CPSC 2151

Lab 10

Due: Friday November 16th at 11:59 pm

In this lab you will be working with the Model View Controller design pattern. While the Model View Controller Design Pattern will be very useful for working with Graphical User interfaces, we will not be working with a GUI for this assignment. We will be building off of the code from Lab 7, specifically the Mortgage and Customer class.

Instructions

You will need to create a new project with a package called called cpsc2150.mortgages. Add your Mortgages.java and Customer.java classes to that package. You will then need 3 more classes:

MortgageApp.java

MortgageApp will be a very simple class. It will contain our main function that will be the entry point of the program. All it will do is declare an instance of MortgageView and MortgageController, and then call MortageController.run(). The code should look like:

```
package cpsc2150.mortgages;

public class MortgageApp {
    public static void main(String [] args)
    {
        MortgageView view = new MortgageView();
        MortgageController controller = new MortgageController(view);
        controller.run();
    }
}
```

This type of set up is common when using MVC architectural pattern in Java. We just need the entry point to set up our Controller and View, then turn over control of the program to the Controller.

MortgageView.java

MortgageView will serve as the View layer of our MVC architecture. That means that Mortgage View is the only class that can get input from the user or print to the screen. The Controller layer will use MortgageView to perform these tasks. Mortgage View also does not know about our Model layer, so it cannot perform any input validation on whether or not the given values meet the preconditions of our Mortgage or Customer classes. MortgageView can still validate whether a “Y” or “N” was entered for yes or no prompts, since that is a requirement of the View itself, not the Model.

The only class level variable that MortgageView should have is a scanner object. Remember, we can have issues if we have multiple scanner objects looking at the same input source, so we don’t want to declare one in each function.

MortgageView should have the following public methods available:

- A constructor
- double getHouseCost()

- double getDownPayment()
- int getYears()
- double getMonthlyDebt()
- double getYearlyIncome()
- int getCreditScore()
- String getName()
- void printToUser(String message)
    - prints the message to the user
- boolean getAnotherMortgage()
    - asks the user if they want to apply for another mortgage for the current customer
    - prompts for Y/N
    - checks to make sure user answered, Y, y, N, or n. If not will prompt again
    - returns true if the user wants to apply for another mortgage, false if they don't
- boolean getAnotherCustomer()
    - asks the user if they want to consider another customer
    - prompts for Y/N
    - checks to make sure user answered, Y, y, N, or n. If not will prompt again
    - returns true if the user wants to consider another customer, false if they don't

  The "get" methods all ask the user for the specified value. They do not perform any data validation (unless specified) as the view does not have access to the model layer. We can assume the user will input the correct datatype (i.e. a number when a number is expected).

MortgageController.java

The MortgageController class serves as the Controller layer in our MVC architecture. It controls the flow of control in our program. This class is able to use both the view and the model layers to accomplish this task. This class will check to make sure the data that the user provides (through the View layer) meets the preconditions that exist in our model layer. If the input is not valid, the controller will (through the view) alert the user to the error and reprompt the user for correct input. The MortgageController class will also use the model layer (Mortgage and Customer class) to actually apply for the mortgage. After a mortgage has been applied for, the Controller will (through the View layer) display the Customer and Mortgage information to the User.

The controller will allow for a Customer to apply for multiple loans (without having to e-enter the customer information), and allow for the user to enter information about multiple customers. Remember, the Controller layer cannot directly interact with the user, it has to go through the View layer.

The MortgageController class will have one private field, a MortgageView Object, which is set by an argument passed into the constructor. The Mortgage Controller class will have one public method, called run() which runs the program.

All of the logic concerning validating the Models preconditions, and the order of events, is handled by the Controller layer. The logic concerning the Mortgage application or the Customer itself are handled by the Model Layer.

TIPS and additional Requirements

- You must provide a make file with make and make run targets
- You do not need to provide any automated testing, although you should test your program.
- You do not need to provide any contracts for MortgageView, MortgageController, or MortgageApp. Mortgage and Customer should have contracts from a previous assignment.
- You do need to comment your code
- You need to follow the Model View Controller architectural pattern for this assignment.
    - All interaction with the user goes through the View class. View does not have access to the Model
    - Controller handles input validation, and the order of events of the program. The controller has access to the view and the model, so it is the go between for those two layers
    - The model handles our entity objects and the "lower level" logic. It does not have any access to the other two layers.
- Follow our best practices: No magic Numbers, information hiding, separation of concerns, etc.

Groups

You may, but are not required to, work with a partner on this lab. Your partner must be in the same lab section as you, not just the same lecture section. If you work with a partner, only one person should submit the assignment. You should put the names of both partners in a comment at the top of the file in order for both partners to get credit. This assignment may take more than just the lab time. Make sure you are able to meet outside of class to work on the assignment before you decide to work with someone else. Remember to actively collaborate and communicate with your partner. Trying to just divide up the work evenly will be problematic.

Before Submitting

You need to make sure your code will run on Unix. You must provide a makefile and use the appropriate file structure. Your TA should be able to unzip your submission, type "make" and "make run" to run your program.

Submitting your file

You will submit your file using handin in the lab section you are enrolled in. If you are unfamiliar with handin, more information is available at https://handin.cs.clemson.edu/help/students/

Sample input and outputs:

```
What's your name?
Jim Halpert
How much is your yearly income?
-12000
Income must be greater than 0.
How much is your yearly income?
56000
```

```
How much are your monthly debt payments?
-247
Debt must be greater than or equal to 0.
How much are your monthly debt payments?
247
What is your credit score?
-800
Credit Score must be greater than 0 and less than 850
What is your credit score?
1000
Credit Score must be greater than 0 and less than 850
What is your credit score?
748
How much does the house cost?
-120000
Cost must be greater than 0.
How much does the house cost?
120000
How much is the down payment?
-10000
Down Payment must be greater than 0 and less than the cost of the
house.
How much is the down payment?
130000
Down Payment must be greater than 0 and less than the cost of the
house.
How much is the down payment?
13000
How many years?
-15
Years must be greater than 0.
How many years?
30
Name: Jim Halpert
Income: $56000.0
Credit Score: 748
Monthly Debt: $247.0
Mortgage info:
Principal Amount: $107000.0
Interest Rate: 4.5%
Term: 30 years
Monthly Payment: $542.1532815136978

Would you like to apply for another mortgage? Y/N
y
How much does the house cost?
125999
How much is the down payment?
```

```
12000
How many years?
15
Name: Jim Halpert
Income: $56000.0
Credit Score: 748
Monthly Debt: $247.0
Mortgage info:
Principal Amount: $113999.0
Interest Rate: 4.0%
Term: 15 years
Monthly Payment: $843.2368383152977

Would you like to apply for another mortgage? Y/N
n
Would you like to consider another customer? Y/N
y
What's your name?
Dwight Schrute
How much is your yearly income?
73000
How much are your monthly debt payments?
...
```