**®**

**RV COLLEGE OF ENGINEERING**

**BENGALURU – 560059**

(Autonomous Institution Affiliated to VTU, Belagavi)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**“Banking Apllication using JavaFX”**

**MINI-PROJECT REPORT**

**OBJECT ORIENTED PROGRAMMING USING JAVA (18CS45) IV SEMESTER**

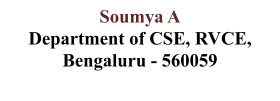
**2020-21**

**Submitted by**

**Navnith Bhardwaj 1RV19CS098**

**Neetanshu Tyagi 1RV19CS099**

**Under the Guidance of**



**RV COLLEGE OF ENGINEERING®, BENGALURU - 560059**

***(Autonomous Institution Affiliated to VTU, Belagavi)***

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



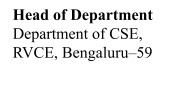
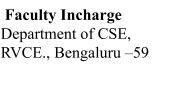
# CERTIFICATE

Certified that the **Mini-**project work titled “**Banking Apllication using JavaFX**” has been carried out by**Navnith Bhardwaj And Neetanshu Tyagi,** bonafide students of RV College of

Engineering, Bengaluru, have submitted in partial fulfillment for the **Assessment of**

**Course: OBJECT ORIENTED PROGRAMMING USING JAVA (18SC45) –**

**Open-Ended Experiments** during the year 2020-2021. It is certified that all corrections/suggestions indicated for the internal assessment have been incorporated in the report.



**RV COLLEGE OF ENGINEERING®, BENGALURU - 560059**

***(Autonomous Institution Affiliated to VTU)***

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# DECLARATION

We, **Navnith Bhardwaj (1RV19CS098) and Neetanshu Tyagi (1RV19CS099),** the students of 4th Semester B.E., Department of Computer Science and Engineering, RV

College of Engineering, Bengaluru hereby declare that the Mini-Project titled **“Banking Apllication using JavaFX”** has been carried out by us and submitted in partial fulfillment

for the **Assessment of Course: OBJECT ORIENTED PROGRAMMING USING**

**JAVA (16CS44) - Open-Ended Experiment** during the year 2020-2021.

|  |  |
| --- | --- |
| **Place: Bengaluru** |  |
| **Date:** |  |

Table Of Contents

Abstract / Synopsis

Certificate

Table of Contents

Contents

1. Introduction
   1. Object Oriented Concepts
      1. Traditional vs Object Oriented Approach
      2. OOA, OOD & OOP and their Relationship
      3. Features of OO approach
   2. Overview of Java Programming Language
      1. Features of Java
      2. Inheritance
      3. Interfaces & Packages
      4. Exception Handling
      5. Multithreaded Programming
      6. Lambda Expressions
      7. Regular Expressions
      8. Strings
      9. Collection Framework
      10. JavaFX framework
   3. Proposed System
      1. Objectives
      2. Methodology

-javafx components 2

-database part 3

-scenebuilder 1

* + 1. Scope

1. Requirement Specifications
   1. Hardware Requirements
   2. Software Requirements
2. System Design and Implementation M
   1. Class Diagrams
   2. Modular Description/ Pseudo-code
3. Results and Snapshots L
4. Conclusion
5. References APPENDIX A- SOURCE CODE Google drive link

.

# Introduction

1.1 Object Oriented Concepts

## 1.1.1 Traditional versus Object Oriented Approach

Traditional Programming can be defined as a programming model which is derived from structured programming, which is based upon the concept of calling procedure. Procedures, which are also known as routines, subroutines or functions, consist of a series of computational steps that need to be carried out. During a program’s execution, any given procedure might be called at any point, including by other procedures or itself.

FORTRAN, ALGOL, COBOL, BASIC, Pascal and C are some of the languages which use Procedural/Traditional Programming.

Object Oriented Programming can be defined as a programming model which is based upon the concept of objects. Objects contain data in the form of attributes and code in the form of methods. In this, computer programs are designed using the concept of objects that can interact with the real world. Object oriented programming languages are various but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types. Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Objective-C, Dart, Swift, Scala are some of the languages which use Object Oriented Programming.

## 1.1.2 OOA, OOD & OOP and their Relationship

Object-Oriented Analysis(OOA)

It emphasizes the building of real-world models, using an object-oriented view of the world.

It is a method of analysis that examines requirements from the perspective of the classes and objects of the problem domain.In the analysis of the task, from the objective existence of things and things The relationship between the related objects (including the attributes and behaviors of the objects) and the relationship between the objects are summarized, and the Objects with the same attributes and behaviors are represented by a class.

