

CIS 494 – Business Systems Development with Java – Spring 2015

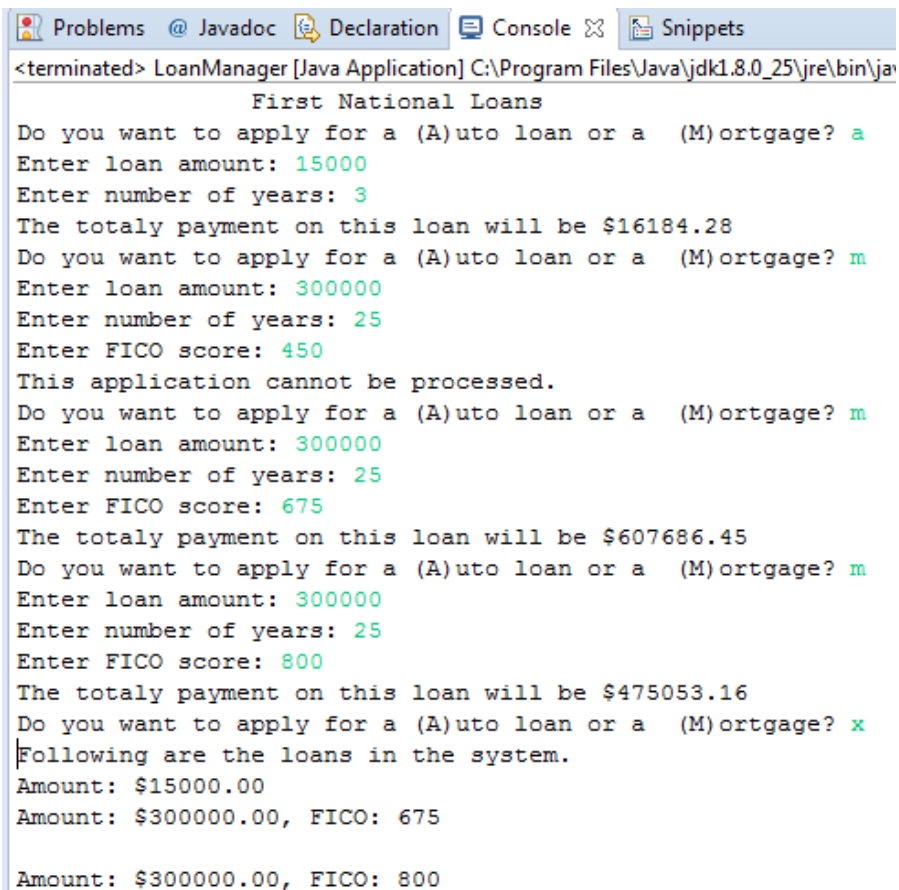
Assignment 2

Due on Blackboard: Friday, February 13 by 11:59 PM

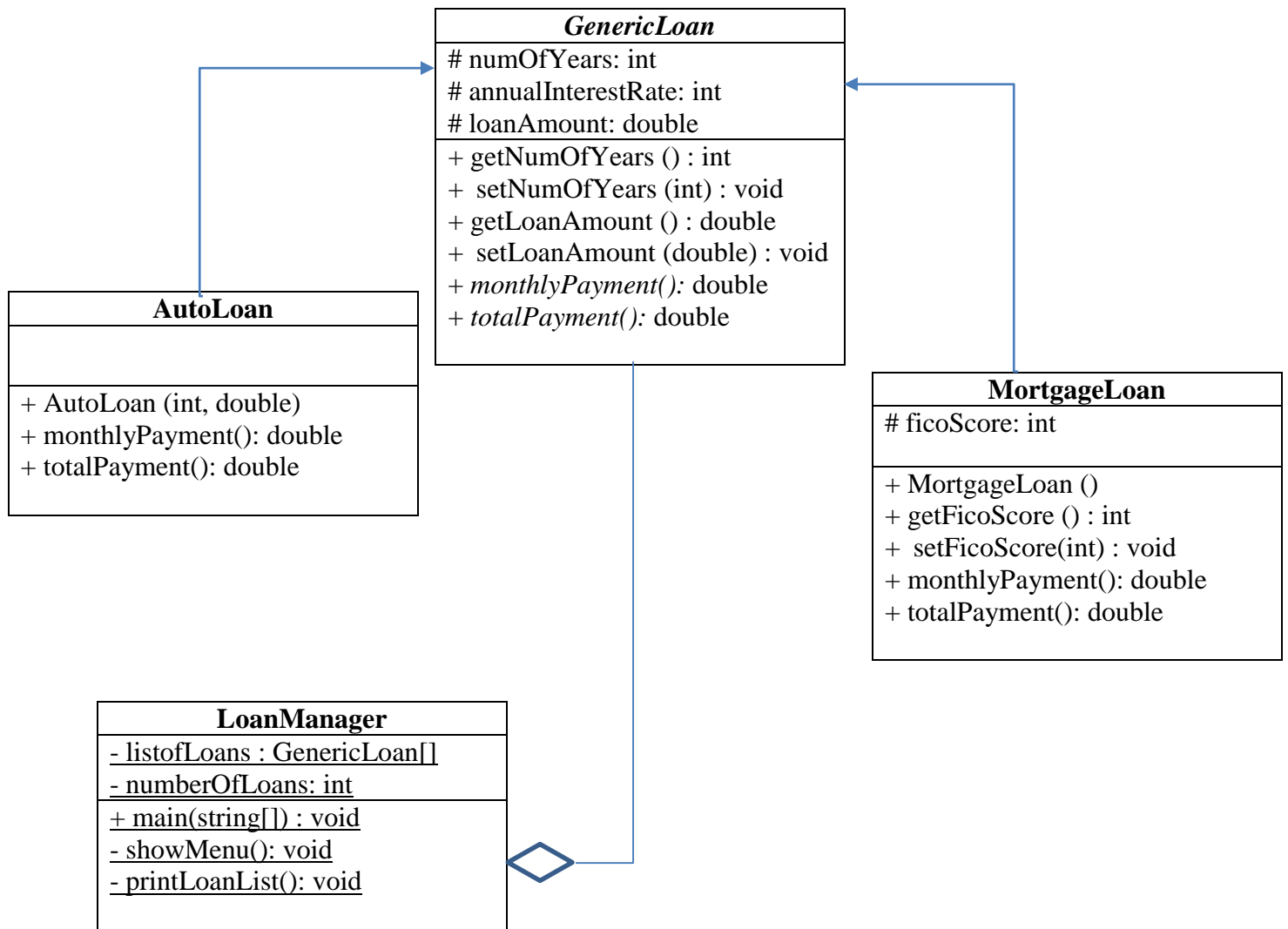
Skills Developed: Implementing classes, abstract classes, inheritance, polymorphism.

Implement a simple loan calculator that processes either auto loans or mortgages. The interest rates for auto loans are constant. However, mortgages use a FICO score. No one should qualify for a mortgage with a score on 500. Create classes to store information for both type of loans.

Sample Output



```
<terminated> LoanManager [Java Application] C:\Program Files\Java\jdk1.8.0_25\jre\bin\ja
First National Loans
Do you want to apply for a (A)uto loan or a (M)ortgage? a
Enter loan amount: 15000
Enter number of years: 3
The totaly payment on this loan will be $16184.28
Do you want to apply for a (A)uto loan or a (M)ortgage? m
Enter loan amount: 300000
Enter number of years: 25
Enter FICO score: 450
This application cannot be processed.
Do you want to apply for a (A)uto loan or a (M)ortgage? m
Enter loan amount: 300000
Enter number of years: 25
Enter FICO score: 675
The totaly payment on this loan will be $607686.45
Do you want to apply for a (A)uto loan or a (M)ortgage? m
Enter loan amount: 300000
Enter number of years: 25
Enter FICO score: 800
The totaly payment on this loan will be $475053.16
Do you want to apply for a (A)uto loan or a (M)ortgage? x
Following are the loans in the system.
Amount: $15000.00
Amount: $300000.00, FICO: 675
Amount: $300000.00, FICO: 800
