

A Project Report

on

CONSUMER LOAN ASSISTANT

*Submitted in partial fulfillment of the
requirement for the award of the degree of*

Bachelor of Technology in Computer Science and Engineering



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

**Under The Supervision of
Dr. Michael Raj TF
Professor
Department of Computer Science and Engineering**

Submitted By

18SCSE1010437 - PRACHI KUMARI

18SCSE1010266 - PRAKHAR AGRAWAL

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER- 2021**



**SCHOOL OF COMPUTING SCIENCE AND
ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA**

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled "**CONSUMER LOAN ASSISTANT**" in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Dr. Michael Raj TF, Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1010437 - Prachi Kumari
18SCSE1010266 - Prakhar Agrawal

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor
(Dr. Michael Raj TF
Professor, GU)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **18SCSE1010437 – PRACHI KUMARI, 18SCSE1010266 – PRAKHAR AGRAWAL** has been held on 18-12-2021 and his/her work is recommended for the award of **Bachelor of-Technology in Computer Science and Engineering**.

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Project Coordinator

Signature of Dean

Date: 18-12-21
Place: Greater Noida

ACKNOWLEDGEMENT

This project becomes a reality with the kind support and help of many individual. I would like to extend my sincere thanks to all of them.

It is indeed a great pleasure to express our sincere thanks to our guide Dr. Michael Raj TF, whose valuable guidance and kind supervision given to us throughout the course which shaped the present work as it shows.

He taught us how to write academic papers, had confidence in us when we doubt ourselves, and brought out the good ideas in us. He was always there to meet and talk about our ideas, to proofread and mark up our paper, and to ask us good questions to help us think through our problems. Without his encouragement and constant guidance, we could not have finished this project.

Last but not the least we are thankful to our family whose unfailing love, affection, sincere prayers and best wishes had been a constant source of strength and encouragement. Last, but not least, we thank our parents, for giving us life in the first place, for educating us with aspects from both arts and sciences, for unconditional support and encouragement to pursue our interests. We dedicate this work to our parents who will feel very proud of us. They deserve real credit for getting us this far, and no words can ever repay them.

ABSTRACT

Money-lending is probably the oldest form of business in the rural areas of India. In spite of the spread of the banking network, the village Mahajan continues to be an indispensable source of credit. Often in rural areas we see that people are not that educated to calculate their own interest that they have to pay and these Mahajan's fool them by taking more money from them or in urban areas people don't have enough time to calculate it so this is when they can use our consumer loan assistant and can get a description on how to pay and how much to pay every month until the end of the loan time.

(Ever wonder just) How much those credit card accounts are costing you ? Flight or movie tickets, restaurant bills or groceries, mobile recharge or utilities, you can pay for a wide range of services and products using a credit card.

Loans on Credit Cards are pre-approved loans extended to you based on your Credit Card usage, repayment and history. Talking about the loan repaying system You must repay the loan in easy monthly installments (EMI) over the tenure you have chosen. You can choose a tenure up to 36 months. The EMIs will be added to your credit card statement and will have to be paid on the due date along with payment for any other purchases you may have made. Your credit limit will be reduced to the extent of your EMI amount.

This project will help us to get a handle on consumer debt. The Consumer Loan Assistant Project we would build computes payments and loan terms given balance and interest information. We look at focus traversal among controls, how to do input validation, and the message box for user feedback.

Contents

Title	Page No.
Candidates Declaration	I
Acknowledgement	II
Abstract	III
Contents	IV
List of Table	V
List of Figures	VI
Acronyms	VII
Chapter 1	Introduction
	1.1 Formulation of Problem
	1.2 Tool and Technology Used
Chapter 2	Literature Survey/Project Design
Chapter 3	Functionality/Working of Project
Chapter 4	Results and Discussion
Chapter 5	Conclusion and Future Scope
	5.1 Conclusion
	5.2 Future Scope
	Reference
	Publication/Copyright/Product

List of Table

S.No.	Caption	Page No.
1.	Revised Balance after Loan Tenure	16

List of Figures

S.No.	Title	Page No.
1	Figure 1: Screenshot showing how java FX hyperlink looks	10
2	Figure 2: Toggle Button Pressed and Unpressed	13
3	Figure 3: Java Virtual Machine working	20
4	Figure 4: Java development Kit	21
5	Figure 5: Use Case Diagram Example	27
6	Figure 6: Actor in Use Case Diagram	28
7	Figure 7: Use Case	29
8	Figure 8: Boundary of System	30
9	Figure 9: Extend Relationship	30
10	Figure 10: Include Relationship	31
11	Figure 11: Generalization Relationship	31
12	Figure 12: Example of Association Link	32
13	Figure 13: Example of Include Relationship	32
14	Figure 14: Example of Extend Relationship	33
15	Figure 15: Example of Generalization Relationship	33
16	Figure 16: Use Case level of details	35
17	Figure 17: Use Case of Consumer Loan Assistant	36
18	Figure 18: Result and screenshot of final application	39

Acronyms

JVM	Java Virtual Machine
JDK	Java Development Kit
JRE	Java Runtime Environment
RIA	Rich Internet Application
GUI	Graphic User Interface
GPU	Graphical Processing Unit
CSS	Cascading Style Sheet
API	Application Programming Interface
XML	Extensible Markup Language

CHAPTER-1 INTRODUCTION

The goal of this project is to make it easier for consumers to compute the monthly EMI of a loan obtained from a bank via credit card, debit card, or other ways. The major goal of creating this project is to manage all aspects of the bank's loans and investments.

The initiative was created to make loan processing more efficient in banks. The user's loan balance and annual interest rate are fed into our proposed technology, which automates the loan process. Customers can apply for a loan and then track their monthly EMI with our Consumer Loan Assistant once it has been authorized.

Consumer Loan Assistant is a very effective technique for handling all loan-related transactions in a timely and correct manner. A loan is a contract between two parties: a lender and a borrower. The lender offers the borrower a certain amount of money with the understanding that the amount borrowed will be paid back in monthly instalments with interest over a predetermined period of time by the borrower.

The simplest approach to figure out your monthly payments and balance your budget is to use a Consumer Loan Assistant. Everyone needs a loan at some point in their lives, whether it's to buy a car or a house, pay their child's education, or consolidate debts, for example. As a result, in today's world, loans have become an integral element of everyone's life.

Loans can be used for a variety of purposes, but the fundamental components of all loans are the same: the loan amount, the loan term, and the interest rate.

1.1 Formulation of Problem

In India's rural areas, money lending is perhaps the oldest form of business. Despite the expansion of the banking network, the village Mahajan remains a vital source of credit. We often see people in rural areas who are not well educated to calculate their own interest, and Mahajan's prey on them by taking more money from them, or people in urban areas who don't have enough time to calculate it, so they can use our consumer loan assistant to get a description of how to pay and how much to pay each month until the loan is paid off.

This effort will assist us in reducing consumer debt. We'd construct a Consumer Loan Assistant project that calculates payments and loan terms based on balance and interest data. We'll look at how to move the focus between controls, how to validate input, and how to use the message box to provide feedback to the user.

1.2 Tool and Technology Used

1.2.1 JAVA FX

JavaFX is a Rich Internet Application (RIA) library written in Java. This library allows apps to execute consistently across multiple platforms. JavaFX applications may run on a variety of platforms, including desktop computers, mobile phones, televisions, and tablets.

The Advanced Windowing Toolkit and Swing libraries are used by programmers to create GUI applications using the Java programming language. With the introduction of JavaFX, these Java programmers can now create rich content GUI apps.

1.2.2 Need For JAVAFX

Programmers used to rely on many libraries to provide features such as Media, UI controls, Web, 2D and 3D, and so on to construct Client Side Applications with rich functionality. JavaFX is a single library that contains all of these functionalities. Aside from these, developers can use the current capabilities of a Java library such as Swing.