Object-Oriented Design(OOD)

It is a method of design encompassing the process of object oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design. Object-oriented design:

1. leads to an object-oriented decomposition and
2. uses different notations to express different models of the logical (class and objectstructure) and physical (module and process architecture) design of a system, in addition to the static and dynamic aspects of the system.

Object-Oriented Programming(OOP)

It is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

How are OOP, OOD and OOA related?

The products of object-oriented analysis serve as the models from which we may start an

object-oriented design. The products of object-oriented design can then be used as blueprints for completely implementing a system using object-oriented programming methods.

1.1.3 Features of OO approach

OOP has the following features:

Encapsulation

Encapsulation is a process of information hiding. It is simply the combination of process and data into a single entity. Data of an object is hidden from the rest of the system and available only through the services of the class. It allows improvement or modification of methods used by objects without affecting other parts of a system.

Abstraction

It is a process of taking or selecting necessary methods and attributes to specify the object. It focuses on essential characteristics of an object relative to the perspective of the user.

Inheritance

Inheritance is a great feature that allows to create sub-classes from an existing class by inheriting the attributes and/or operations of existing classes.

Polymorphism and Dynamic Binding

Polymorphism is the ability to take on many different forms. It applies to both objects and operations. A polymorphic object is one whose true type hides within a super or parent class.

In polymorphic operation, the operation may be carried out differently by different classes of objects. It allows us to manipulate objects of different classes by knowing only their common properties.

1.2 Overview of Java Programming Language

1.2.1 Features of Java

Java has the following features:

1. Simple - Java is very easy to learn, and its syntax is simple, clean and easy to understand. Java language is a simple programming language because Java syntax is based on C++. Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc. There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.
2. Object-Oriented - Java is an OO programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.
3. Portable - Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.
4. Platform-Independent - Java code can be executed on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere (WORA).
5. Multi-Threaded - A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multimedia, Web applications, etc.
6. Dynamic - Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++. Java supports dynamic compilation and automatic memory management (garbage collection).

## 1.2.2 Inheritance

It is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOP. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also. Inheritance represents the IS-A relationship which is also known as a *parent-child* relationship.

Example:

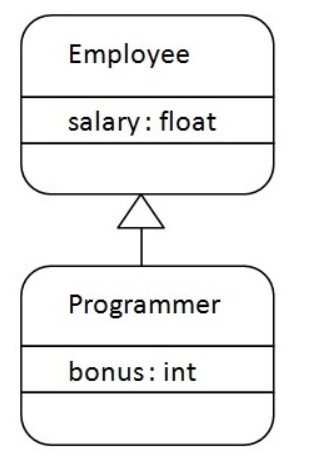


Figure 1.1

## 1.2.3 Interfaces & Packages

A package is a mechanism to group the similar type of classes, interfaces and sub-packages and provide access control. It organizes classes into a single unit.

In Java already many predefined packages are available, used while programming. For example**:** java.lang, java.io, java.util etc.

* Packages provide code reusability, because a package has a group of classes.
* It helps in resolving naming collisions when multiple packages have classes with the same name.
* Package also provides the hiding of class facilities. Thus other programs cannot use the classes from hidden packages.
* Access limitation can be applied with the help of packages.
* One package can be defined in another package.

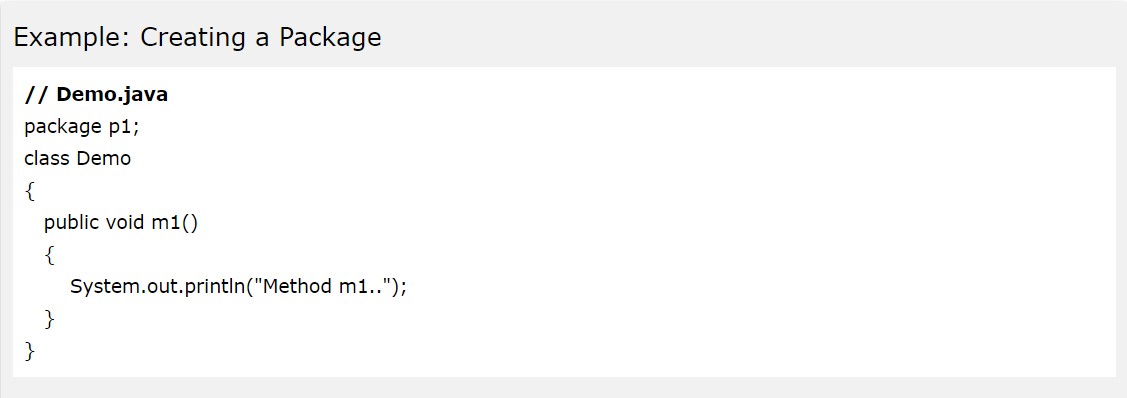
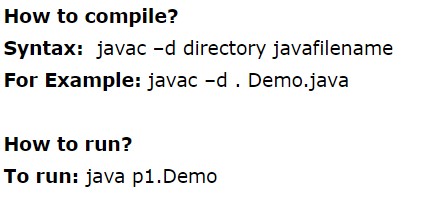


