Problem Set 3, Part I

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

Problem 1: A class that needs your help

1-1) The method cannot access the fields since static methods need an instance of an object for them to be used. We can fix this by making the method non-static. This is done by omitting the word “static.”

1-2) *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

// Constructor
public ValuePair(int a, double b){
this.a = a;
this.b = b;
}

//Get Methods
public int getA(){
return this.a;
}
public double getB(){
return this.b;
}

//Mutator Methods
public void AssignA(int r){
if (r % 2 != 0) {
 throw new IllegalArgumentException(“Value must be even”);
}
this.a = r;
}
public void AssignB(double r){
if (r < 0.0) {
 throw new IllegalArgumentException(“Value must be greater than 0.0”);
}
this.b = r;
}

}
```

Problem 2: Static vs. non-static

2-1)

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
int numAssignment	static	stores the number of assignments within the Assignment worktype; can be static since we only need the class to keep track of this, not every object instance
int numQuiz	static	stores the number of quizzes within the Quiz worktype; can be static since we only need the class to keep track of this, not every object instance
int numExam	static	stores the number of exams within the Exam worktype; can be static since we only need the class to keep track of this, not every object instance
String workType	non-static	stores the type of work associated with a given Grade object; needs to be non-static so every Grade object will have its own instance of this variable

2-2)

a) **static or non-static?**:Non-static
explanation: It is a mutator method and hence will have to change the variables of several different objects

b) **changes it would need to make**:It would need to subtract 1 from int "numQuiz," add 1 to int "numExam," and change the string "workType" from "quiz" to "exam."

2-3)

a) **static or non-static?**: Static
explanation: It doesn't require any objects to run, just two doubles

example of calling it:
g.computePercent(b); //where "b" is a double

2-4)

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

b) **explanation:** It is a mutator method and hence will have to change the variables of several different objects

c) **example of calling it**:
g.addExtraCredit(a); //where “a” is a double

Problem 3: Inheritance and polymorphism

3-1) Zoo overrides the equals() method that is inherited from the root Java Object class since every class in Java stems from this class.

3-2) Inherited: int a, String b, int x, and int y

Declared: String y

3-3)

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	It won’t compile since the method is in Too
fourth one	equals()	yes	The Zoo version
fifth one	toString()	yes	The Woo version

3-4)

```
public double avg();
return ((this.getA() + this.getB() + this.getT() + this.getU())/4.0);
```

3-5)
a) No, neither Too nor Woo inherit from one another.

b)Yes, since Woo inherits from Zoo, this is just an example of upcasting.

c)Yes, since Woo inherits from Zoo and Yoo inherits from Woo, this is just an example of upcasting (but through 2 levels).

d)No, since Too extends Zoo, this is an example of downcasting which causes a compiler error in Java