Problem Set 3, Part I ...

Please try to keep each of the three problems on its own page.

Problem 1: Rectangle class revisited

1-1)

a) This instance method would be a mutator.
b) public void rotate() { … }

1-2)

a) This instance method would be an accessor.
b) public boolean largerThan(Rectangle other) { … }

1-3)

a) The code calls private attributes, height and weight directly without using an accessor method. Also, the code is printing out the object rather than the string representation of the object.

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

Problem 2: A class that needs your help

2-1) Since, the method, product(), is a static method, it can’t access any variable that isn’t also static, which in this case is int a, and double b. We would need to just remove static, so that product() can access int a, and double b.

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

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

    public ValuePair(int a, double b) {
        setA(a);
        setB(b);
    }

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

    public int getA() {
        return this.a;
    }

    public double getB() {
        return this.b;
    }

    public void setA(int num) {
        if (num % 2 == 0)
            this.a = num;
        else
            throw new IllegalArgumentException();
    }

    public void setB(double num) {
        if (this.b >= 0)
            this.b = num;
        else
            throw new IllegalArgumentException();
    }
}
```

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 type of work associated with the Grade object, (whether it be “assignment”, “quiz”, or “exam”; needs to be non-static so every object of Grade will have its own value of this variable
int [] amount	static	stores the amount of “assignment”, “quiz”, and “exam”, where the index 0,1, and 2 stores the respective strings above; needs to be static so it is shared among all objects at this class’s level
int latePenalty	non-static	stores the type amount of late penalty associated with the given Grade object; needs to be non-static so every Grade object will have its own instance of this variable

3-2)

a) **static or non-static?**: non-static
explanation: This method should be non-static because each individual Grade Object might need to change its own individual category. Therefore, the change of its category shouldn’t be shared or static.

b) **changes it would need to make**: this.category = changedCategory; where changedCategory would be the String parameter for the desired change in category.

3-3)

a) **static or non-static?**: static
explanation: This method should be static because every Grade Object would need to access and share this method in order to give a proper average.

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

3-4)

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

explanation: This method should be non-static because each Grade object would have a unique value for adding extra credit. Thus, it shouldn’t be shared among all Grade objects.

b) **example of calling it**: `g.addExtraCredit(2.4);`

Problem 4: Inheritance and polymorphism

4-1) The equals() method that Zoo overrides comes from the Object .equals() method. The Zoo class extends from the Object class, even though it doesn't use the extend keyword.

4-2) String y, int x, int y, int a, String b

4-3)

which println statement?	**which method is called?**	**will the call compile (yes/no?)**	**if the call compiles, which version of the method will be called?**
first one	one()	yes	the Yoo version
second one	two()	yes	the Woo version
third one	three()	no	N/A
fourth one	equals()	yes	the Zoo version
fifth one	toString()	yes	the Woo version

4-4)

```
public double avg() {
      return ((this.t + this.u + this.x + this.y + this.a) / 5.0);
}
```

4-5)

a) This would not be allowed because Woo has no relation with Too, even though they both extend to Zoo. They share the same superclass, but they are different subclasses themselves and are not connected at all.

b) This would be allowed because the subclass is able to invoke a constructor of a superclass.

c) This would be allowed for the same reason as part b. Yoo is a subclass of Zoo and Yoo is allowed to invoke Zoo's constructor.

d) This would not be allowed because Zoo is the superclass of Too and a superclass can't invoke the constructor of the subclass.