**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)**

1. Mutator
2. public void rotate (int weight, int height) {

}

**1-2)**

1. Accessor
2. public boolean largerThan (Rectangle other) {

}

**1-3)**

1. In client code, since the class employs appropriate encapsulation, width and height are private member of the superclass Rectangle. Therefore, we cannot directly access them here.
2. Rectangle r1 = new Rectangle(60, 80);

System.out.println ("r1's height is: " + r1.getheight());

r1.setwidth() = r1.getwidth() + 20;

System.out.println(r1); // print the new dimensions

**Problem 2: A class that needs your help**

**2-1)**

An instance method is a non-static method which is declared inside a class, so inside the class ValuePair, the method product should be a non-static method. We can’t access the instance method and instance variables that inside the class with the help of static method.

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

public class ValuePair {

int a;

double b;

public double product() {

return this.a \* this.b;

}

图形用户界面, 文本

描述已自动生成

}

**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 of the grade object, which is string assignment, string quiz, or string exam; needs to be non-static so each category has its own instance of this variable. |
| int Counter[3] | static | The counter for counting the number of objects in each category type; needs to be static so the counters are created once and then can be used by all grade objects. |

**3-2)**

1. **static or non-static?**: non-static

**explanation:** Because string category is non-static and non-static variables can only be accessed by non-static methods

1. **changes it would need to make**:

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

**3-3)**

1. **static or non-static?**: static

**explanation:** Because computePercent method is only used to compute

two method variables. It doesn’t have any instance variables.

1. **example of calling it**:

double percent = Grade.computePercent(90.0, 100.0);

**3-4)**

1. **static or non-static?**: non-static

**explanation:** Because addExtraCredit method is used to increment

*rawScore* which is a non-static variable and non-static variables can

only be accessed by non-static methods.

1. **example of calling it**: g.addExtraCredit(100.0);

**Problem 4: Inheritance and polymorphism**

**4-1)**

The toString() method that Gee overrides come from Object class. In Java, every class is directly or indirectly a subclass of the Object class. The Object class contains toString() method which returns a string that represent an object. When we try to display an object reference, toString() method will be called if it is not been overwritten. Otherwise the overridden toString() method will be called.

**4-2)**

bar (String field declared in Tee itself)

z (Integer field declared in its parent class Yee)

x (Integer field declared in its parent's parent class Gee)

y (Integer field declared in its parent's parent class Gee)

**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 | equals() | yes | the Tee version |
| third one | foo() | yes | the Gee version |
| fourth one | toString() | yes | the Tee versio |
| fifth one | moo() | no | the Zee version |

**4-4)**

class Too {

int avg() {

return (getA() + t + u)/4;

}

}

**4-5)**

**a)**Wouldn’t be allowed, because Too is not the derived class of Woo

**b)**

Wouldt be allowed, because Woo is the derived class of Zoo

**c)**

Wouldt be allowed, because Yoo is the derived class off Woo and Woo is the derived class of Zoo

**d)**  
Wouldn’t be allowed, because Zoo is not the derived class of Too