Today

• Programming Assignment 6 : Spampede! (10/20)
• Worksheet 2 due today…

On the horizon:

Wed 10/18 “Parsing” -- programming computers to understand text.
Fri 10/20 Spampede applet officially due (actually due Sun.)
Sun 10/22 Review session for midterm exam (time pref.)?
Mon 10/23 After class -- take-home exam given out (2 hr.)
Wed 10/25 Before class -- take-home exam due by 2:30pm.

\[ x = 'N'; \]
\[ \text{if } (x == 'N') x = 'S'; \]
\[ \text{if } (x == 'S') x = 'N'; \]

Interfaces

“Classes without code”

interface ActionListener
{
    void actionPerformed(ActionEvent evt);
}

interface KeyListener
{
    void keyPressed(KeyEvent evt);
    void keyReleased(KeyEvent evt);
    void keyTyped(KeyEvent evt);
}

interface Runnable
{
    void run();
}

interface ActionListener
{
    void actionPerformed(ActionEvent evt);
}

Idea: programming by contract

Events

• Events are things that happen to a graphical application
  • Button Presses
  • Text Entries
  • Key Presses, Key Releases, Key Events

• Each object receiving an event notifies its “Listener”
• The Listener then handles the event appropriately

Today

• Homework 6: Applet!
  * Using the Deque class to model a centipede...

• Midterm exam in class on: Wednesday, March 13

Assignment

```
Deque D1 = new Deque();
Deque D2 = new Deque();
D1.enqueue("0");
D1.enqueue("1");
D1.enqueue("2");
D2.enqueue("3");
D2 = D1;
D2.dequeue();
D2.enqueueFront("I");
D1.enqueue("N");
D2.enqueue("Q");
System.out.println(D1);
System.out.println(D2);
```

What does this code print?

“Thinking like a machine”
Comments?

Commenting of each method is important, but those comments don’t have to be particularly lengthy:

```java
// what the functions below do
// shall remain shrouded in mystery...
public static boolean is_empty(Queue Q) // SDQMT?
{
    return Q.isEmpty();
}
public boolean is_empty() // I12CFQSMT.
{
    return (front == null && back == null);
}
```

Inheritance summary

**Ideas**
- Models the “kind-of” relationship among classes
- Factors out common code from those classes
- Function overriding allows old code to call new code

**Keywords**
- extends, super, implements

Inheritance Application: Applets!

- Applet: A small application designed to run in a browser

Java’s Graphics Support: AWT

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.awt</td>
<td>1.0</td>
<td>Abstract windowing toolkit</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td>Original version, accepted by most web browsers</td>
</tr>
<tr>
<td></td>
<td>1.2, 1.3, 1.4</td>
<td>Monstrous library of components</td>
</tr>
<tr>
<td>java.swing</td>
<td>1.2, 1.3, 1.4</td>
<td>Not supported by any browser by default</td>
</tr>
</tbody>
</table>

AWT Components

- canvas (for drawing)
- applet (is a panel)
- panel (contains other components)

Spampede Example (before)

http://www.cs.hmc.edu/~sstudent/Spampede.html
import java.awt.*;
import java.awt.event.*;

public class Spampede extends Applet
    implements ActionListener, Runnable
{
    Graphics graphics;           // where drawing takes place
    Image    image;              // off-screen image
    private Button redButton;    // initalize things here…
    private Button greenButton;
    private Button startButton;
    private Button pauseButton;
    private TextField textInput;

    public void init()
    {
        image = createImage(getSize().width, getSize().height);   // double buffer
        graphics = image.getGraphics();                           // for drawing
        clear();
        this.addKeyListener(this);
        redButton = new Button("Red");              // initialize things here…
        redButton.addActionListener(this);
        redButton.addKeyListener(this);
        add(redButton);
        …
        textInput = new TextField("Maze Name",25);
        textInput.addActionListener(this);
        add(textInput);
    }