JavaFX delivers a comprehensive collection of graphics and media APIs that take advantage of the contemporary Graphical Processing Unit (GPU) via hardware acceleration. Developers can also use JavaFX's interfaces to integrate graphics animation and UI control.

JavaFX works with JVM-based technologies including Java, Groovy, and JRuby. There is no need to acquire other technologies if developers choose JavaFX because prior understanding of any of the above-mentioned technologies will suffice to create RIAs with JavaFX.

1.2.3 Features of JAVAFX

Following are some of the important features of JavaFX –

- **Written in Java** – The JavaFX library is built in Java and can be used with any language that can run on a JVM, such as Java, Groovy, or JRuby. These JavaFX applications are also cross-platform.
- **FXML** – FXML is a declarative markup language similar to HTML that is included with JavaFX. This language's main purpose is to describe a user interface.
- **Scene Builder** – Scene Builder, a JavaFX application, is available. Users can access a drag-and-drop design interface for developing FXML applications after integrating this application in IDEs such as Eclipse and NetBeans (just like Swing Drag & Drop and Dream Weaver Applications).
- **Swing Interoperability** – The Swing Node class can be used to embed Swing content in a JavaFX application. Similarly, JavaFX features such as embedded web content and rich graphical media can be added to existing Swing applications.
- **Built-in UI controls** – The JavaFX library provides UI features that can be used to create a full-featured application.
- **CSS like Styling** – JavaFX has a CSS-like styling system. With a basic understanding of CSS, you can improve the design of your application.
- **Canvas and Printing API** – Canvas is a rendering API provided by JavaFX that works in immediate mode. The package `javafx.scene.canvas` contains a set of canvas classes that

can be used to draw directly within a JavaFX scene area. In the package `javafx.print`, JavaFX also includes classes for printing.

- Rich set of API's – The JavaFX library provides a comprehensive collection of APIs for creating graphical user interfaces, 2D and 3D graphics, and more. This collection of APIs also contains Java platform features. As a result, you can use this API to access Java language features like Generics, Annotations, Multithreading, and Lambda Expressions. The conventional Java Collections library has been expanded to support notions such as observable lists and maps. The consumers can observe the changes in the data models using these.
- Graphics pipeline – JavaFX supports visuals that use the Prism hardware-accelerated graphics pipeline. It provides smooth graphics when used with a compatible Graphic Card or GPU. Prism uses the software rendering stack if the system does not support graphics cards.

JavaFX is compatible with Windows Vista, Windows 7, Windows 8, Windows 10, macOS, and Linux on desktop computers. Oracle has published beta versions for Open Solaris starting with JavaFX 1.2. On mobile, JavaFX Mobile 1.x can run on a variety of platforms, including Symbian OS, Windows Mobile, and proprietary real-time operating systems.

JavaFX was planned to replace Swing as the standard Java SE GUI toolkit, but it's been deleted from new Standard Editions, but Swing and AWT are still included, ostensibly because JavaFX's market share has been "eroded by the rise of 'mobile first' and 'web first' apps." Oracle made JavaFX part of the OpenJDK under the OpenJFX project with the release of JDK 11 in 2018, in order to speed up its development. JavaFX support is also available for Java JDK 8 until March 2025, according to Oracle.

The open-source JavaFXPorts supports iOS (iPhone and iPad), Android, and embedded (Raspberry Pi), while the commercial "Gluon" software covers the same mobile platforms with additional functionality as well as desktop. This enables the creation of desktop, iOS, and Android applications from a single source code base.

Button, text fields, tables, trees, menus, charts, and other graphical user interface components are all included in JavaFX. JavaFX can be styled programmatically or with CSS. Simple charts can

be created with JavaFX's built-in chart library. 2D and 3D graphics are supported by JavaFX. A WebView in JavaFX may display modern web applications.

Here is a more complete list of concepts, components and features in JavaFX-

CORE

Stage - The `javafx.stage` class is a JavaFX stage. A window in a JavaFX desktop application is represented by the stage. A JavaFX Scene, which represents the content displayed inside a window - inside a Stage - can be inserted into a JavaFX Stage.

A root Stage object is created when a JavaFX application starts up and supplied to the `start(Stage primary Stage)` method of the root class of your JavaFX application. The principal window of your JavaFX application is represented by this Stage object. In the event that your program needs to open extra windows later in its life cycle, you can build new Stage objects.

Creating a Stage

You create a JavaFX Stage object just like any other Java object: Using the `new` command and the Stage constructor. Here is an example of creating a JavaFX Stage object.

```
Stage stage = new Stage();
```

Showing a Stage

Simple creating a JavaFX Stage object will not show it. In order to make the Stage visible you must call either its `show()` or `showAndWait()` method. Here is an example of showing a JavaFX Stage

```
Stage stage = new Stage();
```

```
stage.show();
```

show() vs. showAndWait()

The difference between the JavaFX Stage methods `show()` and `showAndWait()` is that `show()` displays the Stage object and then quits the `show()` function immediately, whereas `showAndWait()` displays the Stage object and then blocks (remains inside the `showAndWait()` method) until the Stage is closed.

Scene - The root of the JavaFX Scene graph is the JavaFX Scene object. To put it another way, the JavaFX Scene contains all of the visual JavaFX GUI components. The class `javafx.scene` represents a JavaFX Scene. Scene. To be visible, a Scene object must be placed on a JavaFX Stage. I'll teach you how to create a Scene object and add GUI components to it in this JavaFX Scene tutorial.

Create Scene

You create a JavaFX Scene object via its constructor. As parameter you must pass the root JavaFX GUI component that is to act as the root view to be displayed inside the Scene. Here is an example of creating a JavaFX Scene object:

```
VBox vBox = new VBox();  
Scene scene = new Scene(vBox);
```

Set Scene on Stage

In order to make a JavaFX Scene visible, it must be set on a JavaFX Stage. Here is an example of setting a JavaFX Scene on a Stage:

```
VBox vBox = new VBox(new Label("A JavaFX Label"));  
Scene scene = new Scene(vBox);  
Stage stage = new Stage();  
stage.setScene(scene);
```

A JavaFX Scene can be attached to only a single Stage at a time, and Stage can also only display one Scene at a time.

Node - The JavaFX Node class, `javafx.scene.Node`, is the base class (superclass) for all components added to the JavaFX Scene Graph. The JavaFX Node class is abstract, so you will only add subclasses of the Node class to the scene graph. All JavaFX Node instances in the scene graph share a set of common properties which are defined by the JavaFX Node class. These common properties will be covered in this JavaFX Node tutorial.

JavaFX Node Basics

Each JavaFX Node (subclass) instance can only be added to the JavaFX scene graph once. In other words, each Node instance can only appear in one place in the scene graph. If you try to add the same Node instance, or Node subclass instance, to the scene graph more than once, JavaFX will throw an exception!

A JavaFX Node can sometimes have subitems - which are also called children. Whether a given Node instance can have children or not depends on the concrete Node subclass. A special subclass of JavaFX Node named Parent is used to model Node instance which can have children. Thus, Node instances that can have children are typically children of the Parent class - and not the Node class directly.

The JavaFX Stage and JavaFX Scene classes are not subclasses of the JavaFX Node class. While these two classes are used to display the JavaFX scene graph, only Node instances added to a JavaFX Scene instance are considered part of the JavaFX Scene graph.

Once a Node instance is attached to the scene graph, only the JavaFX Application Thread, the thread managing the JavaFX scene graph, is allowed to modify the Node instance.

Properties: A JavaFX Property is a special kind member variable of JavaFX controls. JavaFX properties are typically used to contain control properties such as X and Y position, width and height, text, children and other central properties of JavaFX controls. You can attach change listeners to JavaFX properties so other components can get notified when the value of the

property changes, and you can bind properties to each other so when one property value changes, so does the other. In this JavaFX property tutorial I will explain how JavaFX properties work, and how to use them.

