Fabrice Nguegang (R11782740)

Leslie Omonzane (R11756088)

Ayobami Ajala (R11720649

Project 38: Loan Calculator

Object Oriented Programming

April 24th, 2023, 9:30PM

**Introduction:**

This program is that of a Loan calculator which calculates monthly loan payments and total interest amounts from a loan principal amount, an interest rate, and a loan term (in years) obtained from the user. This program consists of a loan interface, a loan super class, a loan calculator subclass, and a test class to test user input.

A list of known bugs include:

* Inconsistent rounding bug: The **calculateMonthlyPayment()** and **calculateTotalInterest()** methods use different methods to round the result to two decimal places. **calculateMonthlyPayment()** uses **Math.round()**, while **calculateTotalInterest()** uses **DecimalFormat**. This could result in inconsistencies in the values calculated.

**Implementors documentation:**

UML diagram:

Diagram

Description automatically generated

There are two classes. Loan and LoanCalculator, and one interface: LoanInterface. The Loan class contains private instance variables for the loan amount, interest rate, and term, it also contains public methods for getting these values, and for calculating the monthly payment and total interest. The LoanCalculator class extends Loan (that is, it is a subclass of Loan) and implements the LoanInterface and contains methods for calculating the monthly payment and total interest of a loan. The LoanInterface contains the method signatures for calculateMonthlyPayment() and calculateTotalInterest(), which must be implemented by any class that implements the interface. This program also traps exceptions by implementing IllegalArgumentException

The formula used to calculate monthly payments and total interest are:



and

Where A= monthly payment amount.

I = total interest amount

P = principal amount

r = monthly interest rate

n = term in months

**User documentation:**

The user is expected to enter a loan amount, an interest rate in decimal corresponding to the percentage (If 8% enter 0.08 )and the output is expected to be a monthly payment value and a total interest amount. The user is expected to enter values that are all greater than zero. If the user enters any value that is less than or equal to zero, a message displaying “Invalid Loan Data” appears. The program keeps running until the user enters the sentinel string “quit” to quit the program.

Test data and outputs:

* Amount = 500000 , Interest Rate = 3%, Term = 15 years

Text

Description automatically generated

* Amount = 10000 , Interest Rate = 10%, Term = 5 years

Text

Description automatically generated

* Amount = 159000, Interest Rate = 6%, Term = 10 years

Text

Description automatically generated

* Amount = 123900, Interest Rate = 1%, Term = 30 years

Text

Description automatically generated

* Amount = 0 , Interest Rate = 9% , Term = 3 years

Graphical user interface, text, application

Description automatically generated

* Amount = 10000, Interest Rate = -8%, Term = 3 years

Graphical user interface, text

Description automatically generated with medium confidence

* Amount = 10000, Interest Rate = 8% , Term = -4 years

Graphical user interface, text, application

Description automatically generated with medium confidence

**Appendix:**

**Source code:**

Loan interface:

// Define an interface to calculate loan payments and interest

**public** **interface** LoanInterface {

**double** calculateMonthlyPayment();

**double** calculateTotalInterest();

}

Loan class:

**public** **class** Loan **implements** LoanInterface {

// Private instance variables to store loan data

**private** **double** amount; //principal amount

**private** **double** interestRate; //interest rate percentage

**private** **int** term; //Loan term in years

// Constructor to create a Loan object with given loan data

**public** Loan(**double** amount, **double** interestRate, **int** term) {

**this**.amount = amount;

**this**.interestRate = interestRate;

**this**.term = term;

}

// Getter methods to access the private instance variables

**public** **double** getAmount() {

**return** amount;

}

**public** **double** getInterestRate() {

**return** interestRate;

}

**public** **int** getTerm() {

**return** term;

}

// Implement the calculateMonthlyPayment() method from the LoanInterface

@Override

**public** **double** calculateMonthlyPayment() {

// Throw an exception if any of the loan data is invalid

**if** (getAmount() <= 0 || getInterestRate() <= 0 || getTerm() <= 0) {

**throw** **new** IllegalArgumentException("Invalid loan data");

}

// Calculate the monthly payment using the loan data

**double** monthlyInterestRate = getInterestRate() / 12;

**int** termInMonths = getTerm() \* 12;

**double** numerator = monthlyInterestRate \* Math.*pow*(1 + monthlyInterestRate, termInMonths);

**double** denominator = Math.*pow*(1 + monthlyInterestRate, termInMonths) - 1;

**double** monthlyPayment = getAmount() \* (numerator / denominator);

**return** monthlyPayment;

}

// Implement the calculateTotalInterest() method from the LoanInterface

@Override

**public** **double** calculateTotalInterest() {

// Calculate the total interest using the monthly payment and loan data

**double** monthlyPayment = calculateMonthlyPayment();

**double** totalInterest = (monthlyPayment \* getTerm() \* 12) - getAmount();

**return** totalInterest;

}

}

LoanCalculator class:

// Define a LoanCalculator class that extends the Loan class

**public** **class** LoanCalculator **extends** Loan **implements** LoanInterface {

**public** LoanCalculator(**double** amount, **double** interestRate, **int** term) {

**super**(amount, interestRate, term);

}

}

Test class:

**import** java.util.Scanner;

**public** **class** Test{

**public** **static** **void** main(String[] args) {

**while** (**true**) {

**try** {

Scanner input = **new** Scanner(System.***in***);

**double** interestrate;

**int** term;

//ask user to enter Amount

System.***out***.print("\nEnter Loan Principal amount (or quit to quit program) :\n ");

//read amount

String in = input.nextLine();

// Check if the user wants to quit

**if** (in.equalsIgnoreCase("quit")) {

**break**;

}

**double** amount = Double.*parseDouble*(in);

//ask user to enter Interest Rate

System.***out***.print("Enter Interest rate in decimal:\n ");

//read interest rate

interestrate = input.nextDouble();

//ask user to enter term in years

System.***out***.print("Enter Term in years:\n ");

//read interest rate

term = input.nextInt();

// Create a LoanCalculator object with given loan data

LoanInterface loan = **new** LoanCalculator(amount, interestrate, term);

// Calculate the monthly payment and total interest using the LoanCalculator object

**double** monthlyPayment = loan.calculateMonthlyPayment();

**double** totalInterest = loan.calculateTotalInterest();

// Print the monthly payment and total interest

System.***out***.printf("Monthly payment: $%.2f\n", monthlyPayment);

System.***out***.printf("Total interest: $%.2f\n", totalInterest);

} **catch** (IllegalArgumentException e) {

// Handle the exception if the loan data is invalid

System.***out***.println(e.getMessage());

}

}

}

}