```



UML Diagram

Class Descriptions

1) GenericLoan

This class should be an abstract class with abstract methods to get monthly payment and total payment.

2) AutoLoan

AutoLoan inherits GenericLoan.

Constructor: Provide a constructor that accepts the number of years and the loan amount, which are used to set the attributes. Use the interest rate of 5.0 and set it within the constructor. All auto loans have the same interest rate as a standard.

Override and implement the abstract methods. Note the following for computation:

Monthly Payment: You will need the month interest rate. Divide the annual interest rate by 1200 and use the resulting double value as the monthly interest rate.

Use the following formula to return the monthly payment:

```
loanAmount * monthlyInterestRate / (1 -  
    (Math.pow(1 / (1 + monthlyInterestRate), numOfYears * 12)));
```

Total Payment: Multiply the monthly payment by 12*number of years to get the total payment.

3) MortgageLoan

MortgageLoan inherits GenericLoan and implements the additional attribute for FICO score.

Constructor: provide a no-args constructor that takes zero arguments. Set the FICO score to default to 600 within the constructor.

Override and implement the abstract methods. Note the following for computation:

Monthly Payment:

- If the FICO score is below 700, set the annual interest rate to 6.5. Otherwise, set the annual interest rate to 4.0.
- You will need the month interest rate. Divide the annual interest rate by 1200 and use the resulting double value as the monthly interest rate.

