1.00 Tutorial 3

Methods, Classes Arrays & ArrayLists

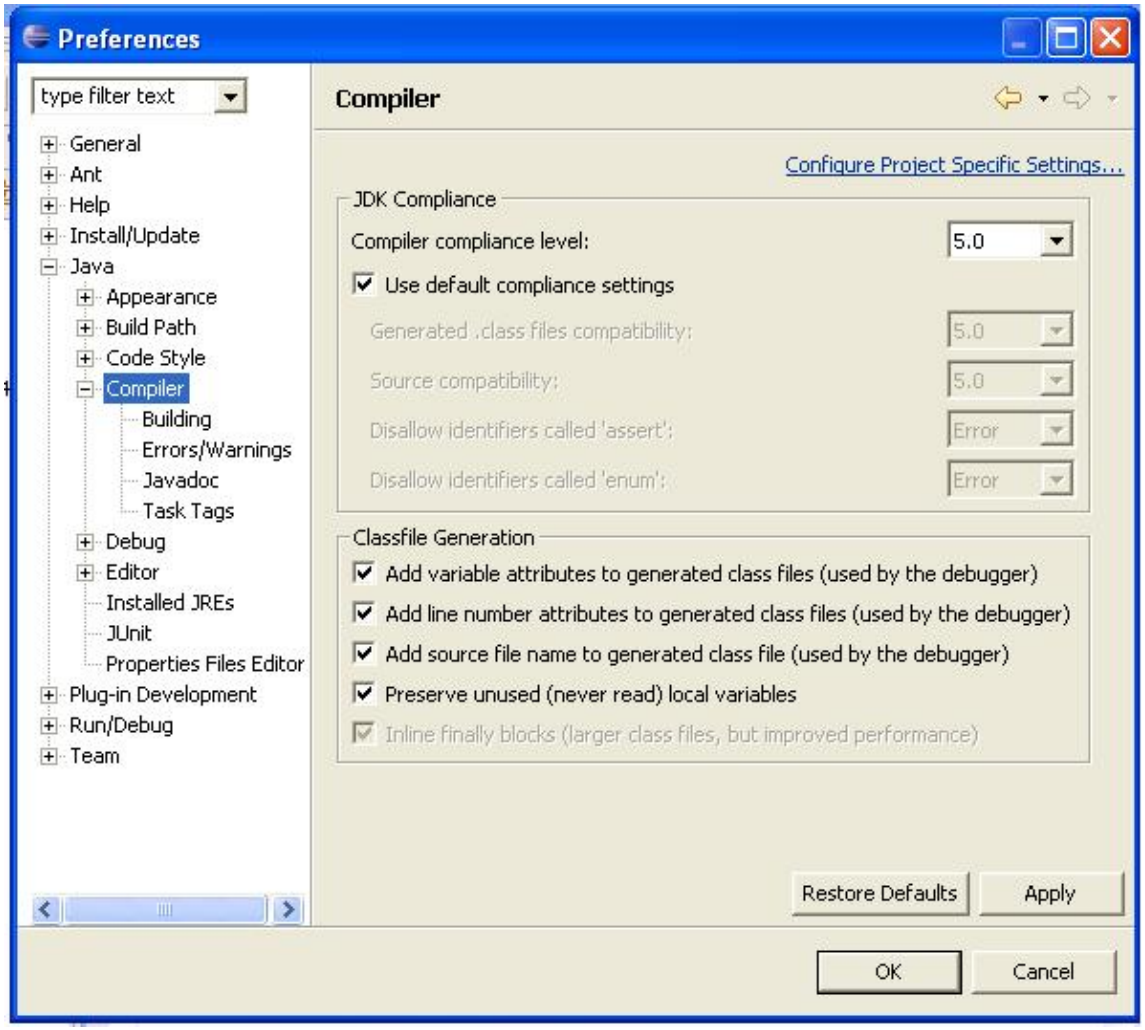
September 26 & 27, 2005

Topics

- Java Compliance
- Methods
  - Pass by Value
  - Access
  - Static methods
- Classes & Objects
- Arrays & ArrayLists
- Problem Set 3 discussion

