AMBERSON

HIGH SCHOOL

Project Report

ON

Banking System

Submitted to

AMBERSON HIGH SCHOOL

in partial fulfilment of the requirements for the final project of

ICS3U

Submitted By		Supervised By
Yiduo Cai	600-609-673	Mrs. Namarta Vij
Shuhan Yu	160-791-125	
Ruolin Jiang	158-400-135	
Mingxi Luo		

COMPUTER SCIENCE

AMBERSON HIGH SCHOOL

MARKHAM

[MARCH, 4th]

To whomsoever it may concern

This is to certify that the project report entitled “Traffic Light Management System” was conducted by Shuhan Yu, Yiduo Cai and Ruolin Jiang , both of whom are officially registered students of Amberson High School, completed under my guidance and supervision. They submitted this project in partial fulfillment of the requirements for the degree of Amberson High School.

Their dissertation represents the original work and is worthy of consideration for the award of the credit of Computer Science.

Declaration

I, “Yiduo Cai, Shuhan Yu, Ruolin Jiang ”, hereby declare that the work presented herein is genuine work done originally by me and has not been published or submitted elsewhere for the requirement of a degree program. Any literature, data or works done by others and cited within this dissertation has been given due acknowledgement listed in the reference section.

Yiduo Cai

Shuhan Yu

Ruolin Jiang

Mingxi Luo

Reg No. 600-609-673. 160-791-125. 158-400-135

Date: 2025.2.28

Abstract

This project creates a traffic light management system that will simulate the functionality of a traffic light at an intersection. The system will allow simulation of traffic light cycles, traffic flow, and monitoring of traffic at various intersections. This project will be implemented in Java and will cover concepts such as arrays, ArrayList, classes, inheritance, and encapsulation.

KEY WORDS

Java, Html, Arrays, ArrayList, Classes, Inheritance, and Encapsulation

Table of Content

Chapter-1

Introduction about java

In our project, the main programming language used is Java. Java is a "write once, run anywhere" programming language. It is a class-based object-oriented programming language designed to minimize implementation dependencies. Java was developed by James Gosling in May 1995 and later acquired by Oracle Corporation. It is like a "master key" that can easily adapt to different devices such as computers, mobile phones, servers and even smart home appliances, thanks to its core mechanism - the Java Virtual Machine (JVM), which can translate code into instructions that can be understood by various devices.

Java is known for its simplicity, security and stability. It supports modular development, making code as easy to reuse and maintain as building blocks; it also comes with an "automatic garbage collection" function to avoid memory leaks. Whether it is an App on a mobile phone, a bank's backend system, or an enterprise-level website and big data processing, Java is the main development language. Millions of developers around the world choose it not only because of its powerful functions, but also because of its rich free tools and active community support.

Chapter-1

Introduction to Eclipse

The software we use in the process of writing code is called Eclipse. Eclipse is often used as a teaching tool in computer courses in colleges and universities. Its clear engineering structure and debugging functions are conducive to algorithm practice. Financial institutions and telecom operators also often use Eclipse because it has a highly constructive and reliable system. Eclipse is an open source, cross-platform integrated development environment (IDE). It has become a multi-language programming tool widely used by developers around the world. It is suitable for multiple scenarios such as academic research, commercial applications and open source projects.

The core design of Eclipse is to simplify complexity. When you enter code, it is like a vigilant assistant, which can automatically complete a part of repeated code, check real-time errors, and immediately underline the errors without having to search hundreds of lines to modify missing brackets. It is also a good debugging tool that allows you to pause code execution, check variables, and gradually fix errors.

Secondly, what really stands out about Eclipse is its flexibility. Eclipse Marketplace offers thousands of plugins that turn the IDE into a cockpit tailored to your project needs. Popular add-ons turn complex tasks into simple clicks. Including the ability to work with Python and HTML.

Finally, Eclipse has strong cross-platform and collaborative capabilities. It can run on Windows, Linux, and macOS systems while maintaining functional consistency and supporting team collaborative development mode.

Chapter-2

Configuration used in project

Debbie Cai
Hardware Configuration:
Processor: .1GHz dual-core Intel Core i3, Turbo Boost up to 3.2GHz, with 4MB L3 cache
RAM: 16.0GB (15.5GM usable)
Hardware UUID: F3EE01D1-0585-593D-A460-BF22E8D66677
Provisioning UDID: 00008122-001479180A80001C
Types of Peripheral Devices:
A peripheral device is an internal or external hardware component that connects to a computer to extend its capabilities. Peripheral devices can be purchased after manufacturing to provide input/output (I/O) functions or improve the way the computer carries out specific processing or storage tasks. The types of peripheral: Input peripherals, Output peripherals, Storage peripherals, Net-working peripherals The peripheral I used most: keyboard and monitor.

