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';
if (x == 'N')  x = 'S';
if (x == 'S')  x = 'N';
```

ouch!

Interfaces "Classes without code"

```
interface KeyListener
{
    void keyPressed(KeyEvent evt);
    void keyReleased(KeyEvent evt);
    void keyTyped(KeyEvent evt);
}
interface Runnable
{
    void run();
}
interface ActionListener
{
    void actionPerformed(ActionEvent evt);
}
```

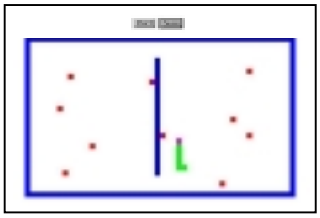
Idea: programming *by contract*

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

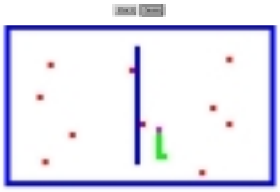
- Homework 6: Applet!
  - Using the Deque class to model a centipede...



- Midterm exam in class on: Wednesday, March 13

Today

- "Double secret extension" … **for HW5's Ex.Cr. to *this* weekend**
- 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:

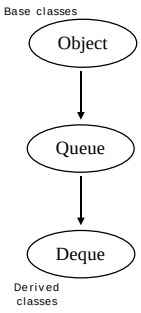
```
// what the functions below do
// shall remain shrouded in mystery...

public static boolean is_empty(Queue Q)  // S D Q M T ?
{
   return Q.isEmpty();
}

public boolean is_empty()      // I 1 2 C F Z Q S M T .
{
   return (front == null && back == null);
}
```

Credits: Jim Norwood

Inheritance summary



Ideas

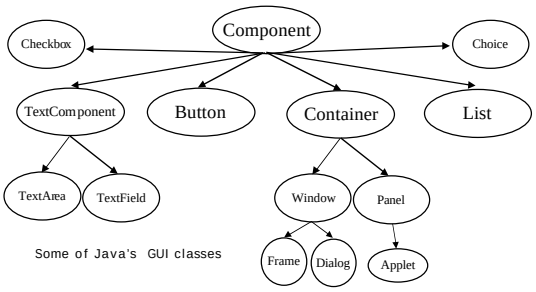
- Models the "kind-of" relationship among classes
- Factors out common code from those classes
- Function overriding allows old code to call new code

Keywords

used with interfaces, not necessarily inheritance...

- extends, super, implements

Inheritance Application: Applets!



Some of Java's GUI classes

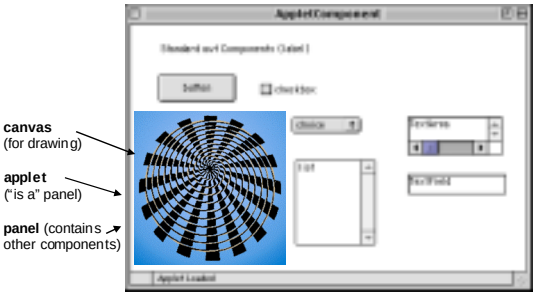
- Applet: A small application designed to run in a browser

Java's Graphics Support: AWT

Abstract windowing toolkit

- java.awt 1.0
  - Original version, accepted by most web browsers
- java.awt 1.1
  - Updated version
  - one difference: cleaner event handling
  - "event listeners"
  - supported by IE and Netscape by default
- java.swing 1.2, 1.3, 1.4
  - monstrous library of components
  - not supported by any browser by default

AWT Components



Spampede Example (before)

http://www.cs.hmc.edu/~situdent/Spampede.html

Spampede.java (before)

/cs/cs60/as/a6/Spampede.java

```
import java.applet.*;                    /* and other fine imported files */
import java.awt.*;
import java.awt.event.*;

public class Spampede extends Applet
implements ActionListener, KeyListener, Runnable

