Unit 5 Practice Test

This test is worth 25 points:
   12 points for the multiple-choice questions (2 points each)
   8 points for question 7, and 5 points for question 8
You may specify up to two answers for each multiple-choice question. You will get one point for any problem in which your first choice is wrong but your second choice is correct.

1st 2nd

_____ 1. Which of the following is a false statement about constructors in Java?

   A. A constructor's header does not include a return type.
   B. When a client invokes a constructor, it must use the keyword new.
   C. You must define a constructor for every blueprint class.
   D. A constructor initializes the instance variables of the object.
   E. Constructors can be overloaded, just as ordinary methods can.

_____ 2. You decide that you want to add a method to the Rectangle class from lecture. You want the method to be a mutator that doubles the width of a Rectangle object. Which of the following is the most appropriate header for that method?

   A. public static int doubleWidth(int width)
   B. public static void doubleWidth()
   C. public int doubleWidth(int width)
   D. public void doubleWidth(int width)
   E. public void doubleWidth()

_____ 3. What is the output of the following Java code fragment?

   ```java
   Rectangle r1 = new Rectangle(0, 0, 5, 10);
   Rectangle r2 = new Rectangle(0, 0, 5, 10);
   Rectangle r3 = r2;
   System.out.println((r1 == r2) + " " + (r2 == r3));
   ```

   A. true true
   B. false true
   C. true false
   D. false false
   E. none of the above

_____ 4. Which of the following (if any) is an invalid statement using our vehicle classes?
   (Assume that appropriate parameters have been specified in place of each ...)

   A. Object o1 = new Vehicle(...);
   B. TractorTrailer t1 = new Truck(...);
   C. Vehicle v1 = new Taxi(...);
   D. Automobile a1 = new Taxi(...);
   E. none of these (i.e., all of these statements are valid)
Questions 5 and 6 are based on the following Java classes:

```java
public class A extends B {
    public void twist() {
        this.shout();
        System.out.print("ho! ");
    }
    public void shout() {
        System.out.print("hey! ");
    }
}
public class B {
    public void twist() {
        System.out.print("ouch! ");
        this.shout();
    }
    public void shout() {
        System.out.print("oh! ");
    }
}
public class C extends A {
    public void gasp() {
        System.out.print("why? ");
    }
    public void shout() {
        System.out.print("whoa! ");
    }
}
```

5. Consider the following statements, all of which are valid:

```java
B b1 = new A();
B b2 = new C();
A a1 = new C();
```

Which of the following method calls is not valid?

A. b1.twist();
B. b1.shout();
C. a1.twist();
D. a1.shout();
E. a1.gasp();

6. Given the object b2 from the previous question, what is the output of the following statements?

```java
b2.twist();
b2.shout();
```

A. whoa! ho! whoa!
B. hey! ho! whoa!
C. hey! ho! hey!
D. ouch! oh! oh!
E. none of the above
7. Consider a class called `Battery` that serves as a blueprint for objects representing a battery for a cell phone or other similar device. Each `Battery` object keeps track of (1) the percentage of charge remaining in the battery (an integer) and (2) whether the battery is being charged (`true` if it is, and `false` if it is not).

Write only the five components of the class that we have requested below. **You should assume that all of the other methods of the class are being written by someone else. You should not write them, and you should not call them.**

Make sure that your class employs proper encapsulation to prevent clients of the class from putting a `Battery` object into an invalid state. In particular, **you should not allow a `Battery` to have a negative charge; rather, you should throw an `IllegalArgumentException` to prevent this from happening.** You only need to worry about this in the context of the code that that you write below.

a) Define the fields of the object. Choose appropriate names.

b) Write a constructor that takes an initial value for the charge remaining and assumes that the battery is not being charged.

c) Write an accessor method for whether the battery is being charged. Choose an appropriate name.
d) Write a method called decrementCharge that decreases the charge remaining in the battery by 1. For example, if the charge remaining is 50, calling this method should reduce it to 49.

e) Write a toString() method that can be used when printing a Battery object. It should correctly override the inherited toString() method, and it should allow clients to see both the charge remaining and whether the Battery is being charged. For example:

- if b1 is a Battery with a remaining charge of 70 that is *not* being charged, printing b1 should display the following:
  70 percent charged (not being charged)
- if b2 is a Battery with a remaining charge of 95 that *is* being charged, printing b2 should display the following:
  95 percent charged (being charged)
8. Write a static method called `processBattery()` that is a client of your `Battery` class from the previous problem. `processBattery()` should take a `Battery` object as a parameter and do the following:

- Display the `Battery`'s current state in the format discussed in problem 7e.
- Determine if the `Battery` is being charged, and
  - if the `Battery` is being charged, the method should return `true`
  - if the `Battery` isn't being charged, the method should decrement the charge remaining in the battery three times (for a total reduction of 3) and then return `false`.

In writing this method, you should take full advantage of the methods that you wrote in problem 7. In addition, you should assume that this new method is not part of the `Battery` class.