Shuhan Yu
Hardware Configuration:
Processor: 1GHz dual-core Intel Core i3, Turbo Boost up to 3.2GHz, with 4MB L3 cache
RAM: 16.0GB (15.5GM usable)
Hardware UUID: F2B3D668-75A5-5540-B837-FB1C98DEBFC7
Provisioning UDID: 00008112-000A28D0343BC01E
Types of Peripheral Devices:
A peripheral device is an internal or external hardware component that connects to a computer to extend its capabilities. Peripheral devices can be purchased after manufacturing to provide input/output (I/O) functions or improve the way the computer carries out specific processing or storage tasks. The types of peripheral: Input peripherals, Output peripherals, Storage peripherals, Net-working peripherals The peripheral I used most: keyboard and monitor.

Mingxi Luo

Hardware Configuration:

Processor: 1GHz dual-core Intel Core i3, Turbo Boost up to 3.2GHz, with 4MB L3 cache
RAM: 16.0GB (15.5GM usable)
Hardware UUID: 6C7A02B0-B5BC-5A6C-86C6-A0B3596399A0
Provisioning UDID: 00008112-000E64163621401E

Types of Peripheral Devices:
A peripheral device is an internal or external hardware component that connects to a computer to extend its capabilities. Peripheral devices can be purchased after manufacturing to provide input/output (I/O) functions or improve the way the computer carries out specific processing or storage tasks. The types of peripheral: Input peripherals, Output peripherals, Storage peripherals, Net-working peripherals The peripheral I used most: keyboard and monitor.

Ruolin Jiang

Hardware Configuration:

Processor: 1GHz dual-core Intel Core i3, Turbo Boost up to 3.2GHz, with 4MB L3 cache

RAM: 16.0GB (15.5GM usable)

Hardware UUID: 0080E53A-532A-52E5-9D53-3190BA16C0A0

Provisioning UDID: 00008122-0008715401F1001C

Types of Peripheral Devices:
A peripheral device is an internal or external hardware component that connects to a computer to extend its capabilities. Peripheral devices can be purchased after manufacturing to provide input/output (I/O) functions or improve the way the computer carries out specific processing or storage tasks. The types of peripheral: Input peripherals, Output peripherals, Storage peripherals, Net-working peripherals The peripheral I used most: keyboard and monitor.

Chapter-3

Project Overview

The Banking System project is designed to simulate a basic banking application where users can create accounts, manage funds, and view transaction history. The system is implemented in Java using Object-Oriented Programming (OOP) principles and various data structures (Arrays, ArrayLists, etc.). Additionally, a simple HTML webpage is included to provide an overview of the banking system.

Introduction

The coding part of Java is divided into four modules, each assigned to a student for collaborative development. The first part is Account Management which allows users to create new bank accounts for depositing, withdrawing, or transferring funds. The second part is User Management. It collects some information about the user like user ID, password, email, address, and contact information. Users can sign up or login in to check the user's information details. The third part is Transaction History & Reporting. It records all the transaction histories and allows users to view the transaction history. The last part is Admin Management. It also stores users' information. It can view all accounts, block users, and manage or view transactions.

Web Page for Banking System (HTML)

A simple HTML webpage provides an overview of the system with the following features:

A complementary static webpage demonstrates front-end skills:

1. Home Page: Title, introductory paragraph, and banking-themed image.
2. Account Overview Table: Displays sample accounts with status (Active/Blocked).
3. User Account Section: Explains features like balance checks and sign-up links.
4. Admin Section: Describes administrative controls.

HTML Elements: Proper use of headings, tables, images, links, and background colors.

Real-World Relevance

This project mimics industry-standard practices, preparing students for software development roles by combining backend logic (Java) with front-end presentation (HTML). It reinforces critical skills like teamwork (module division), debugging, and user-centered design.

By completing this project, students demonstrate readiness for advanced programming challenges while adhering to professional coding standards.

Explanation to project

Our program is a simple system that can provide some bank services, such as creating an account, depositing, withdrawing, and transferring accounts.

The program was divided into four parts. When you start it, it will give the user a list of the functions, and you can choose them or exit running with your input. The program was divided into four different parts, and each part had its own type of service.

In the first part, users need to create a new account, and the bank system can help you to deposit or draw your money from your account and transfer your money to another account. These are the basic functions of bank services.

The second part is used to manage personal information, it will record users' personal information, store them, and verify users' identity before they log in to the system. It is an important part of strengthening security. Also, users can check and change their information every time.

The third part is used to store transaction history information, such as transaction type, date, and amount of money. It can also help users to calculate the total deposit or withdrawal amount. Users can enter their account ID, and check all of this information in this program.

