EECS1022 – Lab 5: Classes & Objects

Fall 2025 **Due Date: Sunday, November 9(11:59pm)**

Overview

Welcome to Lab 5!

This graded lab assignment focuses on object-oriented programming fundamentals, including class design, encapsulation, constructors, accessor and mutator methods, and computational methods. Students will implement a Loan class based on a provided UML diagram to demonstrate proficiency with object instantiation, method invocation, and proper object-oriented design principles.

Lab Policies

- **Academic Integrity**: Submit your own work. Do not copy code from classmates or online sources. All violations will be reported as academic misconduct.
- **Submission Format**: Submit only the file: **Loan.java**. Do not upload ZIP files or project folders. Do not change the name of the file as this will not allow us to test your code.
- **Deadline:** Submit your .java file to the eClass course page by **Sunday, November 9 (11:59pm).** No late submissions are accepted. Email submissions are not accepted.
- Your lab assignment is not graded during the weekly lab sessions scheduled.

Learning Outcomes

By the end of this lab, you will be able to:

- Import a starter project archive file.
- Design and implement a class based on UML specifications
- Create instance variables with appropriate data types and access modifiers
- Implement constructors (both default and parameterized)
- Write accessor (getter) methods that return instance variable values
- Write mutator (setter) methods that modify instance variable values
- Implement computational methods that perform calculations using instance data
- Apply encapsulation principles using private instance variables
- Use the java.util.Date class for date handling
- Use the given JUnit tests (calling the utility methods) to guide the development.

EECS1022 – Lab 5: Classes & Objects

Fall 2025 **Due Date: Sunday, November 9(11:59pm)**

Lab Requirements

- For the JUnit test class LoanTest.java given to you:
  - Do not modify the test methods given to you.
  - You are allowed to add new test methods.
- For each method in the Loan class that you are assigned to implement:
  - Javadoc has been generated for you. You can check the documentation of the project in your browser by clicking on index.html file.
  - No System.out.println statements should appear in each of the utility method.
  - No Scanner operations (e.g., input.nextInt()) should appear in each of the utility methods. Instead, refer to the input parameters of the method.
  - No main method.
  - NO RECURSION - No recursive method calls
  - Don't change the method signatures provided to you. Otherwise, your test cases will fail.
  - Follow the UML diagram specifications exactly
  - All instance variables must be declared as private
  - Implement all constructors as specified
  - Implement all getter and setter methods
  - Implement all computational methods
  - Complete the partial code in Loan.java file to achieve desired solution for the listed programming tasks.
  - A tester file LoanTest.java is provided to you with some of the test cases implemented.

