CS 121 - Intro to Java - Midterm II review

Announcements

Program 7 now up. Two programs possible
Ch 10 OWL problems up, due next Wed

MidTerm Exam: Friday - Monday, this weekend

Special arrangements? Let me hear from you!
What’s on the exam? It’s ch 1-9 - Obvious emphasis on material from ch 5 - 9:

Arrays
Inheritance
Randomness
The while loop
Mulitple constructors
Recursion (light)
Some sample problems..
int value = 0;

for (int j = big; j > 0; j = j/2)
    value++;
System.out.println(value);

1. Show what this loop does when big is 20? 0?
2. Rewrite as a while loop
3. Are there values of big for which this will run forever?
int j = big;

while(j >0)
{
    j = j/2;
    value++;
}

System.out.println(value);
public class Bowl{
    private double weight;
    private boolean empty;
    private String origin;

    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("from "+origin+" weight: "+ weight);
    }
}
Write a `setEmpty` method that allows you to set the empty attribute to true or false
public void setEmpty(boolean newVal) {
    empty = newVal;
}
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 and would create a Bowl object called `myBowl` that weighs .65 Kg, is empty, and comes from Italy.
The + op is overloaded: if one of the ops is a String, it always builds strings. So ("baa"+.65) is a String

Bowl myBowl = new Bowl(.65,true, "Italy");
Add a method to the Bowl class called `heavy`, which is passed an array of Bowl objects, and returns the origin of the heaviest bowl in the array.
public static String heavy(Bowl[] bowls){
    double bigW = 0.0;
    Bowl bigB = null;
    for(Bowl b : bowls){
        if (b.weight > bigW)
            {bigW = b.weight; bigB = b;}
    }
    return (bigB.origin);
}

String s = Bowl.heavy(myBowls); // static call
but
String t = myBowl.getOrigin(); // non-static call
public static String heavy(Bowl[] bowls){
    Bowl bigB = bowls[0];
    double bigW = bowls[0].weight;
    int pos = 1;
    while(pos < bowls.length){
        b = bowls[pos];
        if (b.weight > bigW)
            {bigW = b.weight; bigB = b;}
        pos++;
    }
    return (bigB.origin);
}
Variants:

Add a method to the Bowl class called `heavy2`, which is passed an array of Bowl objects, and which prints to the console the origin of the heaviest bowl in the array.

Add a method to the Bowl class called `aveWt`, which is passed an array of Bowl objects, and which returns the average wt of the bowls in the array.
public static void heavy2(Bowl[] bowls){
    double bigW = 0.0;
    Bowl bigB = null;
    for(Bowl b : bowls){
        if (b.weight > bigW)
            {bigW = b.weight; bigB = b;}
    }
    System.out.println(bigB.origin);
}
public static double AveWt(Bowl[] bowls){
    double total = 0.0;
    for(Bowl b : bowls){
        total = total + b.weight;
    }
    return (total/bowls.length);
}

(what if bowls in empty??)

Ans: start with
    if (bowls.length == 0) return 0.0; else...
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.
public class OvenPfBowl extends Bowl{
    private boolean ovenproof;
    public OvenPfBowl(double w; boolean e; String s;
                       boolean pf){
        super(w,e,s);
        ovenproof = pf;
    }
    public String toString(){
        String s = super.toString();
        return(s + " ovenproof? " + ovenproof);
    }
}
Note: if default is - all bowls are ovenproof, then we can add a second constructor:

```java
public OvenPfBowl(double w; boolean e; String s){
    super(w,e,s);
    ovenproof = true;
}
```
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;`
d. `int a = 5; double r = 4.3; a = (int)(a/r);`
Suppose mystery is defined in class SomeClass

```java
public static int mystery(int[] a, int j){
    if (j < 0) return 1;
    else return(a[j]*mystery(a,j-1));
}
```

Given this array definition:
```java
int[] b = {2,4,6};
```
What does this method call return:

```
SomeClass.mystery(b,2);
```

Then explain in a sentence or two what mystery does in general.
SomeClass.mystery(b,2) \[ b = \{2, 4, 6\} \]

-> 6*mystery(b,1)

Why - plug in to code:

```java
public static int mystery(int[] a, int j){
    if (j < 0) return 1;
    else
        return(a[j]*mystery(a,j-1));
}
```
SomeClass.mystery(b,2)  \[ b = \{2, 4, 6\} \]

\[-\rightarrow 6\times\text{mystery}(b,1)\]

\[-\rightarrow 6\times(4\times\text{mystery}(b,0))\]

\[-\rightarrow 6\times(4\times(2\times(\text{mystery}(b,-1))))\]

non-recursive, = to 1