Figure 1.2



An **interface** is a blueprint of a class. It has static constants and abstract methods. The interface in Java is *a mechanism to achieve* [*abstraction*](https://www.javatpoint.com/abstract-class-in-java). There can be only abstract methods in the Java interface, not method bodies. It is used to achieve abstraction and multiple [inheritance](https://www.javatpoint.com/inheritance-in-java) [in](https://www.javatpoint.com/inheritance-in-java) [Java](https://www.javatpoint.com/inheritance-in-java). In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also

represents the IS-A relationship. It cannot be instantiated just like the abstract class.

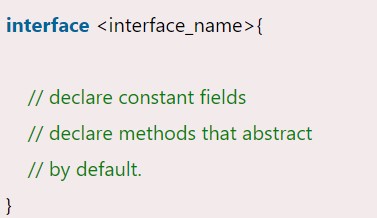
` 

Figure 1.3

## 1.2.4 Exception Handling

Exception Handling is a mechanism to handle runtime errors such as

ClassNotFoundException, IOException, SQLException, RemoteException, etc. The main advantage of exception handling is to maintain the normal flow of the application or code. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

Hierarchy of Java Exception Classes

The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:

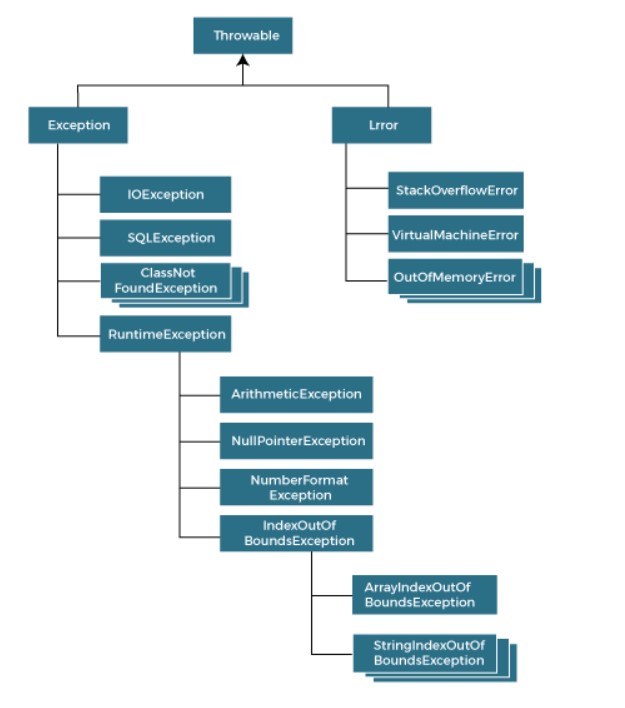


Figure 1.4

## 1.2.5 Multithreaded Programming

Multithreading is a concept wherein a process is divided into two or more subprocesses, to parallelize the processing by running concurrently. Two or more such parts of the process are called threads, with each thread defining a separate path of execution.

A thread cannot exist independently and must be attached to a process. Multithreading programming is used in order to make the process faster by achieving concurrent execution. This has a benefit over creating multiple processes since threads are relatively lightweight and do not require a separate address space unlike processes.

The life-cycle of a thread mainly comprises five stages: ● Newborn state-when the thread is created

* Runnable state- when the thread is ready to execute
* Running state- when the thread occupies the CPU
* Blocked state- when the thread is blocked from entering the runnable state
* Dead state- the thread has either been killed or has finished executing

Threads can go between these states using certain functions such as suspend, sleep, wait (running/runnable to blocked); resume, notify (blocked to running/runnable); yield (running to runnable) and stop (to dead state).