JavaFX Property Example

Here is a JavaFX GUI example showing how to access the `widthProperty` and `prefWidthProperty` of a JavaFX Pane, as well as adding a change listener to both. Notice how one of the change listeners is implemented as an anonymous Java class, and the other as a Java Lambda Expression. This is just to show you two different ways of achieving the same goal of attaching an event listener to a JavaFX property.

When the instruction `prefWidthProperty.set(123);` is called, the `prefWidthProperty` change listener will get called. Additionally, every time the UI is resized then the Pane is resized too, and the `widthProperty` change listener will get called.

FXML - JavaFX FXML is an XML format that enables you to compose JavaFX GUIs in a fashion similar to how you compose web GUIs in HTML. FXML thus enables you to separate your JavaFX layout code from the rest of your application code. This cleans up both the layout code and the rest of the application code.

FXML can be used both to compose the layout of a whole application GUI, or just part of an application GUI, e.g. the layout of one part of a form, tab, dialog etc.

JavaFX FXML Example

The easiest way to start learning about JavaFX FXML is to see an FXML example. Below is a FXML example that composes a simple JavaFX GUI:

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.control.Label?>
```

```
<VBox>

  <children>

    <Label text="Hello world FXML"/>

  </children>

</VBox>
```

This example defines a VBox containing a single Label as child element. The VBox component is a JavaFX layout component. The Label just shows a text in the GUI. Don't worry if you do not yet understand all the JavaFX components. You will once you start playing with them all.

The first line in the FXML document is the standard first line of XML documents.

The following two lines are import statements. In FXML you need to import the classes you want to use. Both JavaFX classes and core Java classes used in FXML must be imported.

After the import statements you have the actual composition of the GUI. A VBox component is declared, and inside its children property is declared a single Label component. The result is that the Label instance will be added to the children property of the VBox instance.

BASIC CONTROL

Label: The JavaFX Label control can display a text or image label inside a JavaFX GUI. The label control must be added to the scene graph to be visible. The JavaFX Label control is represented by the class `javafx.scene.control.Label` .

Creating a Label

You create a label control instance by creating an instance of the Label class. Here is a JavaFX Label instantiation example:

```
Label label = new Label("My Label");
```

HyperLink - The JavaFX Hyperlink control is a text that functions as a button, meaning you can configure a Hyperlink to perform some action when the user clicks it. Just like a hyperlink in a web page. The JavaFX Hyperlink control is represented by the class `javafx.scene.control.Hyperlink`.

Here is a screenshot showing how a JavaFX Hyperlink looks

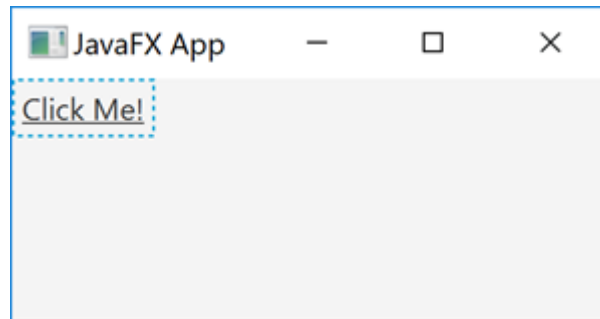


Figure 1: Screenshot of JavaFX Hyperlink

Button: A JavaFX Button control enables a JavaFX application to have some action executed when the application user clicks the button. The JavaFX Button control is represented by the class `javafx.scene.control.Button`. A JavaFX Button can have a text and an icon on it which indicate to the user what clicking the button will do.

Creating a Button

You create a button control by creating an instance of the Button class. Here is a JavaFX Button instantiation example:

```
Button button = new Button("My Label");
```

The text to be displayed on the button is passed as parameters to the Button constructor.

Menu Button: The JavaFX MenuButton control works like a regular JavaFX Button except it provides a list of options which the user can choose to click. Each of these options function like a separate button - meaning your application can listen for clicks and respond individually to each option. In a way, a JavaFX MenuButton works a bit like a JavaFX MenuBar.

The JavaFX MenuButton can show or hide the menu items. The menu items are usually shown when a little arrow button is clicked in the MenuButton. The JavaFX MenuButton control is represented by the class `javafx.scene.control.MenuButton`.

MenuButton vs. ChoiceBox and ComboBox

The MenuButton looks similar to a **ChoiceBox** and **ComboBox**, but the difference is, that the MenuButton is designed to trigger an action when you select one of its menu options, whereas ChoiceBox and ComboBox are designed to just note internally what option was selected so it can be read later.

Creating a MenuButton

You create a JavaFX MenuButton by creating an instance of the MenuButton class. The MenuButton constructor takes a button text and a button graphic. You can pass null for the text and / or the graphic, in case you want a MenuButton without either text or graphic. Here is an example of creating a JavaFX MenuButton with only a text label:

```
MenuItem menuItem1 = new MenuItem("Option 1");
```

```
MenuItem menuItem2 = new MenuItem("Option 2");
```

```
MenuItem menuItem3 = new MenuItem("Option 3");
```

```
MenuButton menuButton = new MenuButton("Options", null, menuItem1, menuItem2, menuItem3);
```

First 3 MenuItem instances are created, each with a different text. Then a MenuButton instance is created, passing a button text, a graphic icon (null) and the 3 MenuItem instances as parameter to the MenuButton constructor.

The second MenuButton constructor parameter is a Node which is used as a graphic icon which is shown next to the MenuButton text. You could use an ImageView control to display an an

image next to the MenuButton text. Just create an ImageView instance and pass a reference to that to the MenuButton constructor, instead of null.

Split Menu Button: The JavaFX SplitMenuButton control can show a list of menu options which the user can choose from, as well as a button which the user can click on when a menu option has been chosen. The JavaFX SplitMenuButton can show or hide the menu items. The menu items are usually shown when a little arrow button is clicked in the SplitMenuButton. The JavaFX SplitMenuButton control is represented by the class `javafx.scene.control.SplitMenuButton`.

Create SplitMenuButton

Before you can use the JavaFX SplitMenuButton you must create an instance of it. Here is an example of creating a JavaFX SplitMenuButton:

```
SplitMenuButton splitMenuButton = new SplitMenuButton();
```

SplitMenuButton Text

You can set the SplitMenuButton's button text via its `setText()` method. Here is an example of setting the button text of a JavaFX SplitMenuButton:

```
splitMenuButton.setText("Click here!");
```

SplitMenuButton Font

The JavaFX Split MenuButton enables you to set the font used to render the text of the SplitMenuButton. You can read more about creating fonts in JavaFX in my JavaFX Fonts tutorial. Here is an example of setting a font on a JavaFX SplitMenuButton:

```
SplitMenuButton splitMenuButton = new SplitMenuButton();
```

```
Font font = Font.font("Courier New", FontWeight.BOLD, 36);
```

```
splitMenuButton.setFont(font);
```

Toggle Button: A JavaFX ToggleButton is a button that can be selected or not selected. Like a button that stays in when you press it, and when you press it the next time it comes out again. Toggled - not toggled. The JavaFX ToggleButton is represented by the class `javafx.scene.control.ToggleButton`.

Creating a ToggleButton

You create a JavaFX ToggleButton by creating an instance of the ToggleButton class. Here is an example of creating a JavaFX ToggleButton instance:

