Code-Writing Quiz 5 Prep

The goal of code-writing quizzes is to help students gain fluency and confidence in their code-writing.

During the quiz students will be given 1 – 3 very short problems and about 15 min. to implement the solution with a pencil/pen on a piece of paper. Then each student will be randomly paired with a peer and the peers will assess each-other's work. All the errors must be marked with a red pen and the quiz will be graded according to a simple rubric. The rubric will be provided. Instructor will collect all the quizzes to quickly check on the works and to post grades in the gradebook.

Students are expected to write syntactically and logically correct Java code. Import statements and main() method heading will not be needed though.

The quiz is closed-book, no-computer-access type of exercise. Students are encouraged to prepare notes on a 3x5 inch index card using both sides. That card is the only reference students can use during the quiz and eventually during the midterm and final exams. It is necessary to spend time practicing for the quiz. It's close to impossible to do well on such a quiz without prior practice. Problems provided below are very similar to the one(s) that will be appearing on the quiz.

If the student missed the quiz and wants to get credit for it, he/she must submit to the instructor hand-written code solutions to all the sample problems listed in the quiz preparation guide.

Topics to Review

- Writing class constructors – no-argument constructors and constructors that take parameter
- Writing class definitions for classes with fields of non-primitive type
- Accessor and mutator methods
- Accessor method for reference field (not of primitive type)
- Mutator methods that throw exceptions
- Writing copy constructor that creates deep copy of non-primitive fields
- Writing equals() method
- Writing toString() method
- STATIC methods (make sure to know the limitations here!)
- STATIC fields

Sample Problems

Given a class described below:

```
public class Rectangle {
   private double length;
   private double width;
```

```
   public Rectangle(double len , double wid) {/*implemented*/}public
   Rectangle(){/*implemented*/}

   public Rectangle(Rectangle obj) {/*implemented*/}

   boolean equals(Rectangle) {/*implemented*/}

   public String toString(){/*implemented*/}

   /* accessor and mutator methods implemented*/

}
```

Write a class MyClass that has 2 fields: Rectangle r and int aField. Implement the following methods / fields to the new class:

1. Accessor for field r. Make a deep copy of the field when returning it.
2. Constructor MyClass(int a, Rectangle b) that sets both fields. Make a deep copy of object b when copying it!
3. No-argument constructor that sets aField to 0, and r field to a rectangle object with length and width set to 1.
4. Mutator for aField. The value stored in that field must not be negative. Throw Illegal ArgumentException when the negative value is detected.
5. copy constructor for MyClass
6. toString() method for MyClass
7. equals() method for MyClass
8. A static field to MyClass that counts how many objects of that class were created. Make adjustments to the constructors as needed to support the object counting feature of MyClass.
9. A static accessor method to return the static field