The last part is used to manage information of this system, and the normal users are limited to using it. The administrator can check all of the accounts which are stored, and block a user to prevent him from using the system. This function can help the administrator check Illegal users, and find all of the information and transaction history.

Chapter 4

Features and Functionality:

This project combines a Java backend and a static HTML frontend to create a functional banking system. Below are the key features and functionalities organized by module.

1. Account Management

Account Creation: Create new accounts with unique IDs, holder names, and types (e.g., Savings, Checking).

Balance Operations:Deposit funds with validation (e.g., prevent negative amounts).Withdraw funds with overdraft protection (balance cannot go below $0).

Fund Transfers: Transfer money between accounts using account IDs. Validate source and destination accounts before processing transactions.

2. User Management

User Registration: Register new users with unique IDs, usernames, passwords, and emails. Store users in an ArrayList<User> or HashMap.

User Authentication: Login with username/password credentials. Basic password validation (e.g., minimum length).

Profile Management: View and update user details (e.g., email, contact information).

3. Transaction History & Reporting

Transaction Logging: Automatically record deposits, withdrawals, and transfers in an ArrayList<Transaction>. Track transaction details: type, amount, date, and associated account.

Customizable Reports: Filter transactions by type (deposit/withdrawal/transfer), date range, or amount. Sort transactions chronologically.

Audit Trail: Generate reports for users or admins to review financial activity.

4. Admin Management

Account Oversight:View all accounts, including balances and statuses (Active/Blocked).

User Moderation: Block/unblock users to restrict system access.

Transaction Oversight: Review and audit all transactions across accounts.

Access Control: Admins inherit from the User class but have elevated privileges (e.g., blocking users).

Chapter 4

Challenge and Solutions:

Challenge 1: Understanding Class Structures

Question: What are the main attributes of the Account class? What is their purpose?

Solution: The Account class has the following attributes. AccountID is for storing the unique identifier for the account. AccountHolderName is for storing the name of the account holder. Balance is for storing the amount of money in the account. AccountType is for defining whether the account is savings, checking, etc. Each attribute helps define an account uniquely and is used in various methods like deposit and withdrawal.

Challenge 5: Identifying Possible Errors

Question: What happens if a user enters a username that does not exist during login?

Solution: If the username is not found in the NAME list, the program should output a message like: "User not found. Please try again." If no such check is implemented, the program may cause an IndexOutOfBoundsException when trying to find the password at a non-existent index.

Challenge: Dynamic Table Generation

Question: We don’t know how to generate a table dynamically with the input from the user.

Solution: We looked up the tutorial from the html website (https://www.w3schools.com/html/) and learned how to do it.

Chapter-5

HTML Implementation:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="background-color:CadetBlue;">Bank Website</h1>
<p>Our bank offers include account creation, loans, deposits.
<img src="./images-5.png" alt="Trulli" width="400" height="333">

</body>
</html>

<html>
<body>
<style>
table, th, td {
  border: 1px solid white;
  border-collapse: collapse;
}
th, td {
  background-color: #96D4D4;
}
</style>
</body>
</html>

<html>
<body>
<h2 style="background-color:CadetBlue;">Account Overview</h2>
    <form id="accountForm">
        <label for="accountId">Account ID:</label><br>
        <input type="text" id="accountId" required><br>

        <label for="accountName">Account Holder Name:</label><br>
        <input type="text" id="accountName" required><br>

        <label for="balance">Balance:</label><br>
        <input type="number" id="balance" required><br>

        <label for="accountType">Account Type:</label><br>
        <input type="text" id="accountType" required><br>

        <label for="status">Status:</label><br>
        <input type="text" id="status" required><br>

        <button type="button" onclick="addAccount()">Add Account</button>
    </form>
```

```
    <h3 style="color:CadetBlue;">Account List</h3>
        <table id="accountTable">
            <thead>
                <tr>
                    <th>Account ID</th>
                    <th>Account Holder Name</th>
                    <th>Balance</th>
                    <th>Account Type</th>
                    <th>Status</th>
                </tr>
            </thead>
            <tbody>

            </tbody>
        </table>
</body>
</html>

<html>
<body>

<h2 style="background-color:CadetBlue;">User Account Section</h2>
<p>In this page, users can <b>create accounts, check balances, and view tran
history</b>.</p >
<a href=" ">Sign Up</a >

</body>
</html>

<html>
<body>
<h2 style="background-color:CadetBlue;">User Account Section</h2>
<p>admins can <b>view, block</b> users, and manage transactions</p >
```

