More about GUIs

GEEN163

"The best programmers are not marginally better than merely good ones. They are an order-of-magnitude better, measured by whatever standard: conceptual creativity, speed, ingenuity of design, or problem-solving ability."

Randall E. Stross

Programming Assignment

- A new programming assignment has been posted to Blackboard
- The program requires an array of objects
- You must create two .java files

Using Arrays

- Consider a program that reads a file of racer's times and names
- It then displays the names and times of the racers plus how much more or less they were from the average

Race Average Solution

```
double[] data = new double[100];
String[] name = new String[100];
double   avg = 0.0;
int      numData = 0;

while (inFile.hasNextDouble()) {
    data[numData] = inFile.nextDouble();
    name[numData] = inFile.next();
    avg = avg + data[numData];
    numData++;
}

avg = avg / numData;
System.out.println("Average time is "+avg);
for (int indx = 0; indx < numData; indx++) {
    System.out.println(name[indx]+" "+data[indx]+
                       " "+ (data[indx]-avg));
}
```

Using an Array of Objects

- Objects can be created for each competitor and put into an array
- Both the runner's name and time can be stored together in an object
- The objects can be stored in one array instead of two arrays

Runner class

```
public class Runner {
    String   name;          // name of the competitor
    double   time;          // time in the event

    // constructor method
    public Runner( String who, double howLong) {
        name = who;
        time = howLong;
    }
}
```

Race Time Lab with Objects

```
Runner[] racer = new Runner[100];
double   avg = 0.0;
int      numData = 0;

while (inFile.hasNextDouble()) {
    double seconds = inFile.nextDouble();
    String person  = inFile.next();
    racer[numData] = new Runner( person, seconds );
    avg = avg + seconds;
    numData++;
}

avg = avg / numData;
System.out.println("Average time is "+avg);
for (int indx = 0; indx < numData; indx++) {
    System.out.println(racer[indx].name+" "+
        racer[indx].time+" "+(racer[indx].time-avg));
}
```

Arrays of Objects

- The racer array has space to hold 100 objects, but was originally empty
- Objects were created as the data was read and put into the array
- The numData variable is used to count the number of objects in the array and as an index for new objects

Interesting Objects

Many GUI program can be created with a few simple components

- **JLabel** – Displays text in the GUI
- **JButton** – Creates a button you can press
- **JTextField** – Creates an input text box

- All of these objects are from the package javax.swing

How do you make an array of 5 buttons for a GUI?

A. JButton[5] gazelle = new JButton[];

B. JButton gazelle = new JButton(5);

C. JButton[] gazelle = new JButton[5];

D. JButton[] gazelle = { JButton(), JButton(), JButton(), JButton(), JButton() };

E. You cannot make an array of GUI objects

Implementing ActionListener

- Implementing the ActionListener interface makes the program respond to GUI input
- Programs implementing ActionListener must have an actionPerformed method

```
public class MyApplet
        extends javax.swing.JFrame
        implements java.awt.event.ActionListener {
```

Interfaces

- A Java interface has a list of method headers that a program must implement
- ActionListener is an interface that requires a program to include an actionPerformed method
- Programmers can create their own interfaces
  - This is a topic for GEEN165

Adding a Listener

- The method addActionListener should be called on each GUI component that can do something
- Do not call addActionListener on components that do not do anything, like JLabels

```
javax.swing.JButton  frog = new
                        javax.swing.JButton("Go");
frog.addActionListener( this );
```

“this” in Java means

A. this program
B. this method
C. this object
D. this class
E. this variable

Taking Action

- When a user clicks an object with an ActionListener, Java calls the method

```
public void actionPerformed(
                java.awt.event.ActionEvent   rabbit) {
    // do something here
}
```

- This method should do what the user expects to happen when the button is pressed

Getting the Text

- You can get the text from a GUI component with the getText() method
- This is useful to get the text a person entered in the JTextField

```
javax.swing.JTextField  llama = new
                                javax.swing.JTextField();
// in the actionPerformed method
String answer = llama.getText();
```

Changing a Component's Text

- A useful method is setText It can be used to change the text of any object that has text, such as JButtons, JLabels and JTextfields

```
javax.swing.JLabel  msg   =
        new javax.swing.JLabel("Do something");

msg.setText("Don't do something");
```

Who Dunit?