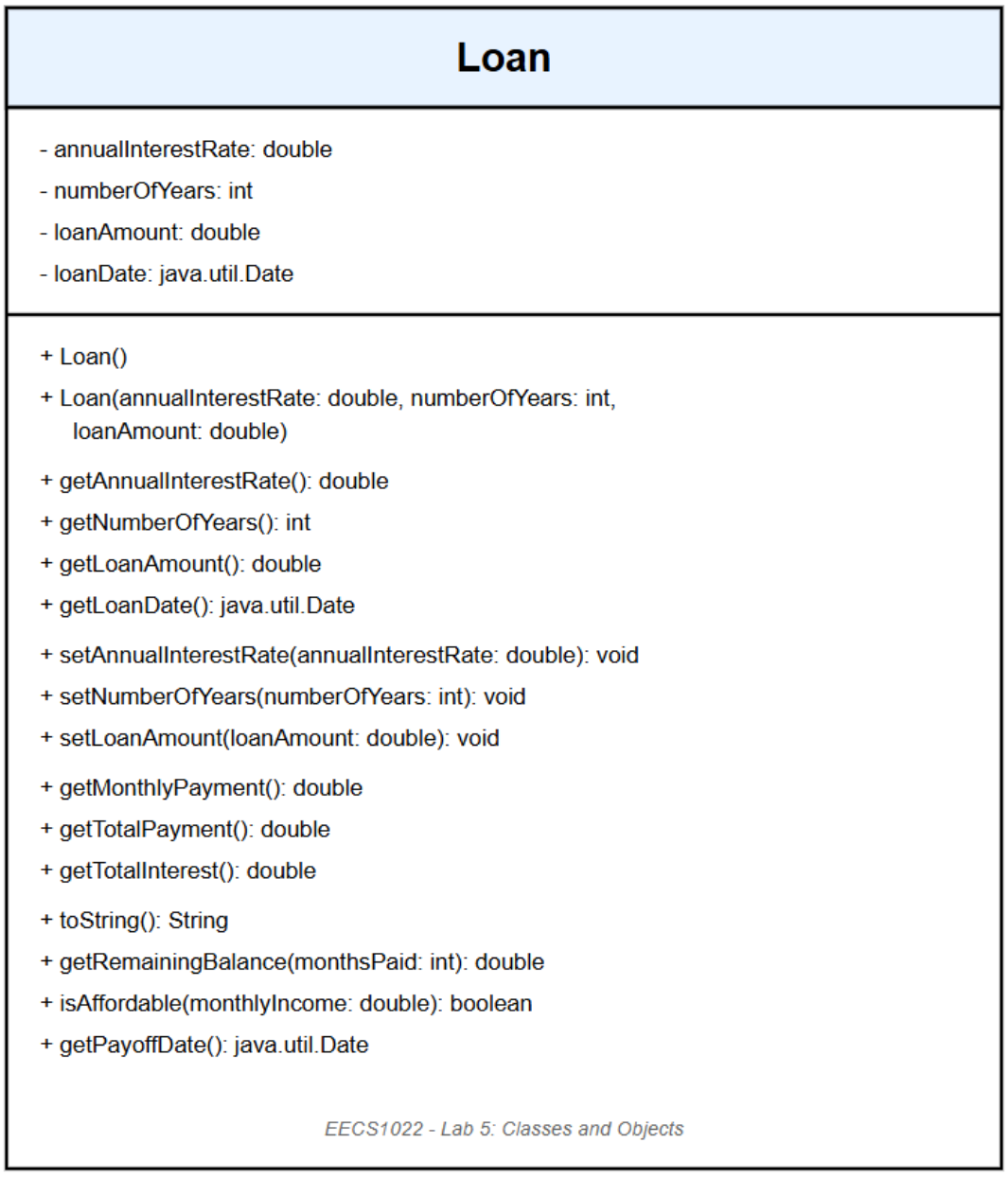
Download and Import the Starter Project

- Download the Eclipse Java project archive file from eClass: EECS1022_Lab5.zip
- Launch Eclipse and browse to EECS1022-workspace (for instance or your own created workspace).
- In Eclipse:
  - Choose File->Import
  - Under General, choose Existing Projects into workspace

Fall 2025 **Due Date: Sunday, November 9(11:59pm)**

- Choose Select archive file. Browse your compressed zip folder and attach it.
- Make sure that the EECS1022_Lab5 box is checked under Projects and you don't have the same project already in the workspace. Then Finish.

UML Diagram:



EECS1022 – Lab 5: Classes & Objects

Fall 2025 **Due Date: Sunday, November 9(11:59pm)**

Programming Tasks

You are required to implement the following tasks in the Loan.java class

Task 1: Define instance variables and Implement constructors, accessor methods and mutator methods – Get the information of them from UML diagram.

Task 2: Computational Methods:

1) **getMonthlyPayment()** - This method calculates and returns the monthly payment for the loan.

   Formula:

   Monthly Payment = (loanAmount × monthlyInterestRate) / (1 - (1 / (1 + monthlyInterestRate)^numberOfMonths))

   Where:

   - monthlyInterestRate = annualInterestRate / 1200 (Divide by 12 for months and by 100 to convert percentage to decimal)
   - numberOfMonths = numberOfYears × 12

   Implementation Notes:

   - If annualInterestRate is 0, return loanAmount / (numberOfYears × 12)
   - Use Math.pow(base, exponent) for exponentiation
   - Round the result to 2 decimal places using: Math.round(value × 100.0) / 100.0

2) **getTotalPayment()-** This method calculates and returns the total payment for the loan.

   Method Signature: public double getTotalPayment()

   Formula:

   Total Payment = getMonthlyPayment() × numberOfYears × 12

   Implementation Notes:

   - Use the getMonthlyPayment() method
   - Round the result to 2 decimal places using: Math.round(value × 100.0) / 100.0

3) **getTotalInterest()-** Calculates and returns the total interest that will be paid over the life of the loan.

   Method Signature: public double getTotalInterest()

   Formula: Total Interest = getTotalPayment() - getLoanAmount()

   Return Value: The total interest amount, rounded to 2 decimal places.

4) **getRemainingBalance(monthsPaid: int)-** Calculates the remaining balance on the loan after a specified number of months have been paid.

Method Signature: public double getRemainingBalance(int monthsPaid)

Formula:

- For loans with interest: Remaining Balance = P * [(1 + r)^n - (1 + r)^p] / [(1 + r)^n - 1]

  Where:

  P = original loan amount

  r = monthly interest rate (annualInterestRate / 1200)

  n = total number of months (numberOfYears * 12)

  p = months paid

- For loans with 0% interest: Remaining Balance = loanAmount - (monthlyPrincipal * monthsPaid) where monthlyPrincipal = loanAmount / totalMonths.

Return Value: The remaining balance, rounded to 2 decimal places

Special Cases:

- If monthsPaid <= 0: return full loan amount
- If monthsPaid >= total months: return 0.0

5) **isAffordable(monthlyIncome: double)-** Determines whether the loan is affordable based on the borrower's monthly income, using the 28% rule (housing payment should not exceed 28% of gross monthly income).

Method Signature: public boolean isAffordable(double monthlyIncome)

Formula:

- maxAffordablePayment = monthlyIncome * 0.28
- isAffordable = (getMonthlyPayment() <= maxAffordablePayment)

Return Value: true if the loan is affordable, false otherwise

6) **getPayoffDate()-**Already Implemented. Calculates and returns the date when the loan will be completely paid off.
7) **toString()-** Already Implemented. But see how it is written. This method will be asked to implement in next labs. It overrides the Object class's toString() method to provide a formatted string representation of the loan object

EECS1022 – Lab 5: Classes & Objects

Fall 2025 **Due Date: Sunday, November 9(11:59pm)**

Submission Instructions

- Submit only one file: **Loan.java**
- Upload it to the Lab 5 assignment on eClass.
- Ensure your file compiles and runs without error.
- Submitting files with compilation errors will receive zero marks. No partial marks.
- Your file will be tested with additional test cases.