```
<script>
        function addAccount() {

            const accountId = document.getElementById('accountId').value;
            const accountName = document.getElementById('accountName').value;
            const balance = document.getElementById('balance').value;
            const accountType = document.getElementById('accountType').value;
            const status = document.getElementById('status').value;



            const tableBody =
document.getElementById('accountTable').getElementsByTagName('tbody')[0];


            const newRow = tableBody.insertRow();


            newRow.insertCell(0).textContent = accountId;
            newRow.insertCell(1).textContent = accountName;
            newRow.insertCell(2).textContent = balance;
            newRow.insertCell(3).textContent = accountType;
            newRow.insertCell(4).textContent = status;


            document.getElementById('accountForm').reset();
        }
</script>


<style>
    body {
        background-color: AliceBlue;
    }
</style>

</body>
</html>
```

Webpage

https://bank-mangement.w3spaces.com/saved-from-Tryit-2025-03-05-b88q0.html

Bank Website

Our bank offers include account creation, loans, deposits. Trulli

Account Overview

Account ID:

Account Holder Name:

Balance:

Account Type:

Status:

Add Account

Account List

Account ID	Account Holder Name	Balance	Account Type	Status
123	Mike	1000	savings	Active
111	John	500	checking account	Blocked

User Account Section

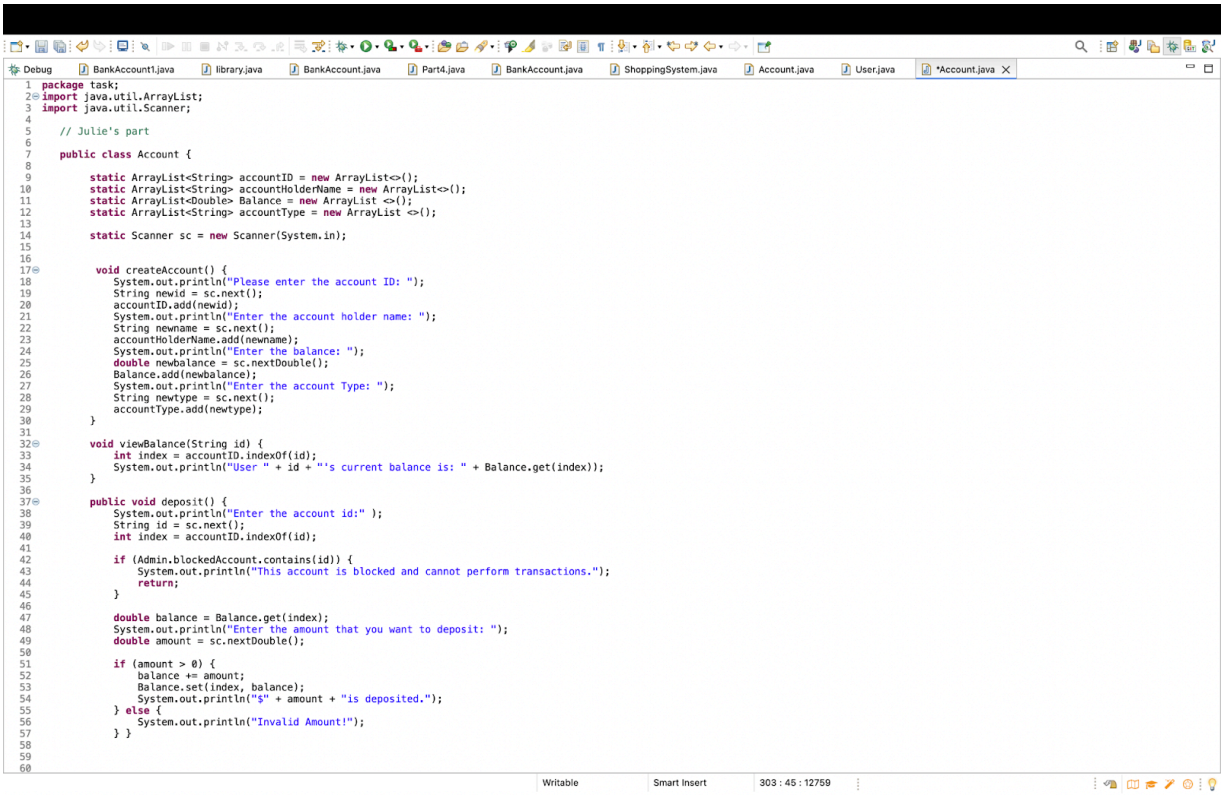
In this page, users can **create accounts, check balances, and view transaction history**.

Sign Up

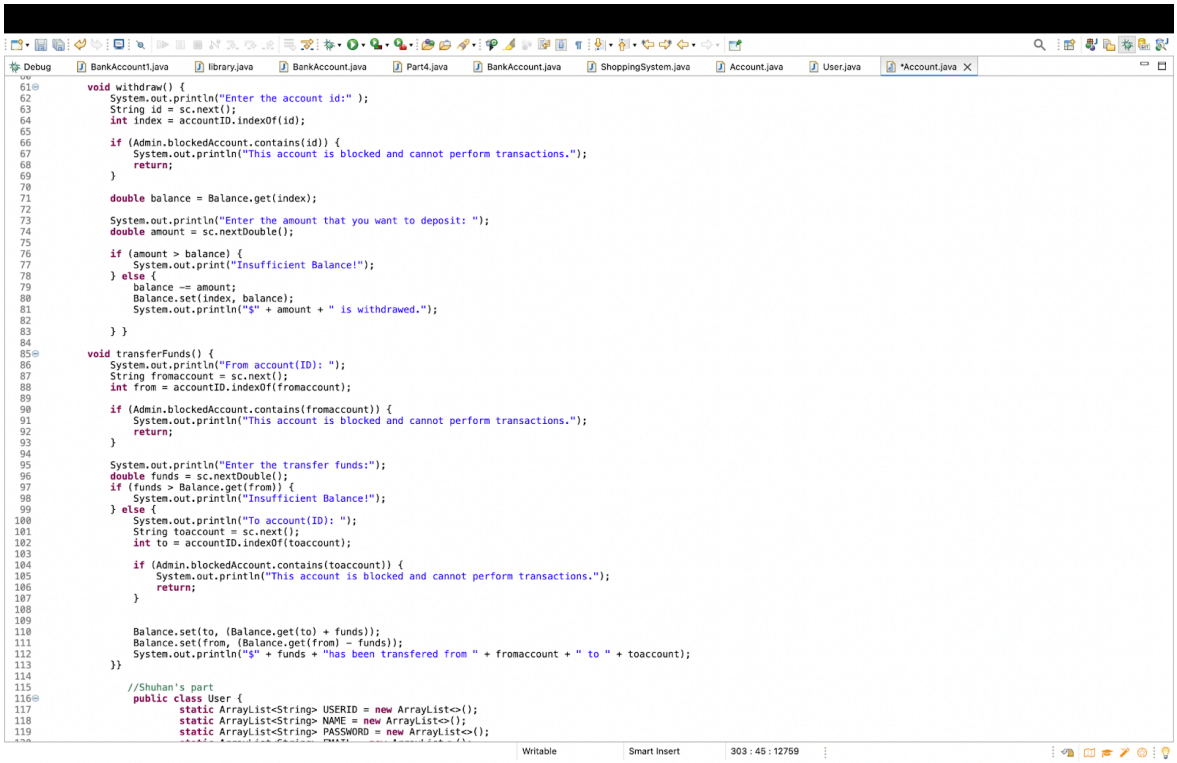
User Account Section

admins can **view, block** users, and manage transactions

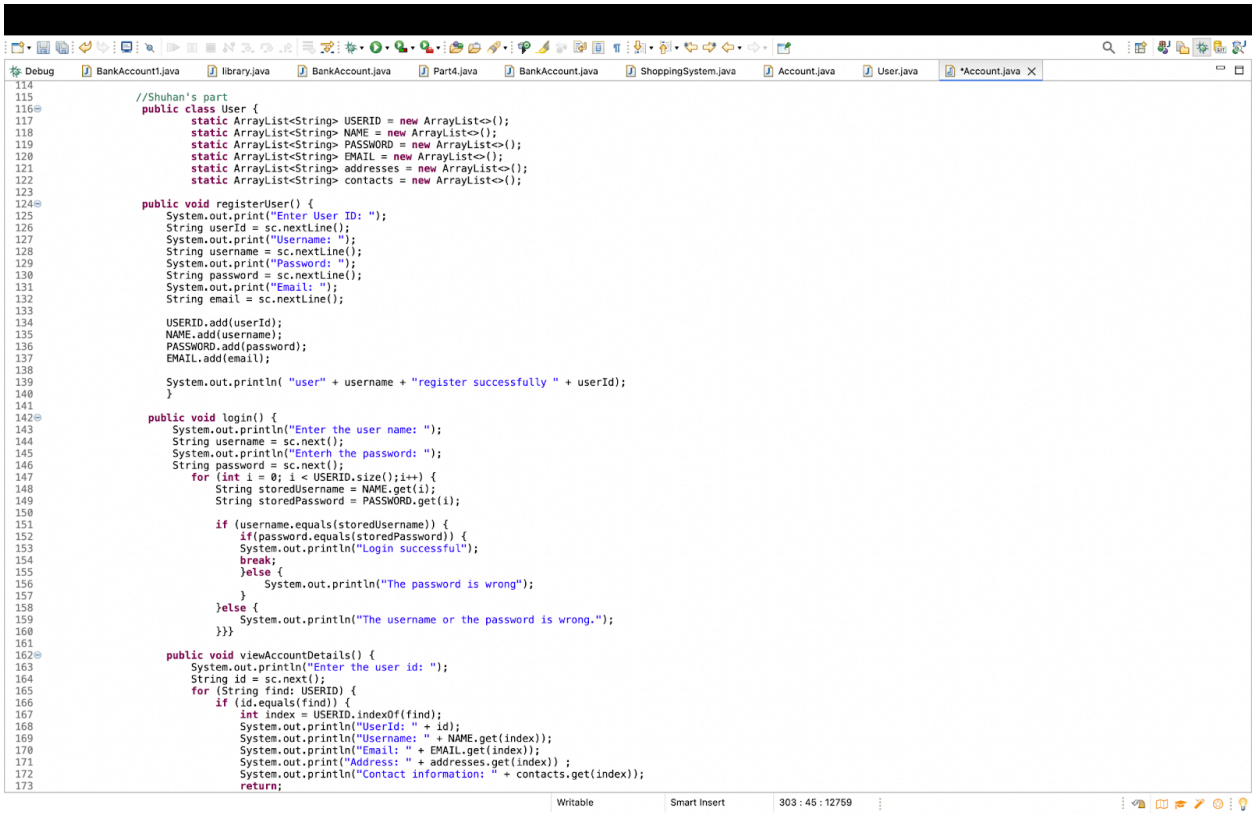
Code:



