**CS 2163 Java ---**Homework 4: create OOP class from scratch and draw UML class diagram

**General description for homework 4 : this is an Eclipse project, NOT a Greenfoot project.**

In this homework, you need to first develop a class using OOP principles: data abstraction and encapsulation, and then create an application class to make use of the encapsulated class developed above, then draw the UML class diagram for the OOP class. After finishing this homework, you demonstrate your skill in the development and use of OOP class, and an understanding of UML class diagram. **Finish this homework on Eclipse, NOT Greenfoot.**

**Here is the detail description for homework 4:**

Create a new Java project in Eclipse. This project has two java files (classes): ***House.java*** and ***JohnDoeHw4.java***. Please refer to file “CompileAndRunTheFirstJavaProgram.docx” in Moodle folder “chap 1” on how to create an Eclipse project, and how to create new class file in an Eclipse project.

**First, create a new java file named *House.java***. At the beginning of this file, copy the program header from line 1~11 in file “CorrectSequence.java” to ***House.java***, then change line 1, 3, 4, and line 7~10, to make these lines suitable for file **House.java**. Then right after the program header, please write down the definition line for public class ***House***. Then in the body of this class ***House***, do the following:

1. define two private data member: total price of double type (for example, a value of 168000 indicates that this house is worth 168000 dollars), and total square feet of double type (for example, a value of 2398 indicates that this house has 2398 square feet in total).
2. define two public constructors, the first one is the default constructor with no parameter, and the second one has two formal parameters to initialize the two private data members of this class.
3. for every data member, provide a public getter and a public setter, so there are two getters and two setters in total. Refer to the Loan.java for examples of getter and setter methods.
4. provide a public method named ***calcPricePerSqf*** . This method returns the average price per square foot of the house by dividing the total price by the total square feet. The return type of this method is double. This method does NOT need any formal parameter, because as a public member method, this method can directly access the private data members of the same class.
5. provide a public method named ***calcPropertyTax***. This method returns a double type value, which is the property tax of this house. This method needs only ONE formal parameter, which is a double type variable named ***taxPercentage***. The property tax of the house is calculated by multiplying the total price of the house (a private data member of the class) with the tax percentage (the formal parameter of this method). For example, if the tax percentage is 1.25 percent for a house of 160000 dollars, then the property tax is 160000 \* 1.25 \* 0.01 = 2000 dollars. Notice that the 0.01 in the above formula is to convert the 1.25 percent to the real value. **Attention**: ***taxPercentage*** is a formal parameter of this method, and it is **NOT** a data member of the ***House*** class, therefore, do not make the mistake of setting ***taxPercentage*** as a private data member of the ***House*** class, and if you do that, you are in the wrong track!!!!

**Secondly,** in the same Eclipse project you created earlier for this homework, **new a second java file named** ***JohnDoeHw4.java***, which has the application class ***JohnDoeHw4*** (replace ***JohnDoe*** with your first and last name). At the beginning of ***JohnDoeHw4.java***, copy the program header from file “CorrectSequence.java” (line 1~11) to ***JohnDoeHw4.java*** , then modify line 1, 3, 4, and line 7~10 to make them suitable for ***JohnDoeHw4.java***. This class has only one method name ***main***, then provide a line like this as the first line inside the main method, which outputs your name, course meeting time, class and semester info:

System.out.println("Spring 16, Java, Wedn 5:30pm, John Doe, hw4");

Then after this first line inside the main method, do the following:

1. create two House objects using the constructor with two parameters, plug in the data below as the actual parameters of the constructor. Each line below contains the two actual parameters for one object:

295000.0 3395.0

189000.0 1675.0

, where the 1st column is total price, and the 2nd column is total square feet. Refer to **TestLoanClass.java**, and see how the ***myLoan*** object is created.

1. create the third House object using the default constructor. Then ask the user to input the two parameters with a Scanner type variable. Then use the setters to set the user input values into the third object object. No data type or range checking of the input parameters is needed. Refer to **TestLoanClass.java**, and see how the ***hisLoan*** object is created and then set to the desire parameters using setters.
2. ask the user to input a tax percentage. For example, a user input of 1.25 means 1.25%. This property tax value will be used in the next step to calculate the property taxes for all three house objects.
3. For each individual House object, invoke two methods: ***calcPricePerSqf*** () and ***calcPropertyTax***() and store the results in two local variables, respectively. For method ***calcPropertyTax***, you need to plug in the tax percentage variable obtained in step 3 above. In summary, you have 6 variables that store the results for three house objects.
4. For each house object, output its total square feet, total price, average price per square feet, and the property tax. If the output information exceeds 80 columns, you need to output them in multiple lines, with each line less than 80 columns. Use System.out.printf() method, and format the total square feet with one digits after the decimal point, and the other three output items with two digits after the decimal point (because they are money values). Assume that the total square feet in each house object is less than 10000 square feet, and the total price of each house is less than 1000000 dollars. Hint: %8.1f can be used for one digit of precision output, and %8.2f can be used for two digits of precision output. Refer to link for explanation of printf method. <http://alvinalexander.com/programming/printf-format-cheat-sheet> , and looat at section with title “Printf – floating point numbers”.