Java Compliance

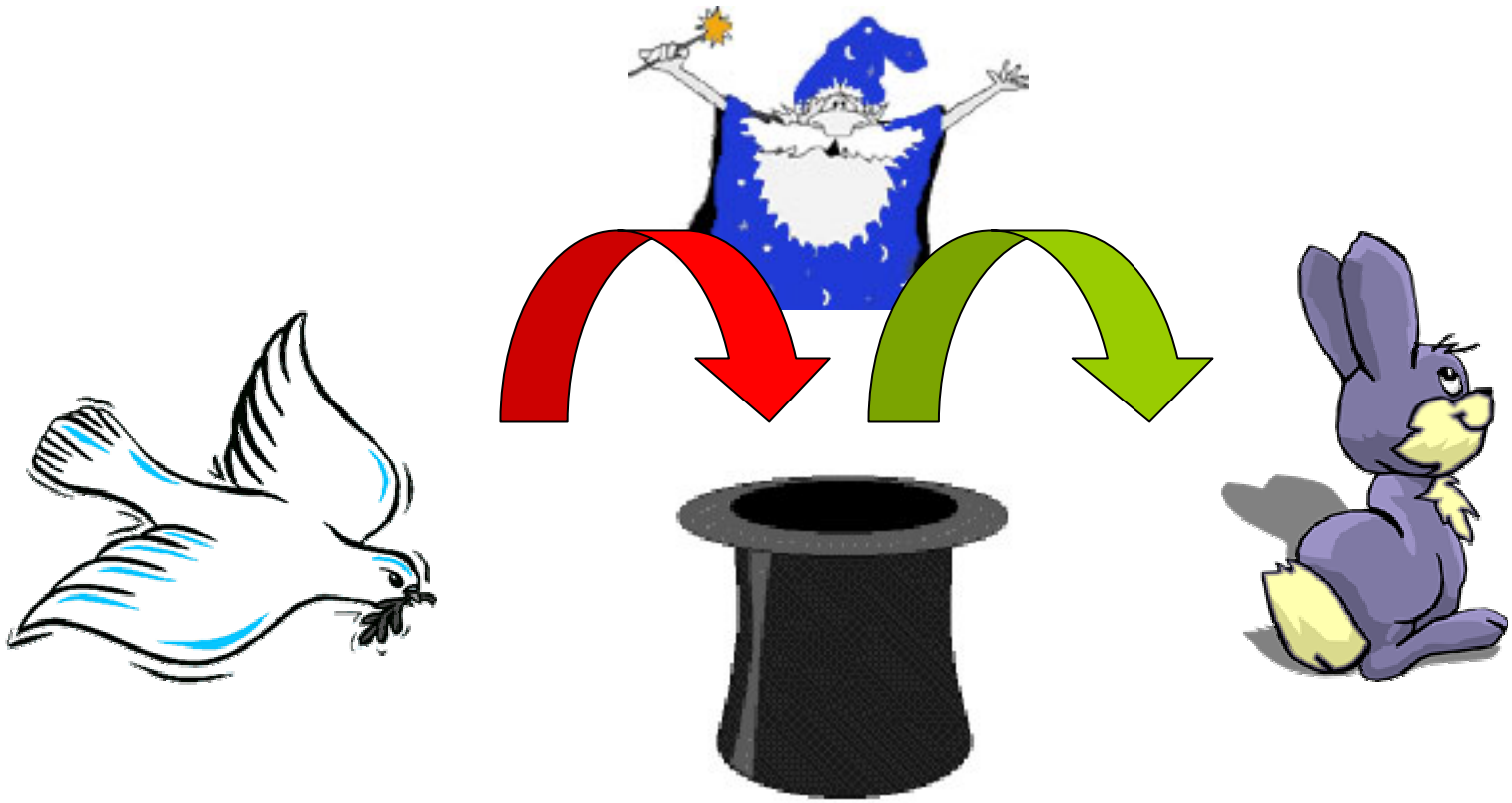
Make sure your Eclipse compiler is set to Java 5.0