```java
package task;
import java.util.ArrayList;
import java.util.Scanner;

   // Julie's part

   public class Account {

        static ArrayList<String> accountID = new ArrayList<>();
        static ArrayList<String> accountHolderName = new ArrayList<>();
        static ArrayList<Double> Balance = new ArrayList <>();
        static ArrayList<String> accountType = new ArrayList <>();

        static Scanner sc = new Scanner(System.in);


         void createAccount() {
            System.out.println("Please enter the account ID: ");
            String newid = sc.next();
            accountID.add(newid);
            System.out.println("Enter the account holder name: ");
            String newname = sc.next();
            accountHolderName.add(newname);
            System.out.println("Enter the balance: ");
            double newbalance = sc.nextDouble();
            Balance.add(newbalance);
            System.out.println("Enter the account Type: ");
            String newtype = sc.next();
            accountType.add(newtype);
        }

        void viewBalance(String id) {
            int index = accountID.indexOf(id);
            System.out.println("User " + id + "'s current balance is: " + Balance.get(index));
        }

        public void deposit() {
            System.out.println("Enter the account id:" );
            String id = sc.next();
            int index = accountID.indexOf(id);

            if (Admin.blockedAccount.contains(id)) {
                System.out.println("This account is blocked and cannot perform transactions.");
                return;
            }

            double balance = Balance.get(index);
            System.out.println("Enter the amount that you want to deposit: ");
            double amount = sc.nextDouble();

