Counter and ModNCounter (Fall 2004 Midterm 1 Problem 1)

Tags: loops, objects, inheritance

Part a.

Given the following two classes:

```
public class Counter {
     protected int myCount;
     public Counter() {
          myCount = 0;
     }
     public void increment() {
          myCount++;
     }
     public void reset() {
          myCount = 0;
     }
     public int value() {
          return myCount;
     }
     public String toString() {
          return "" + value();
     }
}

public class ModNCounter extends Counter {
     private int myN;
     public ModNCounter(int n) {
          myN = n;
     }
     public int value() {
          return myCount % myN;
     }
}
```

Now, consider the following program segment:

```
ModNCounter ctr = new ModNCounter(2);
for(int k = 1; k < 5; k ++) {
     ctr.increment();
}
System.out.println(ctr);
```

What gets printed? Briefly explain your answer.

Part b.

Provide a definition for a class named SeasonProgression that is derived from class ModNCounter via inheritance. A SeasonProgression object will essentially act like a mod 4 counter. A call to its toString method returns one of the strings “spring”, “summer”, “autumn”, or “winter”; the string cycles from “winter” to “spring”. The program segment below should produce the indicated output.

program segment

```
SeasonProgression s = new SeasonProgression();
s.reset();
System.out.println(s);
for(int k = 1; k <= 5; k++) {
    s.increment();
    System.out.println(s);
}
```

desired output

```
spring
summer
autumn
winter
spring
summer
```

Don’t define any new instance variables in your SeasonProgression class, and rely as much as possible on the methods of ModNCounter.

ANSWERS: DO NOT READ UNTIL YOU HAVE ATTEMPTED THE PROBLEM ABOVE.

Part a.

The given program segment prints 1. First the loop iterates five times incrementing the inherited myCount variable to 5. The call to println results in a call to toString, which calls the value method. The ModNCounter object uses the Counter toString method. However, it uses the ModNCounter value method, even though toString appears only in the Counter class, because a ModNCounter will always use its own version of an overridden method.

It is important to understand which method gets inherited and which method does not.

Part b.