```
ToggleButton toggleButton1 = new ToggleButton("Left");
```

This example creates a ToggleButton with the text Left on.

Adding a ToggleButton to the Scene Graph

To make a ToggleButton visible you must add it to the JavaFX scene graph. This means adding it to a Scene, or as child of a layout which is attached to a Scene object.

The application resulting from running the above example code is illustrated in the following two screenshots. The first screenshot shows a ToggleButton which is not pressed, and the second screenshot shows the same ToggleButton pressed (selected, activated etc.):

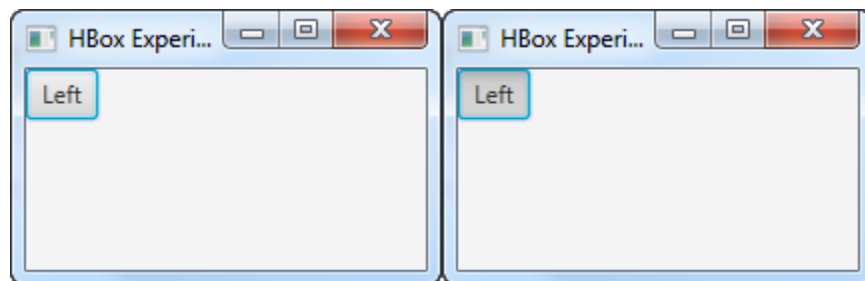


Figure 2: Toggle Button Pressed and Unpressed

Why JAVA FX?

There are several reasons why JavaFX is a great GUI application platform. First of all, Java is still one of the most popular programming languages in the world, with a large set of standard classes, and a rich set of open source toolkits developed by the Java developer community.

Second, JavaFX can run on all of the following OS'es and devices:

- Windows
- Linux
- Mac
- iOS
- Android / Chromebook
- Raspberry Pi

This makes JavaFX a versatile cross OS and cross device application toolkit.

Third, JavaFX comes with a rich set of GUI controls, and open source toolkits add even more tools to the total ecosystem.

CHAPTER-2 LITERATURE SURVEY/PROJECT DESIGN

On availing of a personal loan, you are expected to repay the principal amount along with a certain percentage as interest levied by the loan provider. The payment of this amount is spread throughout the tenure of your loan.

The total interest payable throughout the tenure is an important factor to be taken into consideration because it can significantly increase the total sum payable to the lender.

How is Interest Calculated on Personal Loans?

The interest on your personal loan or any other loan is calculated in the following manner:

$$\text{EMI} = [P \times (R/100) \times \{1+(R/100)\}^N] / [\{1+(R/100)\}^{(N-1)}]$$

Where,

EMI = equated monthly instalments

P = the principal amount borrowed

R = loan interest rate (monthly basis) = annual interest rate/12

N = loan tenure (in months)

Let us assume that a borrower borrows a sum of Rs. 5 lakhs at a rate of 12% for a tenure of 5 years, the interest for the 1st month will be calculated as follows:

$$(5,00,000 \times 0.12/12) = 5,000$$

The total EMI payable is the sum of the interest and principal amount.

How to Calculate Interest Component on Personal Loan EMI?

If you would like to know the interest component of your EMI for a particular month/installment, you can use the IPMT formula in Microsoft Excel.

The formula is as follows

$$=IPMT(\text{rate}, \text{per}, \text{nper}, \text{pv}, [\text{fv}], [\text{type}])$$

Where

rate = the rate of interest. Thus, if you are being charged 12% annually, you need to enter 12%/12 for this field if you are calculating on a monthly basis.

per = the instalment or month for which you are calculating the interest component

nper = the overall loan tenure (in terms of number of EMIs)

pv = principal / loan amount

[fv] = this is an optional field and refers to the desired cash balance at the end of the loan tenure.

Typically, this is set to 0

[type] = this can be 0 or 1 depending on whether the payment is being made at the

Table 1: Table to show Revised Balance

Month	Loan Balance	EMI	Interest	Principal	Revised Balance
1	5,00,000	11,122	5,000	6,122	4,93,878

beginning or end of the month.

Here's a working example for your reference -

For subsequent months, the interest will be calculated on the new loan balance (also known as principal amount outstanding). The new loan balance is calculated as follows:

Loan paid - Principal already paid

The total interest payable is the sum of interest paid over the tenure of the loan.

What are the Factors Used to Calculate Rate of Interest?

While the Central Bank regulates the rate of interest, the rate of interest charged by different loan providers is calculated taking into account different factors like:

- **CIBIL Score**

This score is a reflection of your creditworthiness. The minimum CIBIL score for availing a personal loan varies according to the internal policies of the lender.

- **Principal Amount**

Your chances of obtaining a larger amount are higher for lenders with a good repayment history.

- **Repayment and loan tenure**

The shorter the repayment tenure, the lesser the total interest to be paid.

It is advisable to have a clear idea about the terms and conditions and the interest payable to be able to shortlist a provider best suited to your needs. Log into the website of a loan provider, enter the details required in the personal loan calculator to help you calculate the interest and monthly installments. Make sure to invest adequate time on researching offers provided by different providers since a loan, like any other financial decision, causes a deep impact on your financial health.

Features of the language used:

In my project, I have chosen Java language for developing the code.

About Java

Initially the language was called as “oak” but it was renamed as “Java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

Ø Java is a programmer’s language.

Ø Java is cohesive and consistent.

Ø Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

Finally, Java is to Internet programming where C was to system programming.

Importance of Java to the Internet

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

Java can be used to create two types of programs

Applications and Applets: An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java's ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

Features of Java

Security

Every time you that you download a “normal” program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scanned them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a “firewall” between a network application and your computer.

When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

Portability

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed. As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java's solution to these two problems is both elegant and efficient.

The Byte code

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