To illustrate threads and their lifecycle, we employ the below diagram:

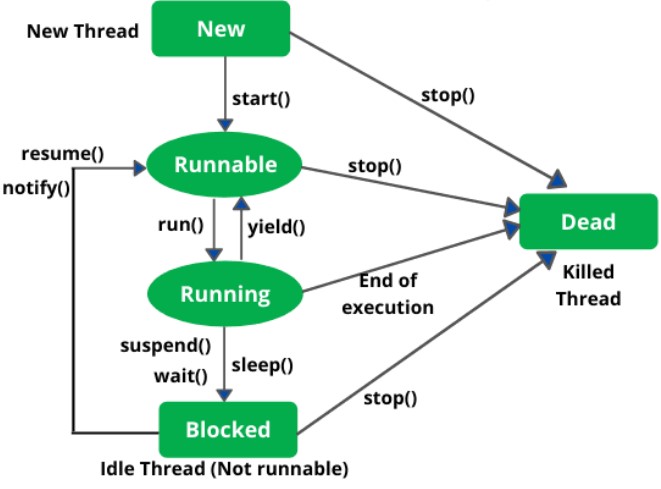


Figure1.5

In Java, all threads branch from the Main thread and can be named, given a particular priority or even put into groups using constructors. To create a thread, we can either extend the Thread class or implement the Runnable interface. Further, threads require the run method for execution and need to be set into motion using the start function.

To obtain the complete use of threads, we need to synchronize them. This enables inter-thread communication and allows us to let multiple threads access common parameters without the risk of a deadlock.

## 1.2.6 Lambda Expressions

Lambda expressions are a concise way to represent one method interface with the help of an expression. Since it can help in iterating, filtering and extracting data from collections, it is useful in this library.

In a lambda expression, we provide the implementation of a functional interface. An important term to define here is functional interface. This is an interface that consists of only one abstract method. This abstract method is later elucidated in the lambda expression which is treated as a function, thereby allowing us to forgo method definition. Thus, it reduces the amount of code required.

The syntax of a lambda expression has three components including the argument list, followed by an arrow (->) and finally succeeded by the body. We can define this in multiple ways, by varying the number of parameters in the argument list or the number of lines in the body of the function.

A concept of interest is the generic functional interfaces, which accept any data type and this further reduces our code while giving us a skeleton to define our lambda expression, which is now not limited by any data type.

In this project, we have used lambda expressions to handle various events, for example the press of a button. In such cases, the EventHandler is taken and further action upon clicking the button is described by the lambda function. An excerpt of our code illustrating the same is given below:

## 1.2.7 Regular Expressions

A regular expression is a character string describing a sequence, called a pattern. This pattern in turn, can be used in finding matches in a given input of character sequences. Regular expressions can be used to match a given string exactly, or identify a string fulfilling a set of conditions defined using wildcard characters, character sets and several quantifiers.

Two classes come into play while using regular expression processing, i.e., Pattern and Matcher. These classes work in tandem for sequence matching. While the Pattern class defines a regular expression, the Matcher class matches such a pattern with another sequence. The Pattern class defines no objects, and uses the compile method, which can be used in combination with the methods from Matcher class such as matches and find.

Apart from the classes, three quantifiers are pivotal to regular expression processing. These are:

+ : Matches if one or more occurrences

\* : Matches if zero or more occurrences

? : Matches if zero or one occurrence

Additionally, wildcards are used to denote any symbol, which when combined with quantifiers make a powerful tool that aids in data pre-processing, natural language processing, web scraping and data validation. The latter has been used in our project, wherein the credentials of the users are checked when the login using regular expressions processing. The illustration is given below:

## 1.2.8 Strings

Strings are inherently instrumental to programming, since they are used extensively in a range of applications. From taking inputs from the user to processing of expressions, we require strings and Java provides a range of functions defined on the Strings to aid in programming.

Defined by an array of characters or string literals, Strings can be equated, concatenated, and various substrings or components such as characters and bytes can be extracted from them.

Furthermore, we can also convert various data types to Strings. Additionally, modifications such as trimming whitespace, changing the case of characters in a string can be performed on Strings. Finally, we have searching, which as described above, is similar to regular expression searching and helps finding the exact string or a substring of required characters.

A comprehensive list of methods on Strings, if not exhaustive is described below:



## 1.2.9 Collections Framework

The Collections Framework is a sophisticated hierarchy of interfaces and classes that provide state-of-the-art technology for managing groups of objects while standardizing them. This high performance framework implements several basic collections such as linked lists, dynamic arrays, etc., and extending them to build upon the basic concepts.