Methods

- Methods are the interface or communications between classes
  - They are a useful way of doing the same operation in many places in your program, avoiding code repetition

Methods: Black boxes



```
public Rabbit hatMagic(Dove d) {
      Abracadabra;
}
```

Methods: Pop Quiz

What does the following code do?

```
public class Tutorial3 {
   public static int simpleExample() {
       int sameName = 3;
       System.out.println("samename = "+sameName);
       return sameName;
   }

   public static void main(String[] args) {
       int sameName = 2;
       System.out.println("samename = "+sameName);
       System.out.println("simpleExample returns
                           "+simpleExample());
   }
}
```

Pass by copy/value

- Method arguments (the things in parentheses) are passed by copying them
- This is called pass by value

Pass by copy/value: Pop Quiz

What does the following code fragment print?

```
public class Tutorial3 {
   public static void main(String[] args)
   {
        int i = 1;
        System.out.println("i = " + i);
        int j = increment(i);
        System.out.println("i = " + i);
        System.out.println("j = " + j);
   }
   public static int increment(int i) {
        i = i + 1;
        System.out.println("i = " + i);
        return i;
   }
}
```

Run this in Debug mode in Eclipse and see what happens.

So, What is going on?
The memory stack

`i = 1 in main()`

So, What is going on?
The memory stack

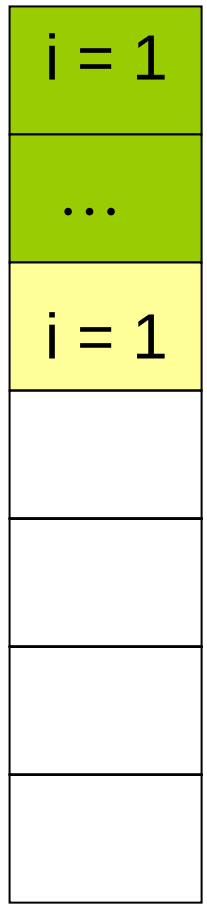
i = 1 in main()

increment(i) is called

i = 1 in increment()

The value of i in main is copied to the variable i in increment.

The original in main is not changed.

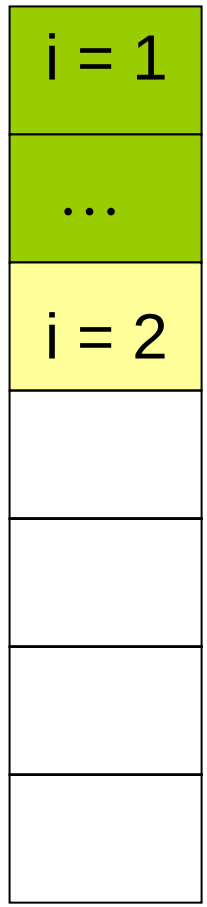


So, What is going on?
The memory stack

i = 1 in main()

increment(i) is called

now i inside increment() is changed



So, What is going on?
The memory stack

i is still 1 in main()

j is assigned the value that is returned by increment.
i inside increment() does not exist anymore.

Access

- private
  - only visible to methods which belong to the same class
- package/default (no access modifier)
  - only visible to methods which belong to the same package
- public
  - visible to all methods

Static

- Static members:
  - are not associated with any particular instance of the class—one copy shared by all instances
  - are accessible to both static and non-static methods
- Static Methods:
  - may only access static members, **not** instance members
  - -may be called using Classname.methodName() or objectReference.methodName()

When to Use Static Methods

- When no access to any instance field is required. Usually one of two scenarios:
  - The method takes in all the information it needs as parameters:

    ```
    Math.pow(double base, double exp)
    ```

  - Or, the method needs access to only static variables.
- Note that the main method must be static
- Example of a static method
  - A method that returns today's date