Java, Virtual Machine (JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all

accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

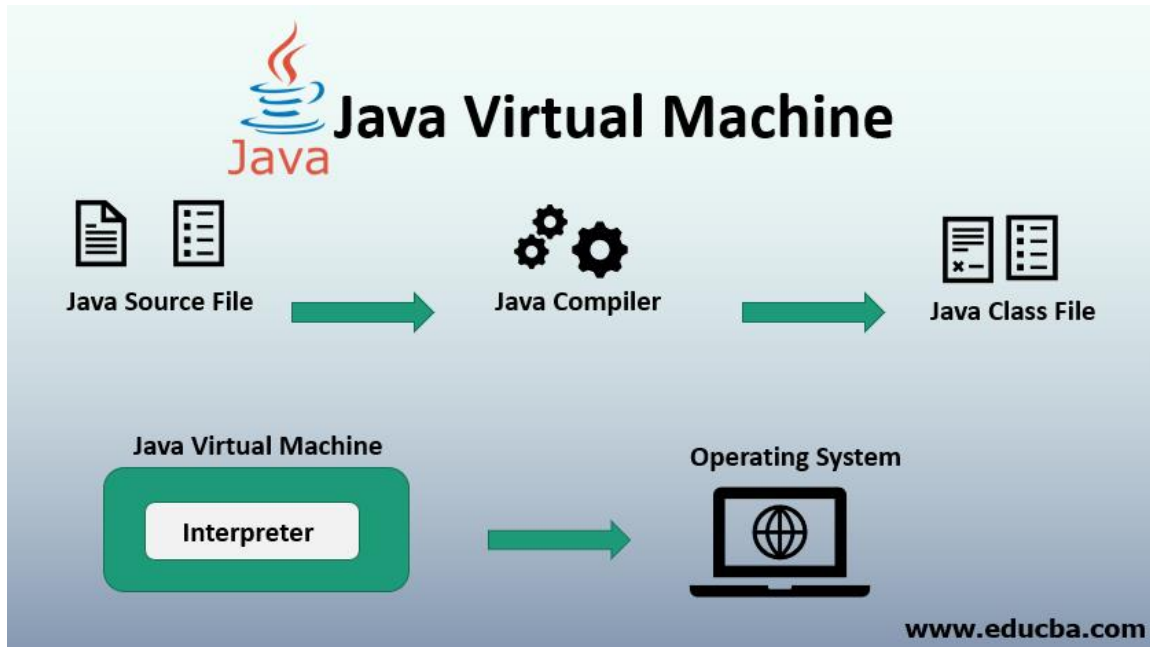


Figure 3: Java Virtual Machine Working

Java Development Kit(JDK)

The Java Development Kit (JDK) is a cross-platformed software development environment that offers a collection of tools and libraries necessary for developing Java-based software applications and applets. It is a core package used in Java, along with the JVM (Java Virtual Machine) and the JRE (Java Runtime Environment).

Beginners often get confused with JRE and JDK, if you are only interested in running Java programs on your machine then you can easily do it using Java Runtime Environment. However, if you would like to develop a Java-based software application then along with JRE you may need some additional necessary tools, which is called JDK.

$$\text{JDK} = \text{JRE} + \text{Development Tools}$$

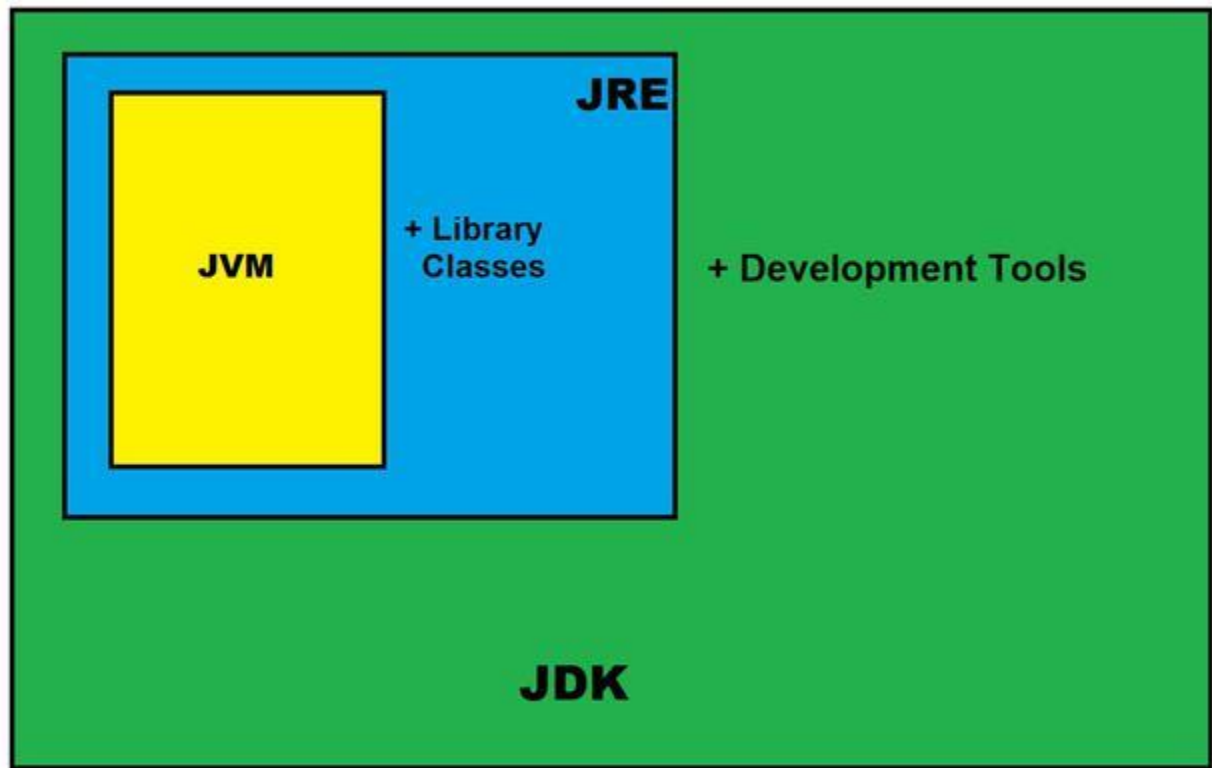


Figure 4: JAVA Development Kit (JDK)

The Java Development Kit is an implementation of one of the Java Platform:

- Standard Edition (Java SE),
- Java Enterprise Edition (Java EE),
- Micro Edition (Java ME),

Java Runtime Environment(JRE)

JRE (Java Runtime Environment) is an installation package that provides an environment to only run (not develop) the java program(or application)onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.

Client Server Technology

Overview

With the varied topic in existence in the fields of computers, Client Server is one, which has generated more heat than light, and also more hype than reality. This technology has acquired a certain critical mass attention with its dedication conferences and magazines. Major computer vendors such as IBM and DEC, have declared that Client Servers is their main future market. A survey of DBMS magazine revealed that 76% of its readers were actively looking at the client server solution. The growth in the client server development tools from \$200 million in 1992 to more than \$1.2 billion in 1996.

Client server implementations are complex but the underlying concept is simple and powerful. A client is an application running with local resources but able to request the database and relate the services from separate remote server. The software mediating this client server interaction is often referred to as Middleware.

The typical client either a PC or a Work Station connected through a network to a more powerful PC, Workstation, Midrange or Main Frames server usually capable of handling request from more than one client. However, with some configuration server may also act as client. A server may need to access other server in order to process the original client request.

The key client server idea is that client as user is essentially insulated from the physical location and formats of the data needs for their application. With the proper middleware, a client input from or report can transparently access and manipulate both local database on the client machine and remote databases on one or more servers. An added bonus is the client server opens the door to multi-vendor database access indulging heterogeneous table joins.

What is a Client Server?

Two prominent systems in existence are client server and file server systems. It is essential to distinguish between client servers and file server systems. Both provide shared network access to data but the comparison ends there! The file server simply provides a remote disk drive that can

be accessed by LAN applications on a file by file basis. The client server offers full relational database services such as SQL-Access, Record modifying, Insert, Delete with full relational integrity backup/ restore performance for high volume of transactions, etc. the client server middleware provides a flexible interface between client and server, who does what, when and to whom.

Why Client Server?

Client server has evolved to solve a problem that has been around since the earliest days of computing: how best to distribute your computing, data generation and data storage resources in order to obtain efficient, cost effective departmental an enterprise wide data processing. During mainframe era choices were quite limited. A central machine housed both the CPU and DATA (cards, tapes, drums and later disks). Access to these resources was initially confined to batched runs that produced departmental reports at the appropriate intervals. A strong central information service department ruled the corporation. The role of the rest of the corporation limited to requesting new or more frequent reports and to provide hand written forms from which the central data banks were created and updated. The earliest client server solutions therefore could best be characterized as “SLAVE-MASTER”.

Time-sharing changed the picture. Remote terminal could view and even change the central data, subject to access permissions. And, as the central data banks evolved in to sophisticated relational database with non-programmer query languages, online users could formulate adhoc queries and produce local reports with out adding to the MIS applications software backlog. However remote access was through dumb terminals, and the client server remained subordinate to the Slave\Master.

Front end or User Interface Design

The entire user interface is planned to be developed in browser specific environment with a touch of Intranet-Based Architecture for achieving the Distributed Concept.

The browser specific components are designed by using the HTML standards, and the dynamism of the designed by concentrating on the constructs of the Java Server Pages.

In this project, we will build a consumer loan assistant. You input a loan balance and yearly interest rate. You then have two options:

- (1) Enter the desired number of payments and the loan assistant computes the monthly payment, or
- (2) Enter the desired monthly payment and the loan assistant determines the number of payments you will make. An analysis of your loan, including total of payments and interest paid is also provided.