- Many GUIs have more than one active object
- You can determine where the action was taken by using the **getSource()** method of the event
- The result of **getSource()** should be compared to each GUI object

```
public void actionPerformed(
                    java.awt.event.ActionEvent   frog ) {
    if ( frog.getSource() == GUIobject ) {
```

Example Use of getSource()

```
JButton delete = new JButton("kill");
JButton save   = new JButton("allow");
        - - - - - - - - - - - - - - - - - - - - - - - - -
public void actionPerformed(
                java.awt.event.ActionEvent   toad) {
    if (toad.getSource() == delete ) {
        // delete something
    } else {
        // don't delete anything
    }
```

Looping in GUI Programs

- When you want a user to enter multiple values in a GUI program, you probably do **not** need a for loop or a while loop
- Each time the user enters a name, it will call actionPerformed

What will select the pressed button?

```
JButton deer = new JButton();
public void actionPerformed(
                    java.awt.event.ActionEvent  bear)
```

A. if(bear == deer)
B. if(bear.getSource() == JButton)
C. if(bear.getSource() == deer)
D. if(deer == JButton.getSource())
E. if(this == bear.getSource())

setVisible

- Most GUI components have a setVisible method

  setVisible(boolean)

- If the boolean parameter is true, the object shows, if false it does not show
- Calling **`setVisible(true)`** makes the frame appear onscreen

Disabling GUI Components

- Individual GUI components (i.e. JButtons) can be made invisible or disabled
- A disabled component is gray and cannot be selected

```
JButton myButton = new JButton("OK");
myButton.setEnabled(false);
```

- This will make the button gray and unusable

setTitle

- The setTitle method of a JFrame sets the text in the upper border of the window

```
setTitle("My GUI program");
```