            if (amount > 0) {
                balance += amount;
                Balance.set(index, balance);
                System.out.println("$" + amount + "is deposited.");
            } else {
                System.out.println("Invalid Amount!");
            } }
```

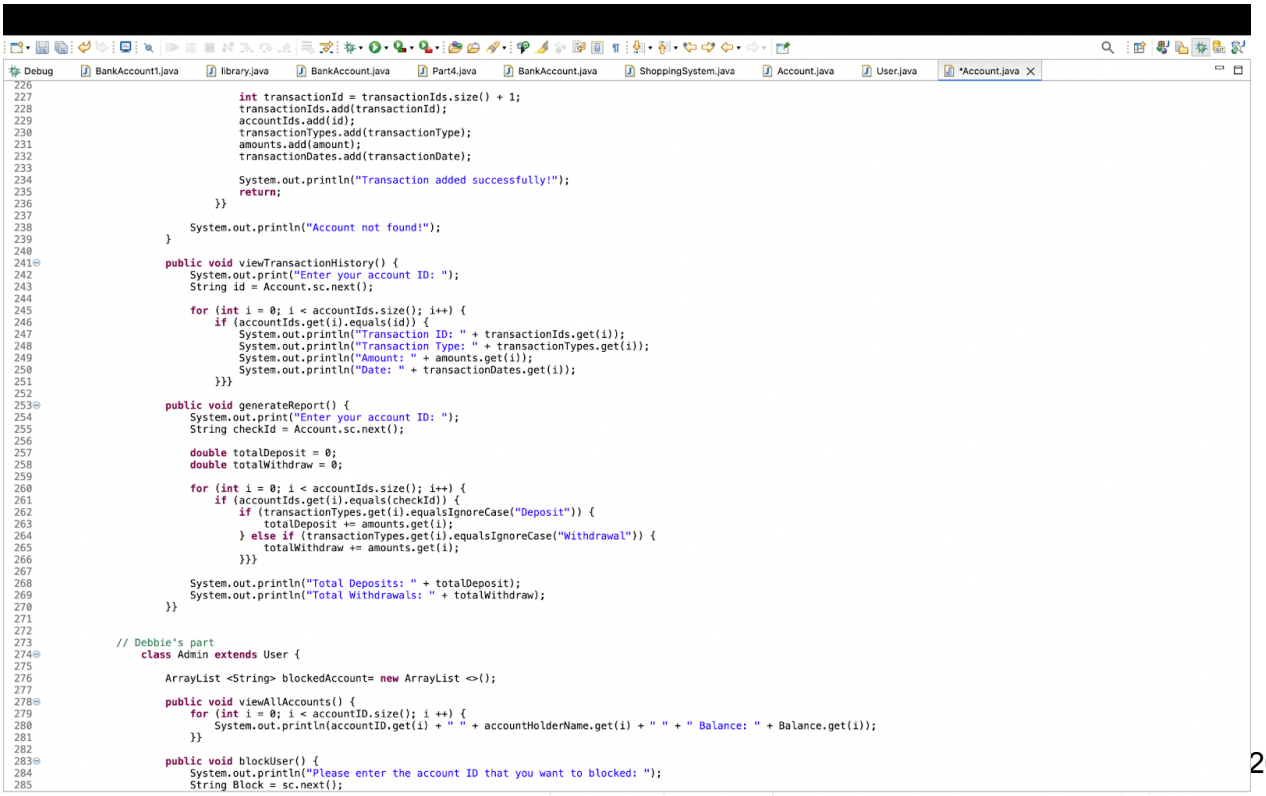


```java
        void withdraw() {
            System.out.println("Enter the account id:" );
            String id = sc.next();
            int index = accountID.indexOf(id);

            if (Admin.blockedAccount.contains(id)) {
                System.out.println("This account is blocked and cannot perform transactions.");
                return;
            }

            double balance = Balance.get(index);

            System.out.println("Enter the amount that you want to deposit: ");
            double amount = sc.nextDouble();

            if (amount > balance) {
                System.out.print("Insufficient Balance!");
            } else {
                balance -= amount;
                Balance.set(index, balance);
                System.out.println("$" + amount + " is withdrawed.");

            } }

        void transferFunds() {
            System.out.println("From account(ID): ");
            String fromaccount = sc.next();
            int from = accountID.indexOf(fromaccount);

            if (Admin.blockedAccount.contains(fromaccount)) {
                System.out.println("This account is blocked and cannot perform transactions.");
                return;
            }

            System.out.println("Enter the transfer funds:");
            double funds = sc.nextDouble();
            if (funds > Balance.get(from)) {
                System.out.println("Insufficient Balance!");
            } else {
                System.out.println("To account(ID): ");
                String toaccount = sc.next();
                int to = accountID.indexOf(toaccount);

                if (Admin.blockedAccount.contains(toaccount)) {
                    System.out.println("This account is blocked and cannot perform transactions.");
                    return;
                }


                Balance.set(to, (Balance.get(to) + funds));
                Balance.set(from, (Balance.get(from) - funds));
                System.out.println("$" + funds + "has been transfered from " + fromaccount + " to " + toaccount);
            }}

