Homework 1

CSI 405

Use 1 eclipse project but different packages for the problems below

The packages should be named as follows

com.ualbany.hw1.problemX

Where the X in problemX can be substituted for the problem number you are solving below. Ex. com.ualbany.hw1.problem1

Failure to follow the above guidelines can cause you to lose points

Problem 1: Create an Employee class

Create a class to represent Employee information called *Employee.* This class includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary (type double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a driver class named *EmployeeTest* that demonstrates class *Employee's* capabilities. Create two *Employee* objects and display the yearly salary for each Employee. Then give each Employee a 10% raise and display each Employee's yearly salary again.

Problem 2: Methods

Create a class called *HW1Problem2*. Create the following (public) methods in the class:

1. Write a method called *multiple* that takes two integers as its arguments and returns true if the first integer is divisible evenly by the second one (i.e., there is no remainder after division); otherwise, the method should return false.
2. Write a method called *reminder* that takes an integer as an argument and returns the remainder of that value divided by 7.
3. Write method *distance* to calculate the distance between two points (x1, y1) and (x2, y2). All numbers and return values should be of type double.
4. **BONUS:** Write a method that uses random numbers to simulate 10 flips of a coin.

Incorporate all of the above methods in a driver class *Problem2Driver*. Allow user to provide inputs for the arguments of the methods and also display results from the methods.

Problem 3: Bank simulation

Model the classes below:

1. CheckingBankAccount – A checking bank account has value, a person can withdraw, deposit money to the account
2. Address – This is a simple class which has all String fields to represent an address (ex. Address line 1, line2, City, State, Zip)
3. Person – A person has a first and last name, address and has a CheckingBankAccount

4. Bank – Bank is "associated" with the checking account. Bank has a String name and address. You do not need to have a CheckingBankAccount object within this class. However it is upto you to design/model this representation.

For simplicity we will assume there is only one instance of each classes described above. Ex. Bank only has 1 account, an account is only linked to 1 person and a person only has single address.

The "has-a" relationship here shows aggregation. Example a person "has-an" address - meaning you can use the Address class within the person class to represent the address for a person.

Once you have the above classes create a main method in the Bank class. Simulate a person arriving to the bank (print the person's name, and the bank's name), then simulate the person opening an account with initial deposit of $1000. Simulate a deposit of $1000 and a withdrawal of $500. Every time a withdrawal or deposit happens, a receipt is printed with the name, address of the person and his/her current balance.

Create the UML diagrams of all the classes in **all** of your projects. The UML diagrams should follow the **exact** formatting shown in class slides. Submit UML diagrams in a separate folder as images with your code submission.

Grading

Implementation and DESIGN of classes	40%
Execution	40%
UML diagrams	10%
Code clarity / organization	10%
Bonus & extra points for good design	5%

Submit any specific instructions needed for grading/executing your homework.

Your project(s) should be a 7-zip (or any other zip format) file submitted on blackboard. It should be an eclipse project that the grader can import into his/her eclipse environment and execute. If this criteria is not followed you will lose points.

The UML diagrams should be submitted separately in the same submission as a folder of images well labelled/organized. (Not inside the project archive described above)

Follow exact deadlines, timing on blackboard. Rules for late submission apply as per syllabus.