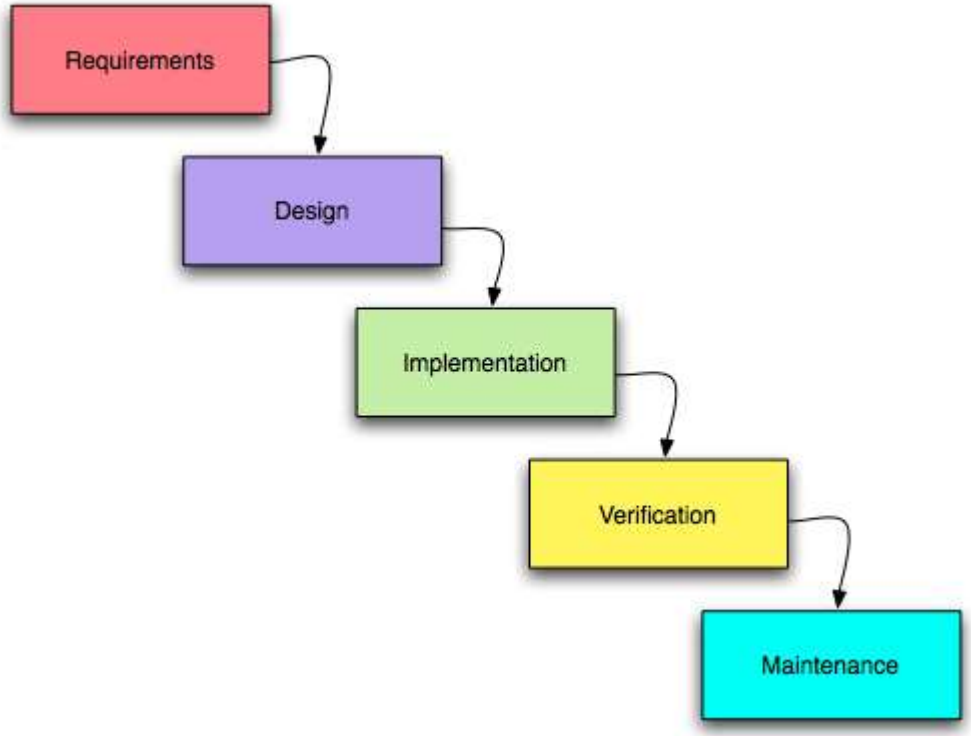
Software Engineering Steps

- **Requirements** – Define exactly what the program is supposed to do from the user's point of view
- **Design** – How will the program run
- **Implementation** – Write the Java
- **Verification** – Test the program looking for any errors
- **Maintenance** – Fix the errors you missed the first time

Waterfall Model

- The waterfall model is one software engineering technique for creating programs

- There are several other software engineering processes



Algorithm

- An algorithm is a step-by-step procedure for solving a problem
- Algorithms can be expressed in several ways
  - English
  - Flow charts
  - Formal logic
  - pseudo code
- An algorithm is less detailed than a program, but provides all the information necessary to solve the problem

What should be done first?

A. Write Java code
B. Write a design document
C. Talk with the customer
D. Fix any bugs

Program Description

- Given the cost of a purchase and the amount paid, determine how many and what coins to give in change

Write an algorithm

- Briefly explain how a program could determine the coins to return
- Assume the program has read the cost and amount paid and calculated the change = paid - cost

Algorithm for Change

- Read the cost and amount paid
- Calculate the amount of change
- For each coin from largest to smallest
- While the change is more or equal to the coin
  - Give a coin
  - subtract the coin from the change

Translating Algorithm to Java

- How should the data be represented?
- Since money is represented as a number with decimal digits, we can use doubles
- Variables needed are:
  - cost = input from user
  - paid = input from user
  - change = paid – cost
  - array of coin values

Initial Data Declarations

```
double cost;       // input cost
double paid;       // input amount paid
double change;  // amount to be returned
double[] coin = {0.25, 0.10, 0.05, 0.01};    // coins
```

- We may find we need more variables later

Input Algorithm

- Read the cost and amount paid
- Calculate the amount of change

```
cost = keyboard.nextDouble();
paid = keyboard.nextDouble();
change = paid – cost;
```

For all coins

- For each coin from largest to smallest

- Since the array is of fixed size, a **`for`** loop would be appropriate

```
for (int denom = 0; denom < coin.length; denom++)
```

Calculate Change for a Coin

- While the change is more or equal to the coin
  - Give a coin *(increment a counter)*
  - subtract the coin from the change

```
int count = 0;
while ( change >= coin[denom] ) {
        change -= coin[denom];
        count++;
}
System.out.println(count+" "+coin[denum]);
```

Full Program

```
public class ChangeDouble {
 public static void main(String[] unused) {
   double[] coin = {0.25, 0.10, 0.05, 0.01};   // coins
   double cost, paid;                          // input cost
   double change;                              // amount to return
   java.util.Scanner keyboard = new
                          java.util.Scanner(System.in);
   System.out.print("Enter the cost and amount paid >");
   cost = keyboard.nextDouble();               // read input
   paid = keyboard.nextDouble();
   change = paid - cost;
   for (int denom = 0; denom < coin.length; denom++) {
      int count = 0;
      while (change >= coin[denom]) {
         change = change - coin[denom];
         count++;
      }
      System.out.println(count+" "+coin[denom]+" cent ");
   }
}}
```

Testing

- What are good test values?
- What will the program do if incorrect values are given?

What will the program do if the cost is greater than the amount paid?

A. Show 0 for all coins
B. Infinite loop
C. Show negative number of coins
D. none of the above

Enhancements

- How might we improve the output by identifying the coins by name (e.g. “dime”)?

- What would it take to provide change for Euros

1c, 2c, 5c, 10c, 20c, 50c, €1, €2

Programming Assignment

- A new programming assignment has been posted to Blackboard
- The program requires an array of objects
- You must create two .java files