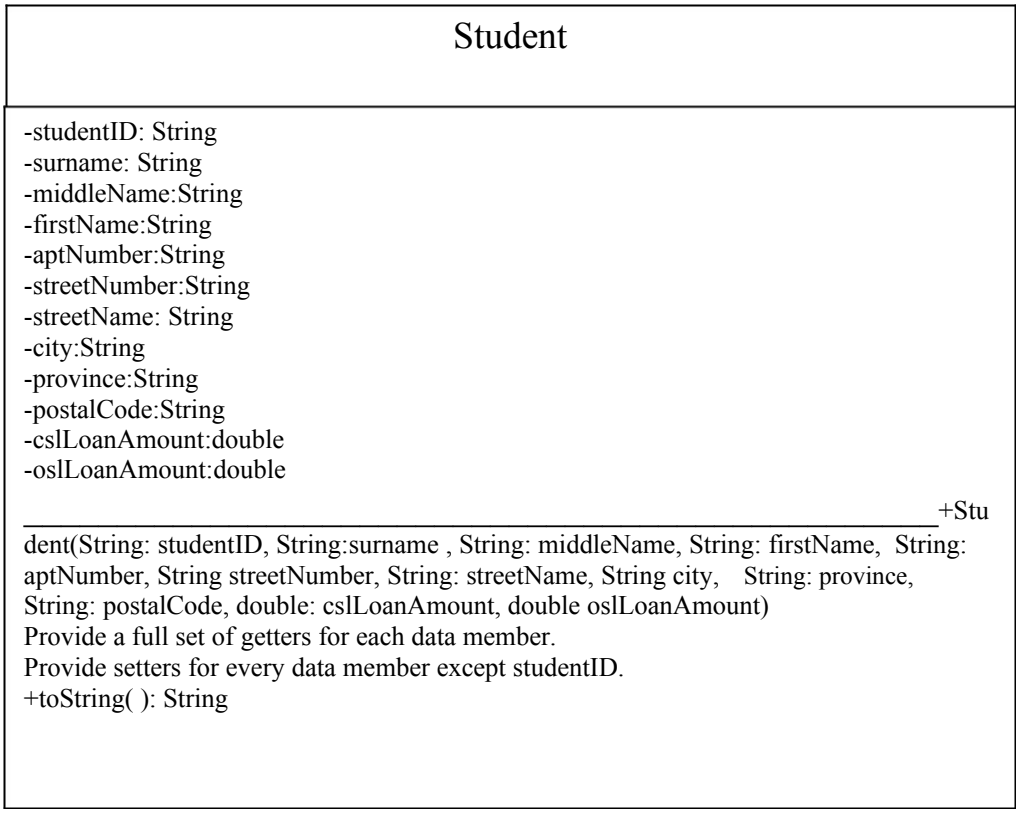
Course:	**INFO 3134 S2019**
Professor:	**Madhavi Mohan**
Assignment:	**Project 2: Student Loan GUI App ver. 1.0**
Due Date:	**Submit zipped source files to the FOL drop box by Thursday August 15, 2019, by 2000 (8 PM).**

Description

You have gotten your first co-op placement with the Student Financial Aid office at Fanshawe College. The manager would like to eventually provide an app that students could download onto their phone which would, among other functions, allow them to see what various re-payment schemes would cost in terms of the interest that the student would pay.
NOTE: the current Ontario Student Assistance Program is being modified by the current provincial government, so actual repayment rules are in a state of flux. This app will follow fictional repayment rules created for this project only.

Classes Needed

1) Write a Student class based on the UML diagram shown below.



The data member studentID is a String consisting of seven numeric characters. The leading character can be a zero, as in 0087901.
The data members **cslLoanAmount** and **oslLoanAmount** will hold the Canada Student Loan and Ontario Student Loan portions of the overall loan amount awarded to the student.
The toString() method will return a String that contains the data shown just below:

```
Student Name: Pulling, William John
Student Number: 1111111
```

```
CSL Amount is $4500.0
OSL Amount is $3200.0
```

Next, write a StudentLoanApp class that extends JFrame.This class will have as a data member an ArrayList<Student>, to which Student objects can be added or removed as required.
Your name and student number must be displayed in the title bar of the JFrame. Also, a label stating "This is [*your name*]'s Student Loan Calculator must appear as the top component inside the frame.

This class will also be responsible for building the GUI. It will have one or more named inner classes that will be responsible for handling events from the GUI depending on what GUI components you decide to use.

Next, write an interface called *Your_Intitials*_LoanPayable.java that will hold a constant value called ANNUAL_RATE_ TO_ MONTHLY_RATE. Assign this constant the decimal equivalent value of 1/1200. You can then use this to convert whatever annual prime interest rate the user enters to the equivalents monthly rate decimal equivalent.

Your interface will also have an abstract method called *calculateLoanPayment( ).* This method will return a double value that represents the loan payment amount, and it will accept three arguments: a double representing the OSL or CSL principal amount, a double representing the annual prime interest rate, and an int value that represents the amortization period in months.