This framework essentially is a collection of interfaces, such as List, Queue and Set. Moreover, these interfaces can be extended or implemented into classes to provide functionality of various data structures with properties as per our requirements. Each of these interfaces provide a standard set of methods, while throwing a set of exceptions. The collection framework hierarchy is given as follows:

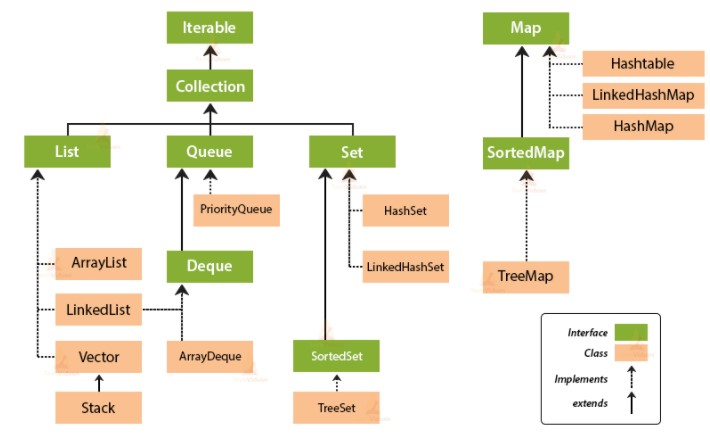


Figure 1.7

We also include Algorithms and Iterables in the Collections framework. While algorithms allow us to define certain static methods that make collection manipulation more convenient, iterables can be used to efficiently traverse through and modify collections.

In our project, we have implemented collection frameworks by using collections such as ArrayList, which provides a dynamic array that we can modify and traverse as per our requirements.

## 1.3 Proposed System

### 1.3.1 Objectives

We have utilized this project to understand the benefits of JavaFx and the full scope of its applications. Further, implementation of object oriented programming concepts have helped us understand the benefits of using OOP over traditional methods of programming.

Through the BankingPortal, we aim to achieve the following:

* Efficient transaction for the user.
* Centralized approach towards managing clients
* Effective storage of user data

### 1.3.2 Methodology

To concisely describe the development of our project, we have divided it into three parts, as follows.

### Designing

In order to fully understand the different scenes required and their functionalities, we developed a class diagram with the basic requirements. Furthermore, the user interface was designed and developed using SceneBuilder, which is integrated into Intellij to easily develop a seamless application interface. Thus we created several frames for login, Admin view, User view ..

### Integrating JavaFx

Building upon the frames developed in the designing phase, we allocated functionalities to them with reference to the class diagram. JavaFx frameworks, and various nodes such as buttons and editable tables were applied while integrating the scenes and making them interactive. The different scenes were thus layered and provided most of the functionality of the program.

### Integrating the MySQL Database

We have integrated the MySQL database in our project to store values and retrieve data from the database and display in the javafx tableview. It has been used to store details like Transaction id,user profile its address contact no,admin profile all the transactions that happen.

### 1.3.3 Scope

This project can be used in real-life t experience what is happening behind the scenes of any banking application . how we make the transactions how the bank side monitor all the dat how they stores that how we make changes etc .

## Requirement Specifications

### 2.1 Hardware specifications

Our program does not have any hardware requirements as such and can be used from any system capable of running VS code platform, or any similar IDE that allows JavaFx.

### 2.2 Software specifications

Along with the Java platform, we need the JavaFx package to be installed to run the query portal. We also need to have the database for the project at hand for storing and loading data such as Tid,Uid etc .

## System Design and Implementation

### 3.1 Class Diagram

Diagram

Description automatically generated

**ENTITIES:**

A picture containing graphical user interface

Description automatically generated

ROLES:

ADMIN,

CLIENT

BANK CLIENT

Figure 3.1

A picture containing diagram

Description automatically generated

Figure 3.2

### 3.2 Modular Description

The model we designed is on online banking portal in which there is two panel one admin panel and the other is a user panel . The major functionality of this application is to give insight of banking applications in which a user making transactions ,checking his bank statements meanwhile the admin can monitor the transaction add users . So when we run the program we will be directed to the login porta where we have to put our id and password whether we are user or a admin both have a set of login credentials to access their sides. When we wntered the id password we gwt a window of admin view /user view.

The functionalities which an admin can perform is that he can monitor all the transactions going on in portal can manage all the users , can change profile . the functionalities user can perform is that he can make transactions,can acees his previous transactions can change his provided info and can also modify his profile

## Results and Snapshots:

1. As we run our program we will move to the app view . In which the first screen is of login portal where the admin will put its admin id and password to access admin portol and user can add his/her user id and password to access user portol.