    public void keyPressed(KeyEvent evt)
    {
        graphics.setColor(Color.white);
        graphics.fillRect(300,180,100,40);
        graphics.setColor(Color.black);
        graphics.drawString("Key " + evt.getKeyChar() + " pressed",300,200);
        repaint();
    }

    final java notes
    Avoid "magic numbers"!
    even Prof. Benjamin agrees…
    static final int EAST = 0;
    static final int WEST = 1;
    static final int NORTH = 2;
    static final int SOUTH = 3;
    // unchangeable values
    These constants make code easier to read and write, e.g.,
    int currentHeading = EAST;

    Interfaces “Classes without code”
    interface KeyListener
    {
        void keyPressed(KeyEvent evt);
        void keyReleased(KeyEvent evt);
    }
    interface Runnable
    {
        void run();
    }
    interface ActionListener
    {
        void actionPerformed(ActionEvent evt);
    }
    public class ExampleApplet extends Applet
        implements KeyListener, ActionListener, Runnable
    {
        If you claim you’ll implement them, you have to implement them.
        (They can be empty methods, but they have to exist.)
    }

    Events
    • Events are things that happen to a graphical application
      • Button Presses      • Text Entries
      • Key Presses, Key Releases, Key Events
    • Each object receiving an event notifies its “Listener”
    • The Listener then handles the event appropriately

    Drawing Calls
    • The drawing commands are encapsulated in the Graphics class
    (graphics is the data member’s name)
    void setColor(Color c)
    void fillRect(int x, int y, int width, int height)
    void fillOval(int x, int y, int width, int height)
    void fillPolygon(int[] xPoints, int yPoints, int nPoints)
    Each of the above have “draw” versions:
    drawRect, drawOval, drawPolygon
    void drawString(String str, int x, int y)
    void drawLine(int x1, int y1, int x2, int y2)
    void drawImage(Image img, int x, int y, null)
Drawing

Some event occurs, such as calling `repaint()` or making the window visible.

`update()` is called on the window’s graphics, which then calls `paint()`.

Whatever is drawn in `paint` is displayed.

Double Buffering

To avoid flicker,

Reuse! (Others’ experience)

You need an HTML file to load your applet.

```
<HTML>
  <APPLET CODE="Spampede.class"
    WIDTH=700
    HEIGHT=500>
  </APPLET>
</HTML>
```

Use the Java Console. Anything printed will go there.

Example applets available at www.cs.hmc.edu/~cs60grad

Reuse! (Others’ code)

Component
  Container
    Panel
      Applet
        Spampede

Test each one!

Bottom-up Software Strategies

Incremental Changes

**CODE**

<table>
<thead>
<tr>
<th>Code</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TEST**

<table>
<thead>
<tr>
<th>Code</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Concentrated (proper phase)

Building tools

**KEY** What will have to be done many times?

**THEN** Write a method (function) to handle it.
Threads

Getting two programs for the price of one
Each thread is considered an independent process
They alternate in controlling the applet.
When they alternate is not specified (in general).

Abstraction
- event handler (default thread)
- centipede and spam updater

Implementation
- event handler (default thread)
- centipede and spam updater

Potential Problem 1
"The Corruptor"
- Thread 1 is changing an object (say, adding 1 to the score) and gets switched out.
- Thread 2 takes over and also changes the same object (say, subtracting 1 from the score).

What can go wrong?
- How to add 1:
  - load score to a register (holding pen)
  - add 1 to the register
  - store the register’s new contents into score

“synchronized” makes code atomic

Potential Problem 2
"The Nonterminator"
- Thread 1 is holding an object, like the "Score" object, and needs another object (say, a network connection), so it sleeps (or switches out).
- Thread 2 takes over and starts using the network connection when it realizes it needs to update the Score object.
- Since the Score object is locked by another thread, Thread 2 goes to sleep (or switches out), and...

Result: Deadlock