GUIs and Event-Driven Programs

February 29–March 1, 2012
CS 60: Principles of Computer Science

Assignment 6 due March 5: Spamede!
Midterm out March 5–6
Midterm due back 5pm Friday, March 9
**Explain the Difference (1)**

```java
class Dog {
    private String name;

    public Dog(String dname) {
        this.name = dname;
    }

    public void speak() {
        System.out.println(this.name + "": woof";)
    }
}

class Cat {
    private String name;

    public Cat(String cname) {
        this.name = cname;
    }

    public void speak() {
        System.out.println(this.name + "": meow";)
    }
}

class Animal {
    protected String name;
    public Animal(String n) {
        name = n;
    }

    protected void say(String s) {
        System.out.println(name + ":" + s);
    }
}

class Dog extends Animal {
    public Dog(String dname) {
        super(dname);
    }

    public void speak() {
        this.say("woof");
    }
}

class Cat extends Animal {
    public Cat(String cname) {
        super(cname);
    }

    public void speak() {
        this.say("meow");
    }
}
```
EXPLAIN THE DIFFERENCE (2)

```java
class Animal {
    protected String name;
    public Animal(String n) {
        this.name = n;
    }

    protected void say(String s) {
        System.out.println((this.name + "::" + s);
    }
}

class Dog extends Animal {
    public Dog(String dname) {
        super(dname);
    }

    public void speak() {
        this.say("woof");
    }
}

class Cat extends Animal {
    public Cat(String cname) {
        super(cname);
    }

    public void speak() {
        this.say("meow");
    }
}

interface Animal {
    public void speak();
}

class Dog implements Animal {
    private String name;

    public Dog(String dname) {
        this.name = dname;
    }

    public void speak() {
        System.out.println((this.name + ":: woof");
    }
}

class Cat implements Animal {
    private int ears;

    public Cat(int e) {
        this.ears = e;
    }

    public void speak() {
        System.out.println((this.ears + ":: meow");
    }
}
```
Some Inheritance in Java

class Spampede extends JApplet

Some of Java’s GUI classes
Spampede INHERITANCE

Object

Component

Container

Panel

JApplet

Spampede

public void requestFocus()
public void addKeyListener()
public void repaint()

public void add(Component c)

public void init()

public void paint()

void drawEnvironment()

void displayMessage()

void reset()
Homework Assignment 6

Spampede

1. Improvements `Maze.java`
   (including changes to permit inheritance)

2. Subclass `SpamMaze.java`
   (support for updates to spam and pede)

3. Graphics and user interface: `Spampede.java`
Step 1: Maze.java 2.0

1. Improve the interfaces of Maze and MazeCell
2. Put the maze in the code (Web applets generally can't read files)
3. Find the nearest destination cell among many
New Maze Methods

Find the nearest destination cell among many:

class Maze extends Object
{
    ...
    public MazeCell multiBFS(MazeCell start, char dest) { ... }
}

Why is dest no longer a MazeCell?
Why does the method return a MazeCell?
What cells must we avoid now?
What if there’s no solution?

For testing: be able to print the maze with the path you found, but remove the path before returning!
```java
private static final String[] mazeStrings = {
    "**************************************************",
    "*PS D ...

'S' == head of the centipede

'P' == body of the centipede

'D' == spam

'*' == wall

30 rows
50 columns

50 columns
30 rows

```
**Note: Avoid Magic Constants!**

Named constants make code easier to read and easier to modify, e.g.,

```java
public static final char SPAM = 'D';
public static final char START = 'S';
public static final char WALL = '*';
public static final char PEDE = 'P';
```

```java
multiBFS( start, Maze.SPAM );
```
**Assignment Part 2: SpamMaze Class**

class SpamMaze extends Maze {

Additional fields:

✓ spamCells
✓ pedeCells

Additional methods

✓ addSpam
✓ removeSpam
✓ advancePede
✓ reversePede

What type should spamCells and pedeCells be?
Code Reuse: Java's LinkedList<...> class

http://docs.oracle.com/javase/6/docs/api/

import java.util.LinkedList;

EEEEk! What is going on here!?
Usage

```java
import java.util.LinkedList;
...

LinkedList<MazeCell> pedeCells
...
pedeCells = new LinkedList<MazeCell>();
pedeCells.addFirst(maze[1][2]);
pedeCells.addLast(maze[1][1]);

MazeCell head = pedeCells.getFirst(); // peek
```

Linked list support the “double-ended queue” operations: adding to the front and back, removing from the front and back, peeking at the front and back
Moving the Centipede

updatePede(char dir)

How might we update the double-ended queue as the centipede moves? (eats?)
Big Idea 1: The Model-View-Controller Pattern

This design pattern separates three basic functional components so that…
Big Idea 1: The Model-View-Controller Pattern

so that the programmer can *take a narrower focus* in building, revising, etc.!
Big Idea 2: Event-Driven Programming

All user-interface “events,” e.g.,

✓ Key presses
✓ Mouse clicks
✓ Window resizes

are gathered into an event queue.