Graphical user interface

Description automatically generated

Figure 4.1

1. So now if the admin or user entered the wrong username or password a small comment will appear on screen “**WRONG USERNAME OR PASSWORD”.**

Graphical user interface, website

Description automatically generated

Figure 4.2

1. So if suppose we entered an admin id and password the program will take us to the ADMIN VIEW . which have access to all the transaction that are happening with TID,user involved . time and the amount.

Table

Description automatically generated

Figure 4.3

1. Admin can also have access to the user basic data . he can also add user by his own but two user cannot have same id,email,password.

Graphical user interface

Description automatically generated

Figure 4.4

Graphical user interface, text, table

Description automatically generated with medium confidence

Figure 4.5

1. So if in login portal we enter a user id and password the program will take us to the client/user view . where we can see all the transactions we have performed along with the user info and the balance in the account..

Text, table

Description automatically generated with medium confidence

Figure 4.6

1. User can perform the transfers also by entering the client id and the amount which he wish to transfer. But the condition is the amount should be less than the available balance also the clien id should be valid unless it will show error”**WRONG RECEIVER ID**”&”**LOW BALANCE**”

Graphical user interface, application

Description automatically generated

Figure 4.7

Graphical user interface, website

Description automatically generated

Figure 4,8

1. The one more feature of this application is at any time the admin as well as the user can change his/her profile info but the condition is it cannot take the already taken credentials. If we try to take already used credentials it will show error message.”USERNAME IS ALREADY TAKEN”

Graphical user interface, website

Description automatically generated

Figure 4.9

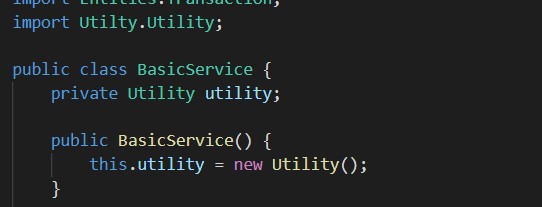
Graphical user interface

Description automatically generated

Figure 4.10

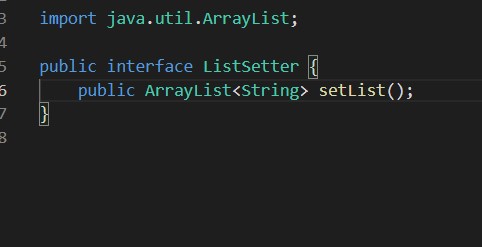
**OOP’S CONCEPTS USED:**

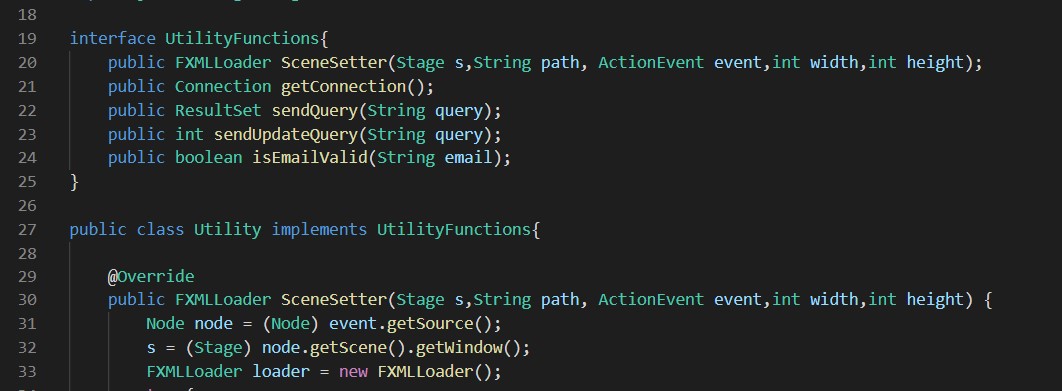
1. Inheritance



Here we have used inheritance through dependency injection.

1. Interfaces

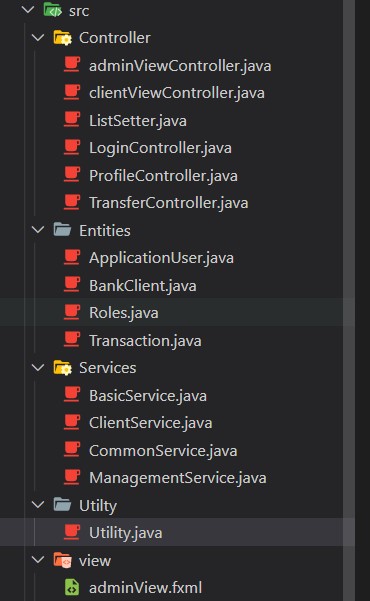




Here the first screenshot shows the use of generic interface for converting a ArrayList of objects to ArrayList of string.

