

**<22/SP-COP-2800-72035> Java Advanced**

**<Assignment 12-04>**

Document Version: 0.1

Version Date: July 2, 2022

Created By: David Duron

# Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Rationale |
| 0.1 | 2022 JUL 2 | David Duron | Submit Assignment |

# Document Purpose

The purpose of this document is to define the Loan class and discuss how to implement and use it.

# Technical Specifications

## Purpose of Technical Implementation

The purpose of the Loan class is create an object that stores information that must be highly accurate and meet certain requirements. The constructor of the class includes the determination of meeting certain requirements such as: positive loan amount, positive annual interest rate, positive number of years. By including these determination checks in the class constructor, the developer can ensure high accuracy and peace of mind knowing the class constructor is secure against reverse engineering; it can be stated that for further development, there should be implementation to ensure the properties are still meeting certain requirements. These additional determination checks could be implemented in the getter and setter methods. Additionally, there should be implementation of a setter and a new overloaded constructor for when the loan date is prior to the actual creation of the object; this can be useful for digitization of old documents.

## Technical Implementation Components

I supplemented the Loan class. The Loan class begins by creating an object with the following arguments passed: annualInterestRate, numberOfYears, loanAmount. Each of the arguments are checked to ensure that they are greater than zero. Once determined they are greater than zero, they are stored as the object’s properties and an epoch is created and stored in the loanDate property. We have three setters for annualInterestRate, loanAmount, numberOfYears and six getters for each property, total payment and monthly payment. The final two getters are based on calculations using the property values.

**Properties**

1. annualInterestRate: this stores the corresponding value passed in the constructor
2. numberOfYears: this stores the corresponding value passed in the constructor
3. loanAmount: this stores the corresponding value passed in the constructor
4. loanDate: this stores the epoch of when the object was created

**Methods**

1. getAnnualInterestRate(): returns property value of annualInterestRate
2. getNumberOfYears(): returns property value of numberOfYears
3. getLoanAmount(): returns property value of loanAmount
4. getLoanDate(): returns property value of loanDate
5. getMonthlyPayment(): calculates annualInterestRate divided by 1200 and stores as monthlyInterestRate, calculates this.loanAmount times monthlyInterestRate using Math.pow and other operations, returns the value of monthlyPayment as a double.
6. getTotalPayment(): calculates total payment by using the getMonthlyPayment() method and multiplying it times this.numberOfYears() method times twelve. Returns the value as a double.
7. setAnnualInterestRate(double annualInterestRate): uses the argument to calculate it meets validity requirements and sets the property to the new value.
8. setLoanAmount(double loanAmount): uses the argument to calculate it meets validity requirements and sets the property to the new value.
9. setNumberOfYears(int numberOfYears): uses the argument to calculate it meets validity requirements and sets the property to the new value.

**Constructors**

The developer can create an instance of the Loan class one way/

1. Loan loan\_variable = new Loan(double annualInterestRate, int numberOfYears, double loanAmount);

# Technical Implementation Pseudocode

Ask the user to enter the loan’s annual interest rate and save as variable

Ask the user to enter how many full years the loan is and save as variable

Ask the user to enter the loan amount and save as variable

Using a try catch, create an object using the loan class and the new variables as the arguments.

They must be in the following order: annual interest rate, number of years, loan amount.

If the arguments are valid, the object will be created.

If the arguments are invalid, the object will not be created and a message will be displayed.

Once the object is created, use the appropriate methods to display properties.

If any information about the loan changes, use the appropriate methods to set new values of the appropriate property.

End