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) it does not compile because the class object ValuePair does not have the constructor, which initializes the object. In addition to that, two fields, int a and double b has to be encapsulated to have Accessor and Mutator methods to compile. Lastly need to remove static header to make the method instance method.

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

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

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

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

    // add the new methods here

    //Accessor method
    public  int setA(int a){
        return this.a;
    }

    //Accessor method
    public double setB(double b){
        return this.b;
    }

    //Mutator method
    public void new_setA(int a){
        if(a % 2 != 0){
            Throw new IllegalArgumentException(“a has to be even”);
        }
        This.a = a;
    }

    //Mutator method
    public void new_setB(double b){
        if(b < 0.0){
            throw new IllegalArgumentException(“b had to be greater than or equal to 0.0);
        }
       This.b = b;


}
```

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 latePenalty	non-static	Stores the latepenalty associated with a given Grade object; need to be non-static along with rawScore, so every Grade object will have its own instance of this variable
String assignment	static	It stores assignment category date. It needs to be static because it belongs to class. Not every object uses it, but the Grade class
String quiz	static	It stores quiz category date. It needs to be static because it belongs to class. Not every object uses it, but the Grade class
String exam	static	It stores exam category date. It needs to be static because it belongs to class. Not every object uses it, but the Grade class
Int percentage	non-static	It stores the percentage of each instance(student). The percentage of how they performed should be varied by their grade. So it should be accessed by instance methods. So it needs to be non-static.

2-2)

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

explanation: because the method Category will be used in instance method, so it should be accessed out of class grade. Mutator method will change parameters for the object.

b) **changes it would need to make**: The changes it needs to make to the values of the variables are that it has to call the variables using like this. Address to access the constructor. It would need to decrement the counter.

2-3)

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

Explanation: Depending on what u put for possiblePoints and points Earned, each instance(students) object should be able to return the percentage that follows the correct points. So it should be non-static method

b) **example of calling it**: Grade.computePercent(a,b);

2-4)

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

Explanation: The method should be static because all the instances will get the same extra credit. Therefore, it doesn't have to have different extra credit for each instance of class Grade.

b) **example of calling it**: a.addExtraCredit(b);

Problem 3: Inheritance and polymorphism

3-1) The Zoo class overrides equals() method from Object class. Every Class in java inherits class, if u don't extend specific class then it inherits Object class

3-2) int a, String b, int x, int y, 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	None
fourth one	equals()	yes	the Zoo version
fifth one	toString()	yes	The Woo version

3-4)
```
public double avg(){
        double result = this.getT() + this.getU() + this.getA() +this.getB();
        result = result / 4.0;
        return result;
    }
```

Int a, double b, int t, int u

3-5)

a) No, Object Too inherits Zoo object, neither inherit each other

b) yes Woo extends Zoo, so by the rules of polymorphism it works

c) yes it is allowed because Yoo inherits Woo and Woo inherits Zoo

d) it is not allowed, Too inherits Zoo not the other way, so it should be reverse