The system (here, Java Swing) examines each event.

If you’ve said you care about this kind of event, the system will let your program know (via a callback). If not, the event is ignored and discarded.
**History: Java AWT**

- **canvas** for graphics and images
- **applet** “is a” panel
- **panel** contains other components

Wow! this is a really, really old image…
More Modern: Java Swing

Basic Controls
Simple components that are used primarily to get input from the user; they may also show simple state.

- JButton
- JCheckBox
- JComboBox
- JList
- JMenu
- JRadioButton
- JSlider
- JSpinner
- JTextField
- JPasswordField
Big Idea 3: Callbacks (Don’t call me, I’ll call you!)

How can we figure out when a button is used?

1. Polling

Hey, is the button being used?
How about now?
How about now?
How about now?
How about now?

2. Callbacks

Dear Swing, please run the `actionPerformed` method (on this object) when this button is used. Sincerely, Spampede

…snoring sounds…
Big Idea 4: Double Buffering

At each step, we erase the picture and redraw it from scratch. This can cause “flickering.” Better: do all the drawing “off screen” and just show the final result.

individual drawing commands

Off screen buffer

raster copy

Screen

text

possibly slow to draw...

gif

image

copy

but fast to copy
Some Java Specifics

class Spampede {
    Image image; // off-screen buffer
    Graphics g; // drawing tools for that buffer
    ... 
    public void init()
    {
        this.image = createImage(getSize().width, getSize().height);
        this.g = image.getGraphics();

        pauseButton = new Button("Pause"); // Create a button
        pauseButton.addActionListener(this); // Register interest in events
        ...
    }
    ...
    g.setColor(Color.red);
    g.fillRect(100,100,10,10);
    ...
}

Lotsofotherdrawingcommandsavailable. See
✓  http://docs.oracle.com/javase/6/docs/api/java/awt/Graphics.html
✓  http://docs.oracle.com/javase/6/docs/api/java/awt/Graphics2D.html
Exception Handling

What could go wrong loading sound and image files?

```java
try {
    URL url = getCodeBase();
    audioCrunch = getAudioClip(url,"crunch.au");
    imageSpam = getImage(url,"spam.gif");
    System.out.println("successful loading of audio/images!");
} catch (Exception e) {
    System.out.println("problem loading audio/images!");
    audioCrunch = null;
    imageSpam = null;
}
```
Spampede.cycle

```java
void cycle()
{
    this.updateCentipede();  // update the Spampede deque
    this.updateSpam();       // update the Spam deque
    this.drawEnvironment();  // draw things to buffer
    this.displayMessage();   // display messages
    repaint();               // send buffer to the screen
    this.cycleNum++;         // One cycle just elapsed
}
```

How could we update the spam more slowly than the spampede?
Name: "Quiz:" Code Treasure Hunt

1. Start by looking over Spampede’s fields…
   1.1 What is the TYPE and NAME of the data structure holding our MazeCells?
   1.2 What is the TYPE of this.dir? What is the type of this?
   1.3 Where do image and g get initialized (assigned to)?

2. Find cycle and then find each method that cycle calls.

3. Find the code that draws the small red square.

4. What do the four arguments to fillRect() mean?

5. How could you use fillRect (repeatedly) to draw the maze?

6. At what coordinates is the spam image being drawn?

7. What keypresses make the large square blue?

8. What keypress is incorrectly identified in the on-screen message?

9. What event produces a hearty spam-consuming crunch sound?

10. EXTRA: How long does it take for the large square to return to magenta?
APPLET: A PROGRAM DELIVERED VIA A BROWSER

<html>
<head>
  <title>Spampede Applet</title>
</head>

<body bgcolor = "#dddddd">

  <center>
  <APPLET
    CODE = "Spampede.class"
    WIDTH = 700
    HEIGHT = 550
    >
  </APPLET>
  
</center>

</body>
Suggested Strategy

1. Finish Maze and SpamMaze first!

2. Make sure you can change, compile, and run Spampede.java
   appletviewer Spampede.html

3. Make sure you understand the purpose of the code in Spampede.java
   (including all the fields!)

4. Write drawEnvironment and test

5. Write updatePede and test

6. Write keyPressed and test (one direction at a time?)

7. Add AI (multiBFS does almost all the work)

8. Add reversing (least important, and can be a little tricky)