All label controls are used for title information. Two button controls are used to compute results and to start a new analysis. Two small button controls (marked with X; only one is seen at a time) control whether you compute the number of payments or the payment amount. One button exits the project. Four text field controls are used for inputs and a large text area is used to present the loan analysis results.

The loan assistant appears as:

In this initial configuration, you enter a Loan Balance, an Interest Rate (annual rate as a percentage) and a Number of Payments value. Click Compute Monthly Payment. The payment will appear in the 'yellow' text field and a complete loan analysis will appear in the large text field. Here are some numbers I tried:

So, if I borrow \$10,000 at 5.5% interest, I will pay \$301.96 for three years (36 months). More specific details on exact payment amounts, including total interest paid, is shown under Loan Analysis. At this point, you can click New Loan Analysis to try some new values:

Note the Loan Balance, Interest Rate, and Number of Payments entries remain. Only the Monthly Payment and the Loan Analysis have been cleared. This lets you try different values with minimal typing of new entries. Change any entry you like to see different results – or even change them all. Try as many combinations as you like. At some point, clear the text fields and click the button with an X next to the number of Payment text field. You will see: Notice the Number of Payments box

is now yellow. The button with an X has moved to the Monthly Payment text field. In this configuration, you enter a Loan Balance, an Interest Rate and a Monthly Payment. The loan

assistant will determine how many payments you need to pay off the loan. Here are some numbers I tried: It will take 59 payments (the last one is smaller) to pay off this particular loan. Again, you can click New Loan Analysis to try other values and see the results. That's all you do with the loan assistant project – there's a lot going on behind the scenes though. The loan assistant has two modes of operation. It can compute the monthly payment, given the balance, interest and number of payments. Or, it can compute the number of payments, given the balance, interest, and payment. The text field representing the computed value is yellow. The button marked X is used to switch from one mode to the next. To exit the project, click the Exit button.

FEASIBILITY REPORT- Consumer Loan Assistant

Feasibility study is the high level capsule version of the entire requirement analysis process. The objective of feasibility study is to determine whether the proposed system can be developed with available resources.

There are three steps to be followed for determining the feasibility study of proposed system.

Ø Technical Feasibility

Ø Operational Feasibility

Ø Economical Feasibility

Technical Feasibility: It is concerned with hardware and software feasibility. In this study, one has to test whether the proposed system can be developed using existing technology or not. If new technology is required, what is the likely hood that it can be developed? The organization for which the system to be developed is not provided online services. Hence there is a requirement of new hardware and software technology for the deployment of proposed system. As per client requirements the system to be developed should have speed response because of fast change info, programming productivity, reliability, security, scalability, integration and availability. To meet these requirements I as a developer found jsp1.1 as a right choice because of its features platform independence, modularity and reusability.

Operational Feasibility: Operational feasibility determines whether the proposed system satisfied the user objectives and can be fitted in to current system operation. The proposed system “Lending Tree” can be justified as operationally feasible basing on the following.

Ø The methods of processing and presentation are completely acceptable by the clients because they meet all the user and client requirements.

Ø The clients have been involved during the preparation of requirement analysis and design process.

Ø The system will certainly satisfy the user objectives and it will also enhance their capability.

Ø The system can be best fitted into current operation and requires little training to both administrator and dealer. With the help of this system customer to place order requires simple data entry through forms provided. The proposed system is completely user friendly.

Economical Feasibility: This includes an evaluation of all incremental costs and benefits expected if proposed system is implemented. costs-benefit analysis which is to be performed during economical feasibility delineates costs for project development and weighs them against benefits of system. In this the proposed system replaces the manual process of receiving orders which benefits the organization to get more orders and good response. So developing this system is economically feasible to organization.

PROJECT DESIGN

Origin of Use Case

These days use case modeling is often associated with UML, although it has been introduced before UML existed. Its brief history is as follow:

- In 1986, Ivar Jacobson first formulated textual and visual modeling techniques for specifying use cases.

- In 1992 his co-authored book Object-Oriented Software Engineering - A Use Case Driven Approach helped to popularize the technique for capturing functional requirements, especially in software development.

Purpose of Use Case Diagram

Use case diagrams are typically developed in the early stage of development and people often apply use case modeling for the following purposes:

- Specify the context of a system
- Capture the requirements of a system
- Validate a systems architecture
- Drive implementation and generate test cases
- Developed by analysts together with domain experts

Use Case Diagram at a Glance

A standard form of use case diagram is defined in the Unified Modeling Language as shown in the Use Case Diagram example below:

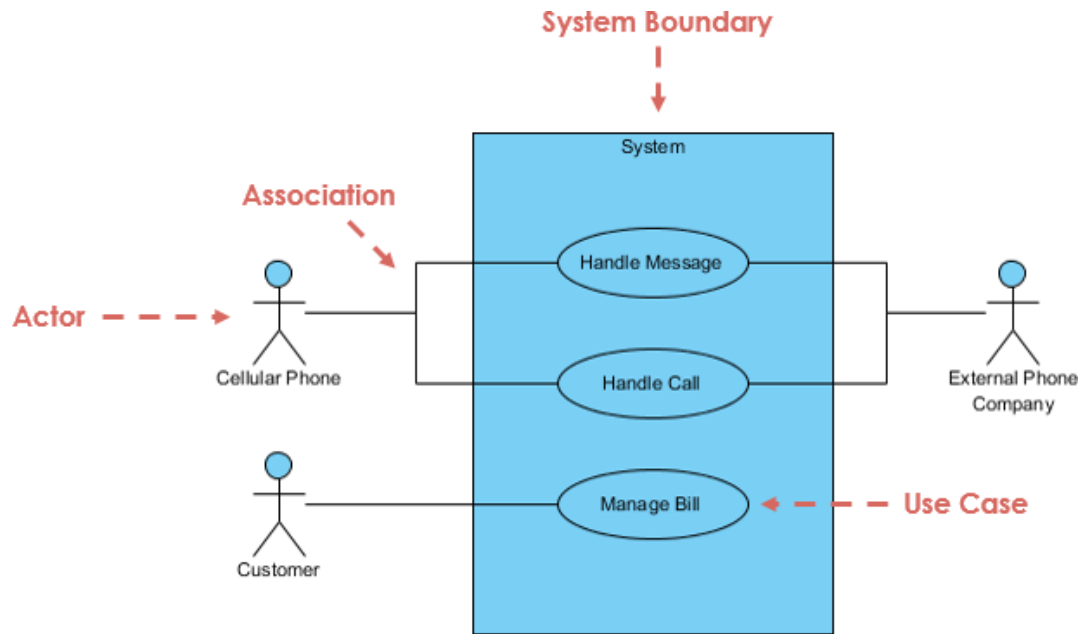


Figure 5: Example of Use Case Diagram

Actor

- Someone interacts with use case (system function).
- Named by noun.
- Actor plays a role in the business
- Similar to the concept of user, but a user can play different roles
- For example:
 - A prof. can be instructor and also researcher
 - plays 2 roles with two systems
- Actor triggers use case(s).
- Actor has a responsibility toward the system (inputs), and Actor has expectations from the system (outputs).

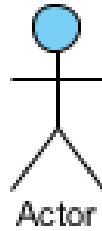


Figure 6: Actor in a use case diagram

Use Case

- System function (process - automated or manual)
- Named by verb + Noun (or Noun Phrase).
- i.e. Do something
- Each Actor must be linked to a use case, while some use cases may not be linked to actors.



Figure 7: Use Case in a Use Case Diagram

Communication Link

- The participation of an actor in a use case is shown by connecting an actor to a use case by a solid link.
- Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.

Figure 8: Communication link in a Use Case Diagram

Boundary of system

- The system boundary is potentially the entire system as defined in the requirements document.
- For large and complex systems, each module may be the system boundary.
- For example, for an ERP system for an organization, each of the modules such as personnel, payroll, accounting, etc.
- can form a system boundary for use cases specific to each of these business functions.
- The entire system can span all of these modules depicting the overall system boundary

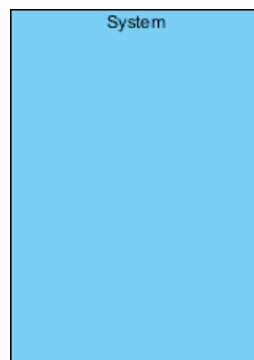


Figure 9: Boundary System in a Use Case Diagram

Structuring Use Case Diagram with Relationships

Use cases share different kinds of relationships. Defining the relationship between two use cases is the decision of the software analysts of the use case diagram. A relationship between two use cases is basically modeling the dependency between the two use cases. The reuse of an existing use case by using different types of relationships reduces the overall effort required in developing a system. Use case relationships are listed as the following:

Use Case Relationship

Extends

- Indicates that an "Invalid Password" use case may include (subject to specified in the extension) the behavior specified by base use case "Login Account".

- Depict with a directed arrow having a dotted line. The tip of arrowhead points to the base use case and the child use case is connected at the base of the arrow.
- The stereotype "<<extends>>" identifies as an extend relationship

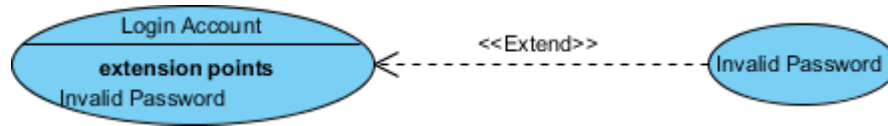


Figure 10: Extend Relationship

Include

- When a use case is depicted as using the functionality of another use case, the relationship between the use cases is named as include or uses relationship.
- A use case includes the functionality described in another use case as a part of its business process flow.
- A uses relationship from base use case to child use case indicates that an instance of the base use case will include the behavior as specified in the child use case.
- An include relationship is depicted with a directed arrow having a dotted line. The tip of arrowhead points to the child use case and the parent use case connected at the base of the arrow.
- The stereotype "<<include>>" identifies the relationship as an include relationship.

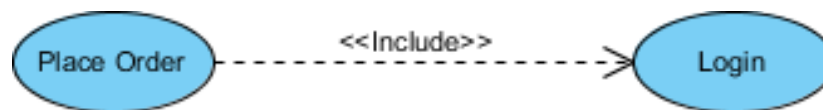


Figure 11: Include Relationship

Generalization

- A generalization relationship is a parent-child relationship between use cases.
- The child use case is an enhancement of the parent use case.

- Generalization is shown as a directed arrow with a triangle arrowhead.
- The child use case is connected at the base of the arrow. The tip of the arrow is connected to the parent use case.

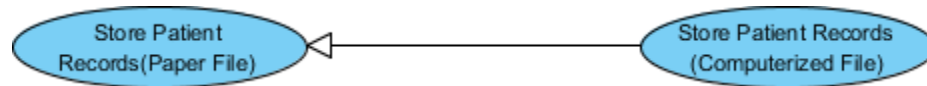


Figure 12: Generalization Relationship

Use Case Examples

Use Case Example - Association Link

A Use Case diagram illustrates a set of use cases for a system, i.e. the actors and the relationships between the actors and use cases.



Figure 13: Example of Association Link

Use Case Example - Include Relationship

The include relationship adds additional functionality not specified in the base use case. The <<Include>> relationship is used to include common behavior from an included use case into a base use case in order to support the reuse of common behavior.

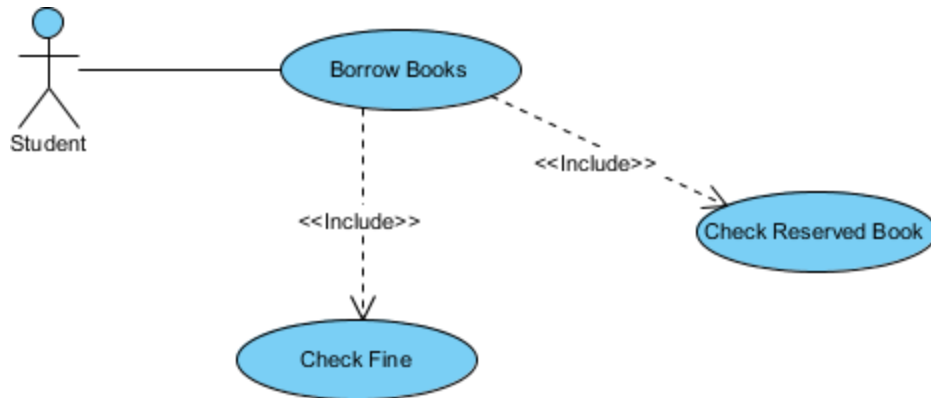


Figure 14: Example of Include Relationship

Use Case Example - Extend Relationship

The extend relationships are important because they show optional functionality or system behavior. The <<extend>> relationship is used to include optional behavior from an extending use case in an extended use case. Take a look at the use case diagram example below. It shows an extend connector and an extension point "Search".

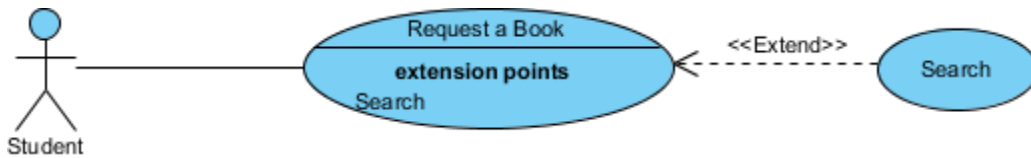


Figure 15: Example of Extend Relationship

Use Case Example - Generalization Relationship

A generalization relationship means that a child use case inherits the behavior and meaning of the parent use case. The child may add or override the behavior of the parent. The figure below provides a use case example by showing two generalization connectors that connect between the three use cases.

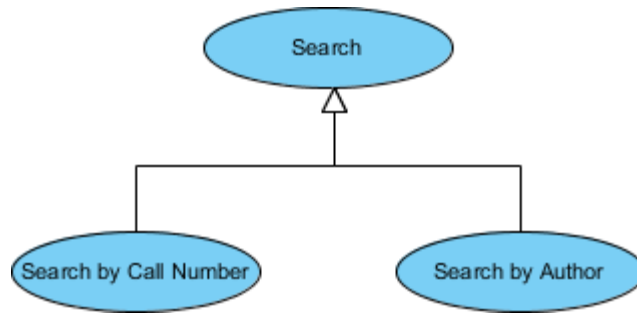


Figure 16: Example of Generalization Relationship

How to Identify Actor

Often, people find it easiest to start the requirements elicitation process by identifying the actors. The following questions can help you identify the actors of your system (Schneider and Winters - 1998):

- Who uses the system?
- Who installs the system?
- Who starts up the system?
- Who maintains the system?
- Who shuts down the system?
- What other systems use this system?
- Who gets information from this system?
- Who provides information to the system?
- Does anything happen automatically at a present time?

How to Identify Use Cases?

Identifying the Use Cases, and then the scenario-based elicitation process carries on by asking what externally visible, observable value that each actor desires. The following questions can be asked to identify use cases, once your actors have been identified (Schneider and Winters - 1998):

- What functions will the actor want from the system?
- Does the system store information? What actors will create, read, update or delete this information?
- Does the system need to notify an actor about changes in the internal state?
- Are there any external events the system must know about? What actor informs the system of those events?