Input Form

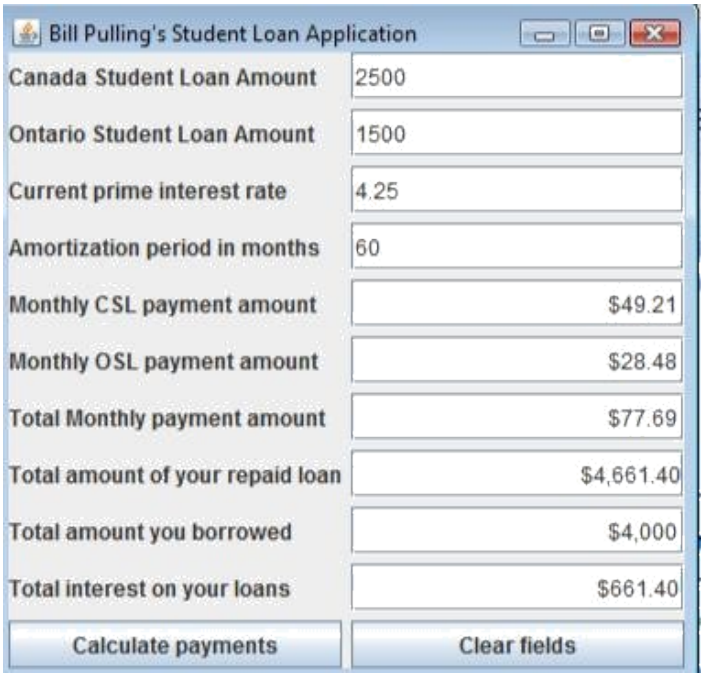
You need to create a form for the user to enter their personal data and the amounts of their loans. Include some data validation for the student number so that only numeric characters are entered here. This data will then be used to create a Student object, which will be entered into the arrayList. Look for ways to minimize what the user has to type in.

Repayment Calculation Form

You need to create a form that will display all of the student's data by retrieving the Student object from the arrayList and using its getter methods to put data onto the form. The form should have a control so that you can press a button to get to the next student or go back to the previous student.
The form should open and show the data for the first student in the arrayList, including the
- amount in dollars of his/her Canada Student Loan (CSL)
- amount in dollars of his/her Ontario Student Loan (OSL)

There needs to be a textfield or some other type of control where the user can enter the current prime interest rate as an ***annual*** **percentage** i.e 4.25% per year. NOTE: interest rates will measured in quarter percent increments, so an interest rate of 4.25% is valid, but a rate of 4.35% would not be valid.
There needs to be a field or some other type of control where the user can enter the amortization period in months, which is the number of months over which repayment will be made. If the repayment was to be spread over five years, then the amortization period in months will be (5 * 12) = 60 months.

The program will then calculate and display the monthly payments required to pay back both parts of the loan, the combined monthly total payments, the total amount that will be repaid with interest, the original amount borrowed, and the total amount of interest that is paid. The screen shot on the next page is from a very early prototype that does not show the complete student data and is a very basic layout intended to show you only some test data. (***We expect more from you than this***! ***We want to see all of the data stored in the Student object on the form).***



The button labeled “Calculate payments” causes your program to read the contents of the relevant fields, do the necessary calculations, and then display the calculation results in the appropriate output fields.
The button “Clear fields” will simply erase any text in the prime interest rate field, and the amortization period field so the user can do another set of calculations. Have the cursor go back to the prime interest field when this button is pushed.

Formula for Calculating the Monthly Payments…

The interest rate on a Canada Student Loan is prime rate **plus** 2.5% (in other words if the primeinterest rate is 4.25%, the interest rate on a CSL is (4.25 + 2.5) = 6.75%.

The interest on an Ontario Student Loan is prime rate **plus** 1.0% (in other words if the prime interest rate is 4.25%, the interest rate on an OSL is (4.25 + 1.0) = 5.25%.

The formula for calculating the monthly payment ***P*** on a loan of amount ***A*** with a *monthly interest rate* of ***i*** over an amortization period of ***N*** months is:

$$P = A * i * (1 + i)_N / ((1 + i)_N – 1)$$

IMPORTANT! The above formula uses a *monthly interest rate expressed as a decimal equivalent rate*. **To obtain this**, **you need to multiply the annual interest rate by 1/1200.**

Now, multiplying the annual rate by 1/1200 is mathematically equivalent to first dividing the entered **annual prime interest rate** by 12 to convert the annual interest rate to a monthly interest rate, then dividing this monthly interest rate by 100 to convert from it a *percentage* to its *decimal equivalent,* which is what we need to use in the actual formula shown above.

