Problem Set 3, Part I

Please start your answer to each problem on a new page, as we have done below!

Problem 1: Our `Rectangle` class revisited

1-1)

```
type of method: mutator
header: public void rotate() {}
```

1-2)

```
type of method: accessor
header: public boolean largerThan(Rectangle rect2) {}
```

1-3)

```
problems in code: Because this class employs appropriate encapsulation, this
call is unable to access the width and height attributes from the rectangle
class directly.

rewritten version:
Rectangle r1 = new Rectangle(60, 80);
System.out.println("r1's height is: " + r1.getHeight());
r1.setWidth(r1.getWidth() + 20);
System.out.println(r1);
```

Problem 2: A class that needs your help

2-1) *Revise the code found below:*

```
public class ValuePair {
    private int a;
    private double b;

    public double product() {
        return this.a * this.b;
    }

    // add the new methods here

    public ValuePair(int a, double b) {
           if (a % 2 == 0){
               aValue = a;
           } else {
               throw new IllegalArgumentException("A must be even.");
           }
           if (b >= 0.0){
               bValue = b;
           } else {
               throw new IllegalArgumentException("B cannot be
negative.");
           }
    }

    public int getA() {
          return this.a;
    }
    public double getB() {
          return this.b;
    }
    public int setA(int value) {
           if (value % 2 == 0){
               this.a = value;
           } else {
               throw new IllegalArgumentException("A must be even.");
           }
    }
    public double setB(double value) {
           if (value >= 0.0){
               this.b = value;
           } else {
               throw new IllegalArgumentException("B must be
positive.");
           }
    }

}
```

The reason this code doesn't compile is because this is a static method, meaning that the method can be called on the class level, but not on an instance of the class. Because it's supposed to be an instance method, or non-static method, the keyword static needs to be removed from the method header.

2-2)

Problem 3: Static vs. non-static

3-1)

type and name of the variable	**static or non-static?**	**purpose of the variable, and why it needs to be static or non-static**
double rawScore	non-static	stores the raw score associated with a given Grade object; needs to be non-static so every Grade object will have its own instance of this variable
String category	non-static	stores the category associated with a given Grade object; needs to be non-static so every Grade object will have its own instance of this variable
int numAssignments	static	stores the number of Grade objects that fall into the category of assignment; needs to be static so the number can be updated and accessible by all Grade objects
int numQuizzes	static	stores the number of Grade objects that fall into the category of quiz; needs to be static so the number can be updated and accessible by all Grade objects
int numExams	static	stores the number of Grade objects that fall into the category of exam; needs to be static so the number can be updated and accessible by all Grade objects

3-2)

a) static or non-static?: non-static

explanation: this method needs to be non-static because the method must have access to the fields of a particular called Grade object.

b) changes it would need to make:

The method would first need to change the non-static category variable of the specific Grade object from quiz to exam. It would then need to lower the numQuizzes variable by one, and raise the numExams variable by one.

3-3)

a) static or non-static?: static

   explanation: This method should be static because it doesn’t need a called object, the parameters the method uses will be passed in directly.

b) example of calling it: Grade.computePercent(30.0, 50.0);

3-4)

a) static or non-static?: non-static

   explanation: This method should be non-static because it needs to be called on an object and access parameters from said object.

b) example of calling it: g.addExtraCredit(12.2);