Location added successfully!");
    AddBtn.setVisible(false);
    locO.setVisible(false);
    loc.setVisible(false);
    loc_list.setVisible(false);
    LocBtn.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)locjlist.getSelectedItem());
    message.setText("Location added successfully!");
    AddBtn.setVisible(false);
    locO.setVisible(false);
    loc.setVisible(false);
    loc_list.setVisible(false);
    LocBtn.setVisible(true);
    loc.setEditable(false);
    }
    }
}
9.19 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 = IPIlocProvider.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) {
}
}

9.20 Meeting.java

/**
 * Meeting stores information about a reminder/event
 * @author Marc Bourget
 */
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 for regular alarm, Meetings create these alarms
 */
 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 for meeting and add it to AlarmList
 */
 GregorianCalendar alert = copyDate( meetingDate );
 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);
 }

 public boolean isSmart() { return( smart ); }

 public int getNotification() {
 return( notification ); }

 public String getSummary() {
 return( summary ); }

 public String getLocation() {
```java
public String getName() {
    return( meetingName );
}

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 );
}
}

9.21 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;
        }
        index++;
    }
}
```
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);
}

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 mol = a.get(a.MONTH);
    int mo2 = b.get(b.MONTH);
    int dtl = 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 " + mol + mo2 + "
date " +dtl + dt2 + " hour " +hr1 + hr2);
    if(year1==year2 &mol==mo2 & dtl==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 date from
    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
date from search meeting list " + t.getMeetingDate());
        return t;
    }
}

// did not find meeting that matches date
Meeting in2 = invalid();
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;
    // unserialze 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);
    }
}

9.22 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();
        }
    }
}

9.23 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);
        updateTimeL.setBounds(15,60,160,15);
        getContentPane().add(updateTime);
        updateTime.setFont(f);
        updateTime.setSelectedItem(Long.toString(SR.defaultUpdate/1000));

        getContentPane().add(secL);
        secL.setBounds(205,60,30,15);
        getContentPane().add(waitTimeL);
        waitTimeL.setBounds(15,85,95,15);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitTime);
        getContentPane().add(waitT
getContentPane().add(secl_L);
secl_L.setBounds(170, 85, 30, 15);
secl_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.addMouseLister(aSymMouse);
Close_Btn.addMouseLister(aSymMouse);
Restore_Btn.addMouseLister(aSymMouse);
loc0.addMouseLister(aSymMouse);
loc1.addMouseLister(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())!=-l) {
            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.LISTLOC);
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");
}

9.24 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);
    Thread locServ = new Thread(new LocationService()); // create
    LocationService thread
    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 );
}

9.25 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.addMouseMotionListener(aSymMouse);
        DayG.addMouseMotionListener(aSymMouse);
        LocAG.addMouseMotionListener(aSymMouse);
        Pref.addMouseMotionListener(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);
        }
    }
}

9.26 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);

groups = new JList();
groups.addPathToAncestor(expander);
groups.setFont(f);
groups.setBounds(10,125,100,15);
groups.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

all_loc = new JList();
all_loc.addPathToAncestor(expander);
all_loc.setFont(f);
all_loc.setBounds(10,25,130,85);
all_loc.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
all_loc.scrollPane.setBoundX(10,25,130,85);
all_loc.scrollPane.setFont(f);
all_loc.scrollPane.setViewport().setView(all_loc);

all_loc = new JList();
all_loc.addPathToAncestor(expander);
all_loc.setFont(f);
all_loc.setBounds(10,145,130,85);
all_loc.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
all_loc.scrollPane.setBounds(10,145,130,85);
all_loc.scrollPane.setFont(f);
all_loc.scrollPane.setViewport().setView(all_loc);

all_loc = new JList();
all_loc.addPathToAncestor(expander);
all_loc.setFont(f);
all_loc.setBounds(10,195,60,15);
gName = new JList();
gName.setFont(f);
gName.setBounds(75,195,140,15);
gName.setVisible(false);

Add_Btn = new JButton("Add");
Add_Btn.setBounds(150,85,70,15);
Add_Btn.setFont(f);

Remove_Btn = new JButton("Remove");
Remove_Btn.setBounds(145,145,80,15);
Remove_Btn.setFont(f);

Save_Btn = new JButton("Save");
Save_Btn.setBounds(10,250,75,15);
Save_Btn.setFont(f);

Edit_Btn = new JButton("Edit");
Edit_Btn.setBounds(130,250,100,15);
Edit_Btn.setFont(f);

Cancel_Btn = new JButton("Cancel");
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.JLabel all_loc = new
javax.swing.JLabel(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+"\n and loc is: "+l);
                ((Vector)Group.GROUPS.get(i)).remove(l);
                group_loc.setListData(((Vector)Group.GROUPS.get(i)));
            }
        }
        else if (object == Close_Btn) {
            Close_BtnMouseClicked(event);
        } else if (object == Edit_Btn) {
            Edit_BtnMouseClicked(event);
        } else if (object == Cancel_Btn) {
            Cancel_BtnMouseClicked(event);
        } else if (object == Save_Btn) {
            Save_BtnMouseClicked(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);
    gNameL.setVisible(false);
    gName.setVisible(false);
    group_loc.setVisible(false);
    groups.setBounds(10,145,100,15);
    Add_Btn.setVisible(false);
    Remove_Btn.setVisible(true);
    Save_Btn.setVisible(true);
    Save_Btn.setBounds(85,225,70,15);
    Cancel_Btn.setBounds(10,270,75,15);
    grouploc.setVisible(false);
    groups.setVisible(true);
    groupscrollPane.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.setVisible(true);
    Save_Btn.setBounds(85,225,70,15);
    Cancel_Btn.setBounds(10,270,75,15);
    groups.setVisible(true);
    groups.setBounds(10,125,100,15);
    groupscrollPane.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
}