**Thirdly, use Visual Paradigm for UML, Community Edition** to create an UML class diagram for class **House.java**, and in this diagram, all data members and all method members in class **House.java** should appear in the class diagram. Refer to document “***DownloadAndUseVisualParadigmUML.docx***” on how to create UML class diagram with Visual Paradigm for UML. Attention: create one UML class diagram ONLY for class **House**, and no need to create UML class diagram for class JohnDoeHw4**.**

After the UML class diagram is created in Visual Paradigm, export the UML class diagram to a jpg file, and name it ***House.jpg***.

**What and where to submit:** you need to ***zip*** these three files together, and name the zip file ***JohnDoeHw4.zip***:

* ***House.java***
* ***JohnDoeHw4.java***
* ***House.jpg*** (this is the exported image file of the UML class diagram for class House)

Submit this zip file to Moodle homework 4 drop box. After submission, be sure to follow the **VERIFICATION** process: download your .zip file to a local folder in your computer, unzip the zip file and compile then run the Java program in Eclipse. If the verification process works correctly, then you have submitted the .zip file successfully in Moodle; if not, you need to fix the problem and re-submit. **DO NOT zip any eclipse project folder into your submission zip file**. If the submitted java files program does not compile or does not run, your score will be zero.

**How to zip multiple files**:

* First you need to copy the java source code files ***House.java*** and ***JohnDoeHw4.java*** from the Eclipse project folder “src” into a different folder, such as ***c:\work*** folder.
* Then copy ***House.jpg*** filde into this c:\work folder also.
* Then inside this c:\work folder, hold the **ctrl** key in the keyboard, then **left-click** your mouse to select the three required files, and then **right-click** any one of the selected files to show the context menu, and then zip the selected files from the context menu. In windows, the context menu item is “Send to 🡪 compressed (zipped) folder” ; in Mac, the context menu item is “Compress Items”.
* If the generated zip file name is not ***JohnDoeHw4.zip***, then you need to change the zip file name to the required name before submitting the zip file.

Click the Moodle homework drop box to see the due day of each homework assignment.

**Implementation requirements:**

* No need to use array in this homework.
* File ***House.java*** contains only one class named ***House***, and file ***JohnDoeHw4.java*** contains only one class named ***JohnDoeHw4***. File ***House.java*** and file ***JohnDoeHw4.java*** are two different java files.
* In file ***House.java***, you need to have two private data members, and all methods are public, including: two constructors; one setter and one getter for each private data member; effector methods include: method ***calcPricePerSqf*** () and method ***calcPropertyTax***().
* When coding **House.java**, please refer to file “**CorrectSequence.java**” for the correct sequence of components in an encapsulated class. Class **Loan.java** follows the correct sequence strictly, which also serves as a reference.
* In file ***JohnDoeHw4.java***, you have only one method: the main method.
* The signature and return type of the methods in these two classes have already been defined in the above homework descriptions, and you cannot change the signature and the return type of any method.
* Add suitable comments in the source code.
* Please read the above implementation details carefully **BEFORE** start coding***.***
* As a general reference, read the source code files: **Loan.java** and **TestLoanClass.java** (you get these two files after you unzip file **UML-classDiagram.zip**) on how to code class with OOP principle and how to code application class that uses OOP class.
* After finishing coding, you need to debug your program and test it multiple times with different input parameters supplied for each test run.
* Don’t forget the correct program header needs to presented in every Java source code file, as in line 1~ 11 of file **Loan.java** and file **TestLoanClass.java**.

This assignment has **20 points. Grading components is listed in the table below:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **House class created correctly** | **main method implemented correctly** | **Correct UMLclass diagram** | **Code indent & align** | **program header &**  **comment** | **Java naming convention** | **Submit file format** | **Program logic** |
| **4** | **4** | **3** | **3** | **2** | **1** | **1** | **2** |

The grading components are, but not limited to: code alignment and indentation, variable/method/class naming convention, programmer header, suitable comments, submitted file format, overall program logic.

For Eclipse java file, if there is still any red circle check mark in java source code, you will receive zero point, because red circle check mark in Eclipse IDE means the java source code still has compilation error. You have to fix this compilation error first, before you proceed to the next phase of running java program. Even in some rare scenarios, you can run the program in Eclipse with compilation error, but a compilation error in java source code will result in zero point for the homework.