Use Case Diagram Tips

Now, check the tips below to see how to apply use case effectively in your software project.

- Always structure and organize the use case diagram from the perspective of actors.
- Use cases should start off simple and at the highest view possible. Only then can they be refined and detailed further.
- Use case diagrams are based upon functionality and thus should focus on the "what" and not the "how".

Use Case Levels of Details

Use case granularity refers to the way in which information is organized within use case specifications, and to some extent, the level of detail at which they are written. Achieving the right level of use case granularity eases communication between stakeholders and developers and improves project planning.

Alastair Cockburn in *Writing Effective Use Cases* gives us an easy way to visualize different levels of goal level by thinking in terms of the sea:



Level	Example	Summary
 Cloud	Product	High Summary
 Kite	Sell Products	Summary
 Sea	Sell Furniture	User goal
 Fish	Browse Catalog	Sub-function
 Clam	Insert Orderline	Low level

Figure 17: Use Case Level of Details

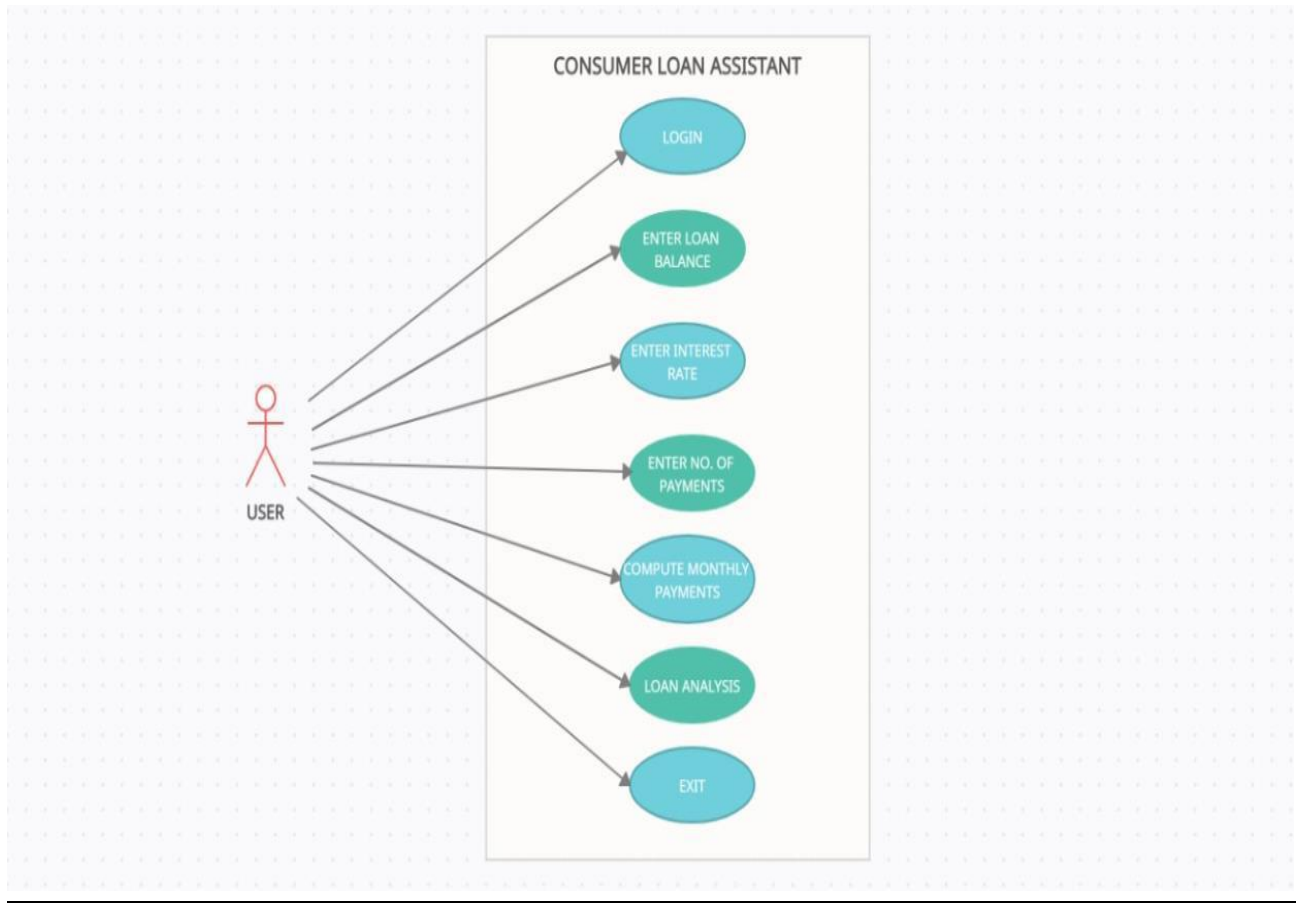


Figure 18: Use Case Diagram of Consumer Loan Assistant

CHAPTER-3 FUNCTIONALITY/WORKING OF PROJECT

In this project, we will build a consumer loan assistant. You input a loan balance and yearly interest rate. You then have two options:

(1) Enter the desired number of payments and the loan assistant computes the monthly payment, or

(2) Enter the desired monthly payment and the loan assistant determines the number of payments you will make. An analysis of your loan, including total of payments and interest paid is also provided.

All label controls are used for title information. Two button controls are used to compute results and to start a new analysis. Two small button controls (marked with X; only one is seen at a time) control whether you compute the number of payments or the payment amount. One button exits the project. Four text field controls are used for inputs and a large text area is used to present the loan analysis results.

The loan assistant appears as:

In this initial configuration, you enter a Loan Balance, an Interest Rate (annual rate as a percentage) and a Number of Payments value. Click Compute Monthly Payment. The payment will appear in the 'yellow' text field and a complete loan analysis will appear in the large text field. Here are some numbers I tried:

So, if I borrow \$10,000 at 5.5% interest, I will pay \$301.96 for three years (36 months). More specific details on exact payment amounts, including total interest paid, is shown under Loan Analysis. At this point, you can click New Loan Analysis to try some new values:

Note the Loan Balance, Interest Rate, and Number of Payments entries remain. Only the Monthly Payment and the Loan Analysis have been cleared. This lets you try different values with minimal typing of new entries. Change any entry you like to see different results – or even change them all. Try as many combinations as you like. At some point, clear the text fields and click the button with an X next to the number of Payment text field. You will see: Notice the Number of Payments box

is now yellow. The button with an X has moved to the Monthly Payment text field. In this configuration, you enter a Loan Balance, an Interest Rate and a Monthly Payment. The loan

assistant will determine how many payments you need to pay off the loan. Here are some numbers I tried: It will take 59 payments (the last one is smaller) to pay off this particular loan. Again, you can click New Loan Analysis to try other values and see the results. That's all you do with the loan assistant project – there's a lot going on behind the scenes though. The loan assistant has two modes of operation. It can compute the monthly payment, given the balance, interest and number of payments. Or, it can compute the number of payments, given the balance, interest, and payment. The text field representing the computed value is yellow. The button marked X is used to switch from one mode to the next. To exit the project, click the Exit button.

CHAPTER-4 RESULT AND DISCUSSION

Loan Assistant

Loan Balance: 10000
Interest Rate: 5.5
Number of Payments: 36
Monthly Payment: 301.96

Loan Analysis:
Loan Balance: \$10000.00
Interest Rate: 5.50%
35 Payments of \$301.96
Final Payment of: \$300.54
Total Payments: \$10869.14
Interest Paid \$869.14

Buttons: Compute Monthly Payment, New Loan Analysis, Exit

Loan Assistant

Loan Balance: 20000
Interest Rate: 5.5
Number of Payments: 1230
Monthly Payment: 92.00

Loan Analysis:
Loan Balance: \$20000.00
Interest Rate: 5.50%
1229 Payments of \$92.00
Final Payment of: \$7.10
Total Payments: \$113075.10
Interest Paid \$93075.10

Buttons: Compute Number of Payments, New Loan Analysis, Exit

CHAPTER-5 CONCLUSION AND FUTURE ENHANCEMENT

Ultimately, in the loan management system, the result of all diligence in a solid loan management system is here. is software that helps the user to find out about various banks and their branches easily

This software reduces the amount of manual data import and provides more performance. Its system is very friendly and can easily be used by anyone. It also reduces the amount of time it takes to document customer information and other modules.

Finally, we can say that this software performs all the tasks correctly and performs the task.

- I will make an online EMI payment for the company.
- I will do a live chat with the customer.
- I will do a live chat about the user's email mailing system.
- I will also include a media subscription field.
- Build online transaction.
- Build email facilities.
- Add on SMS facilities

REFERENCES

[1] <https://www.fullertonindia.com/how-personal-loan-interest-rate-calculated.aspx>

[2] <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>

[3] <http://tutorials.jenkov.com/javafx/index.html>

[4] <https://www.geeksforgeeks.org/differences-jdk-jre-jvm/>

[5] <http://ignousupport.blogspot.com/p/loan-management-system-project-report.html>

[6] <https://techvidvan.com/tutorials/java-virtual-machine/#:~:text=JVM%20stands%20for%20Java%20Virtual%20Machine%20that%20drives%20the%20Java,Just%2Din%2Dtime%20compiler.>