Modified Class from PSet 2

```
public class Investment {
   //Data Members
   private int type, currentAge;
   private double monthlyRate, moneyInvested,
   totalValue, minValue;

   //Constructor
   public Investment(int type, double moneyInvested){
       this.type = type;
       this.moneyInvested = moneyInvested;
       this.totalValue = moneyInvested;
       this.rate = 0.1162; //Note: this rate is not
                           // the same as in pset 2.
   }
```

Modified Class from PSet 2

```
// A Get method example
   public double getTotalValue () {
        return totalValue;
   }
//A Set method example
   public void setCurrentAge (int newAge) {
        currentAge=newAge;
   }
//Other Method examples
   private double calculateInterest() {
        return totalValue * rate;
   }
   public void updateTotalValue () {
        totalValue += calculateInterest();
   }
}
```

Using the class

```
/* declare variable */
Investment inv;
/* call constructor */
inv = new Investment(1,5600);
System.out.println(inv.updateTotalVal
  ue());
```

Pop quiz:

What happens when you try to call inv.calculateInterest() ?

Arrays vs. ArrayLists

- Arrays are fixed in size;
- Arrays can only hold elements of the same type.
- Arrays can hold both Objects and primitive types;

- ArrayLists can grow & shrink as needed
- In previous versions: ArrayLists can hold any type of object

  In 1.5, have ArrayList type and its elements must be of same type
- In previous versions: no primitive types.

  ArrayLists auto box (& unbox) primitive types into their wrapper class object.

Using Arrays

Three things to do:

- Declare an array

```
Integer[] myIntObject; // Array of Objects
int[] myIntPrimitive ; //Array of primitive data
```

- Create an array

```
myIntObject   = new Integer[2];
myIntPrimitive   = new int[2];
```

- Create/initialize each object in the array

```
myIntObject[0]    = new Integer(1);
myIntObject[1]    = new Integer(2);
myIntPrimitive[0]= 1;
myIntPrimitive[1]= 2;
```

Shortcuts

- Declaring and creating in one step:
  ```
  Integer[] myInts = new Integer[2];
  ```
- Sometimes can declare, create, and initialize all in one step!
  ```
  /* Create an object w/o new keyword! */
  int[] powers={0,1,10,100};
  String[] tas = {“Karin”, “Felicia”, “Daniel”,
    “Charu”};
  Integer[] ints = {new Integer(1), new
    Integer(10)};
  ```
- Use arrayName.length to get number of elements

Using ArrayLists

- Must import java.util.*;
- Common constructors (e.g. of constructor overloading)

```
ArrayList<String> list1 = new ArrayList<String>();
ArrayList<String> list2 = new ArrayList<String>(20);
```

- Adding to a ArrayList

```
list1.add ("Felicia");
list1.add (3, "Daniel");
```

- Getting things out

```
String TA = list1.get(2);
```

- Other methods (look at javadoc):

```
int noTAs = list1.size()
list1.remove ("Karin");
……..
```

Exercise : Using Arrays

- Create an array containing the numbers 1 to 10. Print the values of the array & their sum at each step.

Exercise : Using Arraylists

- Create an ArrayList containing the Integer objects that correspond to the numbers 1 to 10. Print the values of the ArrayList & their sum at each step.

Problem Set 3 : Goals

- To start designing a TIVO system
- To create classes
- To create methods
- To use arrays for data storage

Problem Set 3 : Calendar

Useful things to know:

- Import statement: `import java.util.Calendar;`
- You do not have to use the `Calendar()` constructor directly
  - `Calendar.getInstance()`: returns an `Calendar` object set to the date and time that the method was called.

Problem Set 3 : Calendar

Useful methods. See javadoc for more.

- `get(int field)`
- `clear(int field)`
- `clear()`
- `set(int field, int value)`
- `set(int year, int month, int date)`

Problem Set 3 : Calendar

Useful fields. See javadoc for more.

- `Calendar.DAY_OF_MONTH`,
  `Calendar.DAY_OF_WEEK`,
  `Calendar.YEAR`
- `Calendar.MONDAY`,
  `Calendar.TUESDAY`,
  `Calendar.JANUARY`,
  `Calendar.DECEMBER`