The second screenshot shows the use of UtilityFunctions interface implemented by Utility class.

1. Packages



In our project we have used five packages- Controller, Entities, Services, Utility, view

Controller holds all classes and interfaces for functioning of each view screen.

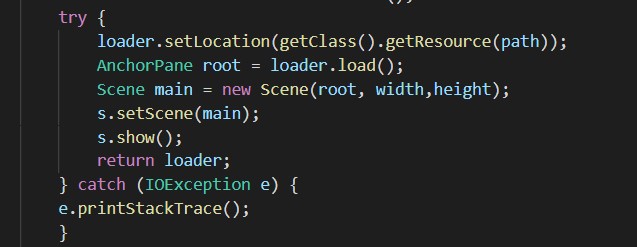
Entities holds all the objects used

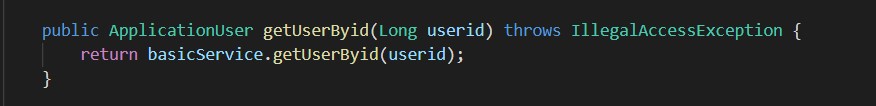
Services provides service to the controllers

Utility provides support to the services and controllers

View contains the .fxml files which determine the view screen.

1. Exception Handling



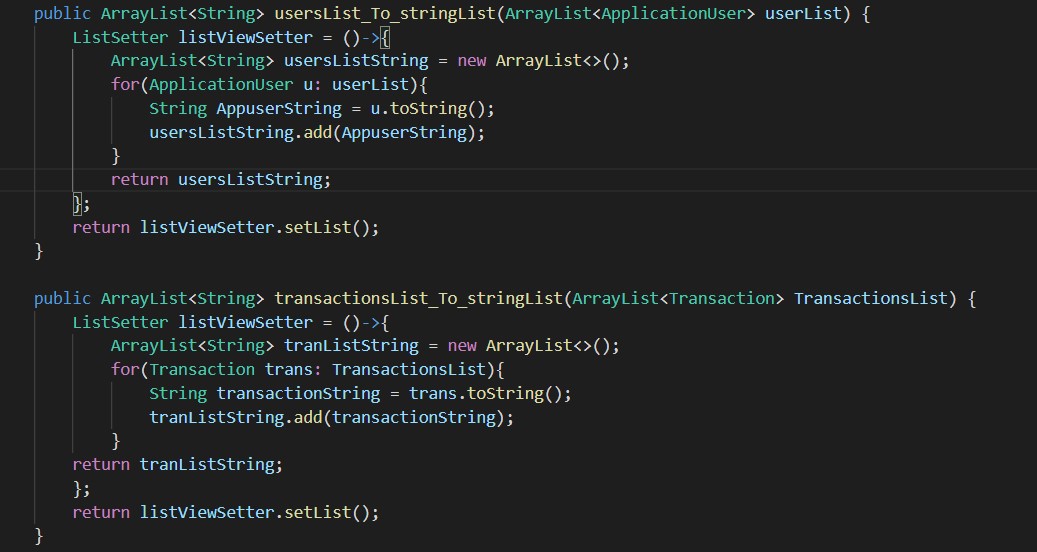




The above screenshots show the different methods in exception handling.

Try-catch, throws, throw

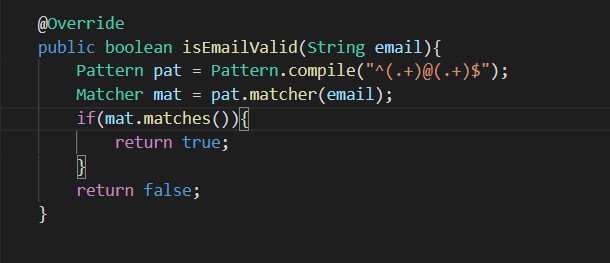
1. Lambda expressions



The above screenshot shows the use of lambda expressions 1. To convert ArrayList of ApplicationUser to ArrayList of string, 2. To convert ArrayList of Transaction to ArrayList of string.

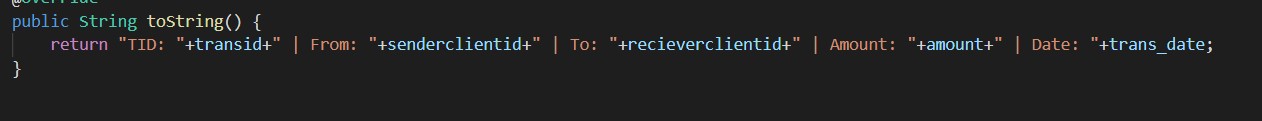
The generic interface for these lambda expression was shown in the interfaces section.