Use the following formula to return the monthly payment:

```
loanAmount * monthlyInterestRate / (1 -  
    (Math.pow(1 / (1 + monthlyInterestRate), numOfYears * 12)));
```

Total Payment

Multiply the monthly payment by 12*number of years to get the total payment.

4) LoanManager

Main: Initialize your private variables. Allocate 10 elements to your array of GenericLoans. Call your showMenu Method.

4.1) ShowMenu

ShowMenu should loop asking for input till the user presses “X”.

Within this method, ask the user if they want to apply for an auto loan or a mortgage. Read their choice and process it using a switch statement. Since all loans require year and amount, ask the user for them and store them in appropriate variables that you declare locally.

Case “M”:

Ask for and store FICO score.

If the FICO score is under 500, output an error message and do not process the loan.

Create an instance of Mortgage loan and store it locally. Set the amount, year, and FICO score. Add the instance of MortgageLoan to your array and increment your count.

Tell the users what the total payment will be.

Case “P”:

Create an instance of AutoLoan and store it locally. Set the amount and year.

Add the instance of AutoLoan to your array and increment your count.

Tell the users what the total payment will be.

Before exiting the program because the user pressed “X”, call your printLoanList() method to print out a list of all Loans.

printLoanList() Method

Loop through all your loans and print out the amount to the console screen. If the loan is for a mortgage, print out the FICO score for that loan as well. Use the keyword “instanceof” seen in In-class 3 to test if a loan is a mortgage. You will need to use downcasting / polymorphism.

Additional Note on Assignment

Bear in mind that this Assignment closely follows the two exercises in object-oriented design and polymorphism that were done in class – In-Class2 and In-Class3. Refer to those for help in working with objects, array of objects, abstract classes, and polymorphism.

Submitting Files

Submission should be made using a zip file that contains the entire Eclipse project folder. You will need to **zip the entire project folder**. The folder will automatically contain the class source files as well as the compiled .class files.

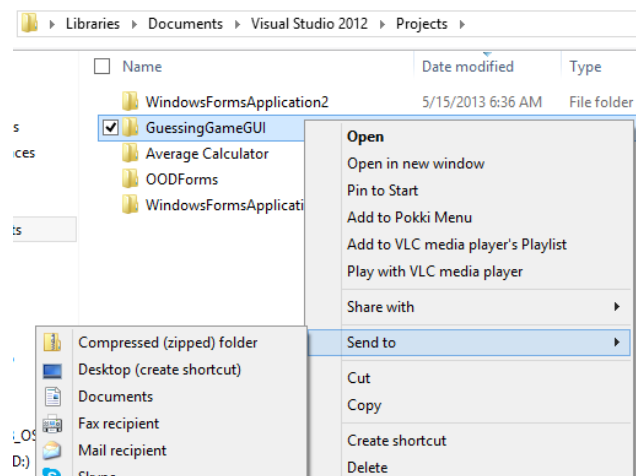
ZIP file should be named: **AX.zip or IC.zip**

where X is the in-class assignment number, e.g., A1.zip is the submission file for Assignment 1 and IC1.zip is the submission file for InClass1.

Note: The ZIP filename is independent of the Project name. Do not name your project A1.

How to Properly ZIP and Submit Your Project Files:

- Go to the folder within {Eclipse Workspace}\
- ZIP the entire top-level folder for your project by right-clicking your project folder and selecting Send to | Compressed (zipped) folder.
- Finally, submit the ZIP file using the submission link on Blackboard by the due-date and time listed on the assignment. Upload the ZIP file.



Using built-in windows zip tools: <http://windows.microsoft.com/en-us/windows/compress-uncompress-files-zip-files>

Verify your files BEFORE and AFTER submission:

- Check for actual class files being present in the folder before you zip it.
- ***Ensure that you are not zipping a short-cut to the folder.***
- After zipping, check file size. A file size under 4K likely does not contain all the files.
- Unzip, extract all files, and verify you see actual files, not a solitary short-cut.
- Uncompress your zip file before submitting and verify that files are present.
- Make sure you have submitted your file and not just saved a draft on Blackboard. ***A blue clock indicates a submission in progress, i.e. a draft, not a submission. The draft is accessible only to you. You will get a ZERO if you only ever save a draft on Blackboard and never submit your files.***
- Download your zip file after submitting, uncompress, and again verify that your files are present. Test your files in Visual Studio after uncompressing.

This takes an extra couple of minutes. Please do it if your grade is important to you. If you do this, you will not end up submitting a bad file. If you submit an empty file, or one containing only a shortcut, or a bad zip file, or a bad project file, you will receive a score of zero and your only recourse will be to do the makeup assignment at the end of the semester.