{                                        /* data members */
        Graphics  graphics;              //  where drawing takes place
        Image     image;                 //  off-screen image
  private Button  redButton;
  private Button  greenButton;
  private Button  startButton;
  private Button  pauseButton;
  private TextField  textInput;
```

Spampede.java (before)

```
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);
}
```

- No constructor needed -- use init()

indicates what Object should handle the button or key events

Interfaces "Classes without code"

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

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

```
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;

static final int XOFFSET = 100;
static final int YOFFSET = 100;
static final int CELLSIZE = 10;
```

unchangeable values

These constants make code easier to read and write, e.g.,

```
int currentHeading = EAST;
```

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

These methods can be overridden.

update() is called on the window's graphics, which then calls **paint**()

whatever is drawn in paint is displayed

Double Buffering

to avoid flicker

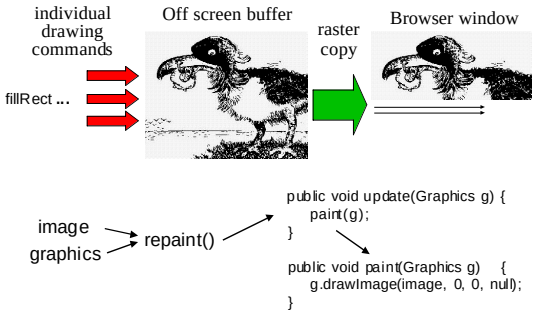


image
graphics
repaint()

```
public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g)  {
    g.drawImage(image, 0, 0, null);
}
```

Reuse ! (others' experience)

You need an HTML file to load your applet

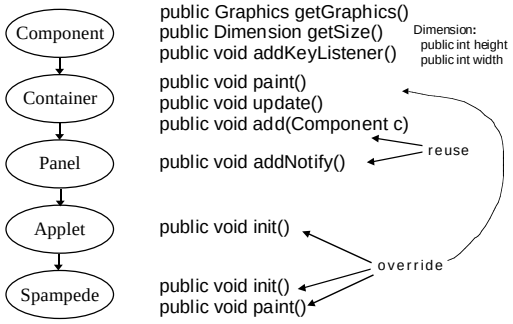
```
< HTML >
< APPLET CODE = "Spampede.class"
        WIDTH = 700
        HEIGHT = 500 >
< /APPLET >
< /HTML >
```

Use the Java Console ← anything printed will go there

Use shift-reload in Netscape, control-refresh in IE

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

Reuse ! (others' code)



Bottom-up Software Strategies

Incremental Changes

CODE

TEST

concentrated despair phase

code
test
code
test
code
test
...

closing in on a solution

Building tools

KEY What will have to be done many times ?

THEN write a method (function) to handle it .

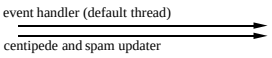
Test each one!

Threads

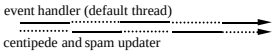
Getting two programs for the price of one

- Each thread is considered an independent process
- They alternate in controlling the applet.
- When they alternate is not specified (in general).

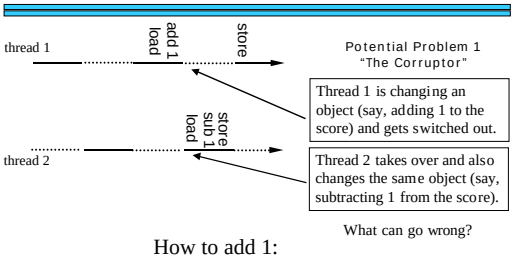
Abstraction



Implementation



Threads



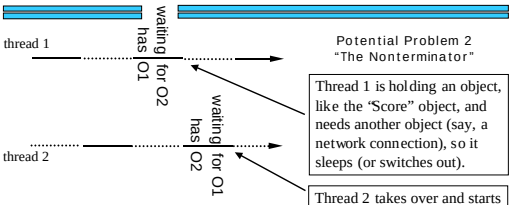
What can go wrong?

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

Threads



Result: Deadlock