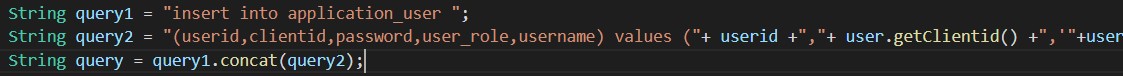
1. Regex



Here regex is used to validate email format.

1. Strings

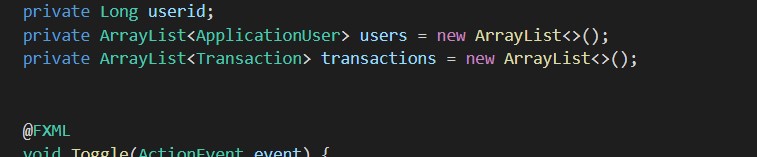




In the first screenshot we have overrided toString method for the Transaction object.

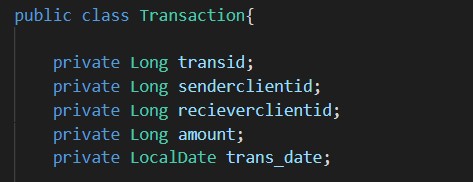
In the second screenshot we have used string function concat to combine two strings query1 and query2 to give a string query.

1. Collections



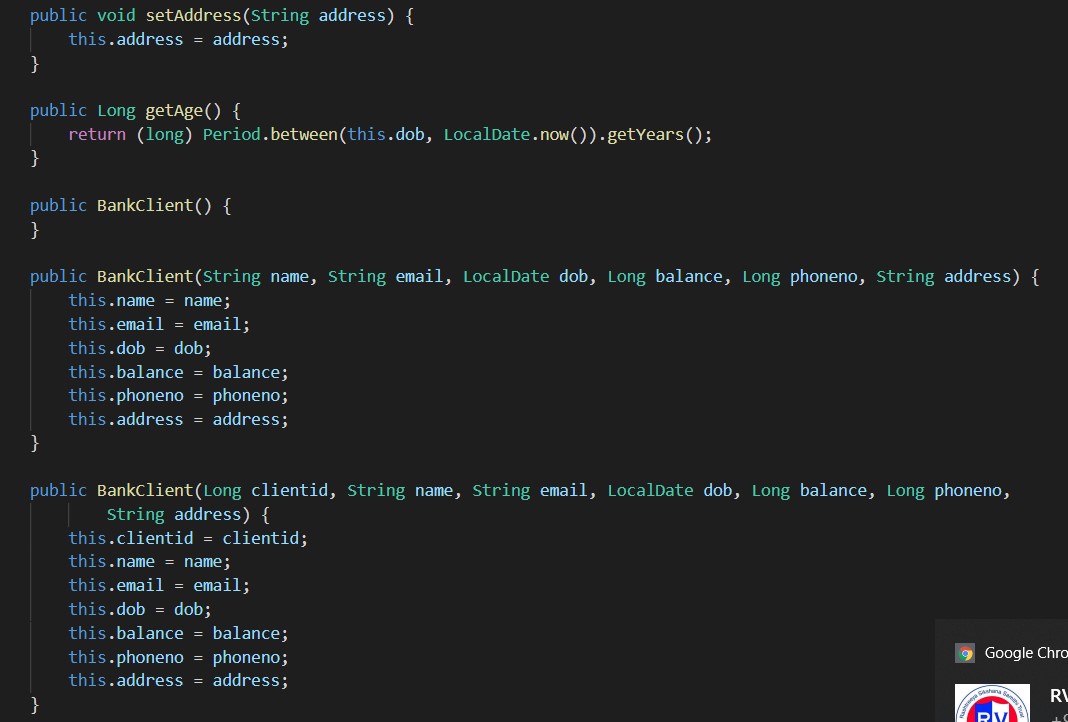
We have used collections extensively, here we have used ArrayList of type ApplicationUser and ArrayList of type Transaction.

1. Access specifiers



Here we have the five fields in the Transaction class, all have the private access specifier inorder to protect data.

1. Constructors and getters and setter functions



Here we show some of getter and setter functions in the BankClient class( setAddress(), getAge() ).

We have also used Constructor Overloading to set data provided in different ways.

## References and Source Code

The source code of our project can be found at drive linked below:

https://drive.google.com/drive/folders/1dSUdN1GzyvAk3VYon-DjixA18-Urv5Mo?usp=sharing