Now, to accomplish this, you will code an interface (*interface*…not a GUI!) called ***Your_Intitials_LoanInterface.java*** that will hold a constant value called ANNUAL_RATE_TO_MONTHLY_RATE. Assign this constant the decimal equivalent value of 1/1200. You can then use this to convert whatever **annual prime interest rate** the user enters to the equivalents **monthly rate decimal equivalent**.

This interface will also have an abstract method called ***calculateLoanPayment( )*** . This method will return a double value that represents the loan payment amount, and it will accept three arguments: a double representing the OSL or CSL principal amount, a double representing the annual prime interest rate, and an int value that represents the amortization period in months.

Event Handling: please do all of your event handling in one or more named inner classes. This makes it easier for your teacher to assess your event handling code. One of your inner classes will have to implement your interface method to do the calculations of the monthly loan payment.

Here is a complete sample calculation that uses the sample data shown in the GUI on page 3.

Part 1 – Canada Student Loan (CSL):

A = $2500.00	The loan amount is $2500.
i = 0.005625	The prime interest rate is 4.25%. The CSL uses an interest rate of *prime plus 2.5, which* would be 6.75%. To convert this annual rate to a *monthly interest rate* expressed as a decimal we multiply 6.75 * 1/1200 to get a monthly rate of 0.005625.
N=60	The amortization period is 60 months.

Step by step calculations…

$P = A * i * (1 + i)_N /((1 + i)_N – 1)$
$= 2500 * 0.005625 * (1 + 0.005625)_{60} / ((1 + 0.005625)_{60} – 1)$
$= 2500 * 0.005625 * (1.40011493) / ((0.40011493))$
$= 49.20865$
= $49.21 per month (rounded)

Part 2 – Ontario Student Loan (OSL):

A = $1500.00	The loan amount is $1500.
i = 0.004375	The OSL uses an interest rate of *prime plus 1.0*, which is 5.25%. To convert this to a monthly interest rate expressed as a decimal this would be 5.25 * 1/1200 or 0.004375.
N=60	The amortization period is 60 months.

$P = A * i * (1 + i)_N /((1 + i)_N – 1)$
$= 1500 * 0.004375 *(1 + 0.004375)_{60} / ((1 + 0.004375)_{60} – 1)$
$= 1500 * 0.004375 * (1.299432266) / (0.299432266)$
$= 28.47897576$
= $28.48 per month (rounded)

Total monthly payment is the sum of the two individual loan payment amounts.

Therefore the monthly payments are $49.21 for the CSL and $28.48 for the OSL for a combined monthly payment of $77.69 on an amortization period of 60 months.

Methods

You must implement the abstract method ***calculateLoanPayment( )***.that is declared in your interface. It will calculate a monthly payment according to the formula given above. This method should accept three arguments: the loan amount, *the annual interest rate that is being charged for the particular loan type* (remember, you pay prime + 1.0 on the Ontario student loan amount, and prime + 2.5 for the Canadian student loan amount) , and the amortization period (number of months).

The method will return the monthly payment in dollars and cents ***rounded to the nearest cent*** to the line of code that called the method. Note that you will call this method twice during your calculations, once to calculate the monthly payment for the CSL and again to calculate the monthly payment for the Ontario Student Loan amount.

Exception Handling:

Create your own custom exception class called *Your_Intitials_NegativeValueException*.java and implement it in the appropriate method using try and catch blocks.

The idea here is that if the user enters a negative value in either of the two input fields, then this exception should be thrown and then caught in a try-catch structure. The catch block should have code that will pop up a JOptionPane warning message box that advises the user that they cannot enter a negative value in the input fields. The message should then say that it will convert the negative value to a positive one and continue with the calculations. When the user clicks on the OK button of the message box, the application will do exactly that…convert the negative value to a positive one and continue with the calculations.

Submission:

1. Zip up all of your .java files into one zip file named Your_Name_Project2.zip and submit it to the INFO3134 Project 2 drop box by Thursday, August 15, 2019, by 2000 hours (8 PM).
2. LATE PENALTIES: If you miss the submission deadline, the following late penalties will be applied.

   Up to 24 hours, the late penalty will be 20%.

   Over 24 hours and up to 48 hours late, penalty is an additional 30% for total of

   50%. After 48 hours, a mark of zero will be assigned.

Marks Available	What are the Marks Awarded For?:	Marks Awarded
2	Programming style : Each class doc header is complete, all naming conventions followed, good descriptive variable names, appropriate comments on classes and utility methods, Half mark deduction for each violation found.	
2	Student class is correctly coded	
2	Interface containing constant value and abstract method is coded and implemented by inner event handling class	
6	StudentLoanApp class GUI class contains all of the necessary elements and presents them in a unique, clear, and uncluttered manner.	
1	User is able to scroll through the students in the arrayList and see their data displayed on the form	
2	Event handling code is functional in all respects	
2	Calculations are done correctly and all calculated amounts are displayed to two decimal places.	
3	Custom exception class is written to throw an exception if a negative value is input. Exception should be caught and user is advised by JOptionPane pop up message dialog box.	
20	TOTAL MARKS	