Announcements

Program 8 up, due Friday. Requires visit w/ staff
OWL assignments fini!

Final Exam: Wednesday, 12/17, 8 AM, Bartlett 65

Last year’s final up, discussed Wed in discussion.

Special arrangements? Let me hear from you!
What's on the final? - Obvious emphasis on material from last half of class:

Arrays

Inheritance

Files

Interfaces

Graphics

The event Model

Also: randomness, exceptions, layouts, methods
Also, some new (this year) topics are possible:

- Recursion
- Regular Expressions
- ArrayLists
- 2D arrays
- Menus
Scrabble analysis

K1 - points array local to getScore; for loop in getScore is awkward; processLine logic very good

K2 - StringTokenizer incorrect; scoring should be separate method; naming conventions screwy; final report about words of a certain size out of place; PointAssign method oddly done, inefficient.

K3 - StringTokenizer incorrect; toLowerCase issues; placement of scoring array; recomputes scores

K4 - processLine does too much work!!! Scoring based on arrays more efficient (but if stmts at least are in right order)
public class MenuDriver{

    public static void main(String[] args){
        DisplayWindow d = new DisplayWindow();
        JMenuBar menuBar = new JMenuBar();
        d.setJMenuBar(menuBar);
        ColorPanel2 p = new ColorPanel2(menuBar);
        d.addPanel(p);
        d.showFrame();
    }
}
JMenu colorMenu = new JMenu("Colors");
JMenuItem red = new JMenuItem("red");
JMenuItem blue = new JMenuItem("blue");
JMenuItem quit = new JMenuItem("Quit");
JMenu shapeMenu = new JMenu("Shapes");
JMenuItem square = new JMenuItem("square");
JMenuItem circle = new JMenuItem("circle");
boolean squareShape = true;
public ColorPanel2(JMenuBar b) {
    b.add(colorMenu);  // first menu
    b.add(shapeMenu);  // second menu
    colorMenu.add(red); colorMenu.add(blue);
    colorMenu.add(quit);
    shapeMenu.add(square); shapeMenu.add(circle);

    setPreferredSize(new Dimension(700,600));

    red.addActionListener(this);
    blue.addActionListener(this);
    quit.addActionListener(this);
    square.addActionListener(this);
    circle.addActionListener(this);
}
public void paintComponent(Graphics g){
    super.paintComponent(g);
    if (squareShape) g.drawRect(100, 100, 200, 200);
    else g.drawOval(100, 100, 200, 200);
}
public void actionPerformed(ActionEvent e) {

    JMenuItem m = (JMenuItem)e.getSource();

    if (m == red) this.setBackground(Color.red);
    if (m == blue) this.setBackground(Color.blue);
    if (m == square) squareShape = true; repaint();
    if (m == circle) squareShape = false; repaint();
    if (m == quit) System.exit(0);
}
import java.awt.*;import java.awt.event.*;import javax.swing.*;

public class ButtonCounter extends JPanel implements ActionListener{
   int counter = 0;
   JButton clicker = new JButton("press me");
   JLabel count = new JLabel("" + counter);

   public ButtonCounter(){
      setBackground(Color.red);
      this.add(clicker); this.add(count);// 3
      // register ButtonClickListener obj as listener for clicker 4
   }

   public void actionPerformed(ActionEvent e){ // 5
      if(e.getSource() == clicker){ // 5
         counter++;
         count.setText(""+counter); repaint();
      }
   }
}
a. Explain the connection between the method header on line 19 and the class header on line 5.
b. What is the type of the JLabel constructor parameter that appears in the constructor call on line 9?
c. What does “this” refer to on line 14?
d. On line 16, provide the code that registers ButtonCounter objects as listener for the clicker JButton.
e. Why can’t you use a different name for the actionPerformed method on line 19, say doAction, for example?
f. What class does the getSource() method (line 20) belong to?
g. Describe, in a few sentences, what ButtonCounter is intended to do when the clicker button is clicked multiple times.
int value = 0;
for (int j = big; j > 0; j = j/2)
    value++;
System.out.println(value);

Rewrite this as a while loop
public class Bowl{
    double weight; // kilograms
    boolean empty;
    String origin; // where made -

    public Bowl(double w, boolean e, String origin){
        weight = w;
        empty = e;
        this.origin = origin;
    }
    public double getWeight() { return weight;}
    public boolean getEmpty(){return empty;}
    public String getOrigin(){ return origin;}

    public String toString(){
        return("bowl from " + origin + " weight: " + weight);  
    }
}
Write a `setEmpty` method that allows you to set the empty attribute to true or false.

Explain how `toString` could possibly work properly, since the return type is `String`, but the weight instance variable is part of the return expression.

Write a statement that would appear in a driver class that would create a Bowl object called `myBowl` that weighs .65 Kg, is empty, and comes from Italy.

Add a method to the Bowl class called `majorityEmpty`, which is passed an array of Bowl objects, and returns true if strictly more than half of the bowls in the array are empty.
Write a one class application that reads a single integer from the keyboard using a Scanner object. If that value is negative, program should print “negative”; otherwise program should print the square root of the number.

-------------------

What does the import statement do?
Use inheritance to extend the Bowl class to a new class called OvenProofBowl. This class should add one attribute to the base class, the boolean attribute “ovenproof”. Be sure to include in your class declaration

1) a constructor that takes four parameters;

2) get and set methods for the ovenproof attribute; and

3) a version of toString that includes an embedded call to toString from the base class.
Which of these is legal?

a. int a = 5; double r = 4.3; a = a/r;

b. int a = 5; double r = 4.3; a = (int)a/r;

c. int a = 5; double r = 4.3; a = a/(int)r;
public class BankDriver{
    public static void main(String[] args){
        int pennies = 143; int nickels = 94;
        int dimes = 11; int quarters = 44;
        PiggyBank b = new PiggyBank(pennies,nickels,dimes,quarters);
        System.out.println("decimal fraction of coins that are quarters");
        System.out.println(b.quarterFraction());
        System.out.println("total dollar value,in dollars,cents");
        System.out.println(b.totalValue());
    }
}

// give PiggyBank class definition