               //Shuhan's part
                public class User {
                        static ArrayList<String> USERID = new ArrayList<>();
                        static ArrayList<String> NAME = new ArrayList<>();
                        static ArrayList<String> PASSWORD = new ArrayList<>();
```



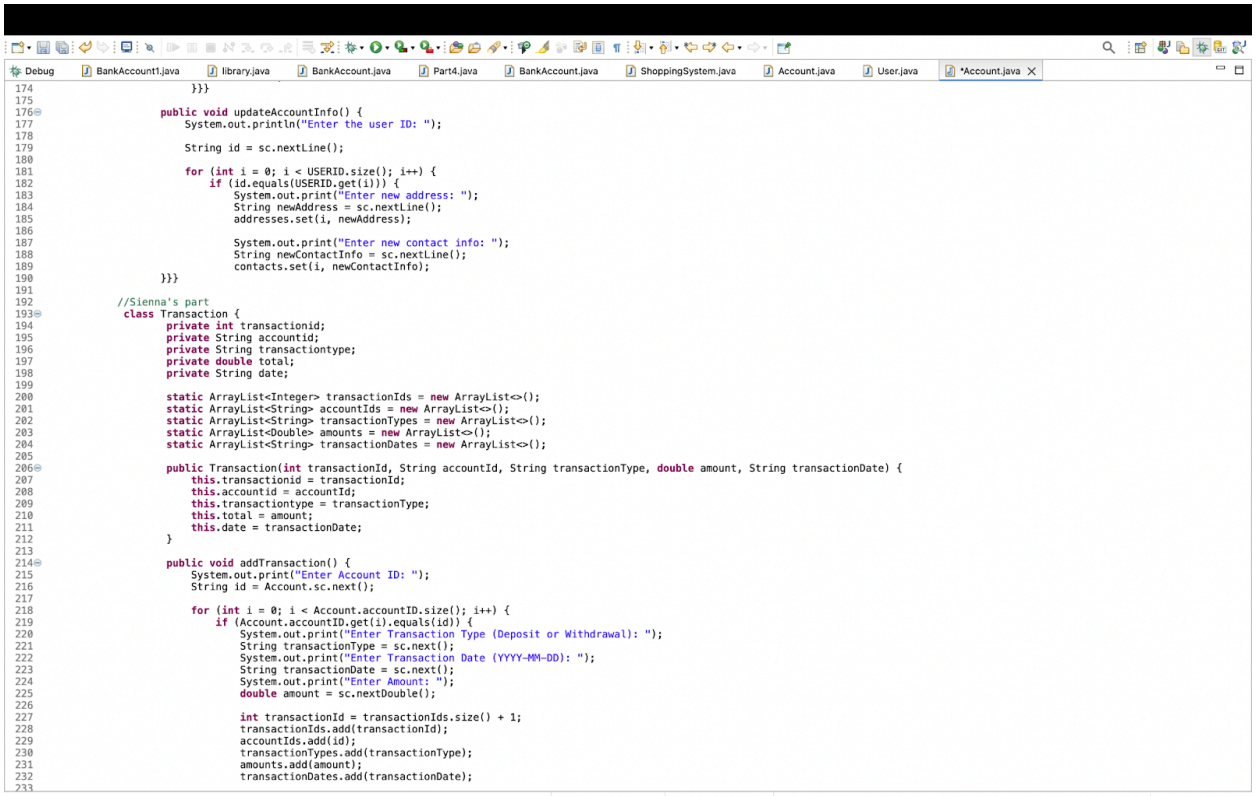
```
114
115             //Shuhan's part
116              public class User {
117                     static ArrayList<String> USERID = new ArrayList<>();
118                     static ArrayList<String> NAME = new ArrayList<>();
119                     static ArrayList<String> PASSWORD = new ArrayList<>();
120                     static ArrayList<String> EMAIL = new ArrayList<>();
121                     static ArrayList<String> addresses = new ArrayList<>();
122                     static ArrayList<String> contacts = new ArrayList<>();
123
124              public void registerUser() {
125                  System.out.print("Enter User ID: ");
126                  String userId = sc.nextLine();
127                  System.out.print("Username: ");
128                  String username = sc.nextLine();
129                  System.out.print("Password: ");
130                  String password = sc.nextLine();
131                  System.out.print("Email: ");
132                  String email = sc.nextLine();
133
134                  USERID.add(userId);
135                  NAME.add(username);
136                  PASSWORD.add(password);
137                  EMAIL.add(email);
138
139                  System.out.println( "user" + username + "register successfully " + userId);
140                  }
141
142               public void login() {
143                   System.out.println("Enter the user name: ");
144                   String username = sc.next();
145                   System.out.println("Enterh the password: ");
146                   String password = sc.next();
147                      for (int i = 0; i < USERID.size();i++) {
148                          String storedUsername = NAME.get(i);
149                          String storedPassword = PASSWORD.get(i);
150
151                          if (username.equals(storedUsername)) {
152                              if(password.equals(storedPassword)) {
153                              System.out.println("Login successful");
154                              break;
155                              }else {
156                                  System.out.println("The password is wrong");
157                              }
158                          }else {
159                              System.out.println("The username or the password is wrong.");
160                          }}}
161
162                  public void viewAccountDetails() {
163                      System.out.println("Enter the user id: ");
164                      String id = sc.next();
165                      for (String find: USERID) {
166                          if (id.equals(find)) {
167                              int index = USERID.indexOf(find);
168                              System.out.println("UserId: " + id);
169                              System.out.println("Username: " + NAME.get(index));
170                              System.out.println("Email: " + EMAIL.get(index));
171                              System.out.print("Address: " + addresses.get(index)) ;
172                              System.out.println("Contact information: " + contacts.get(index));
173                              return;
```

Writable | Smart Insert | 303 : 45 : 12759



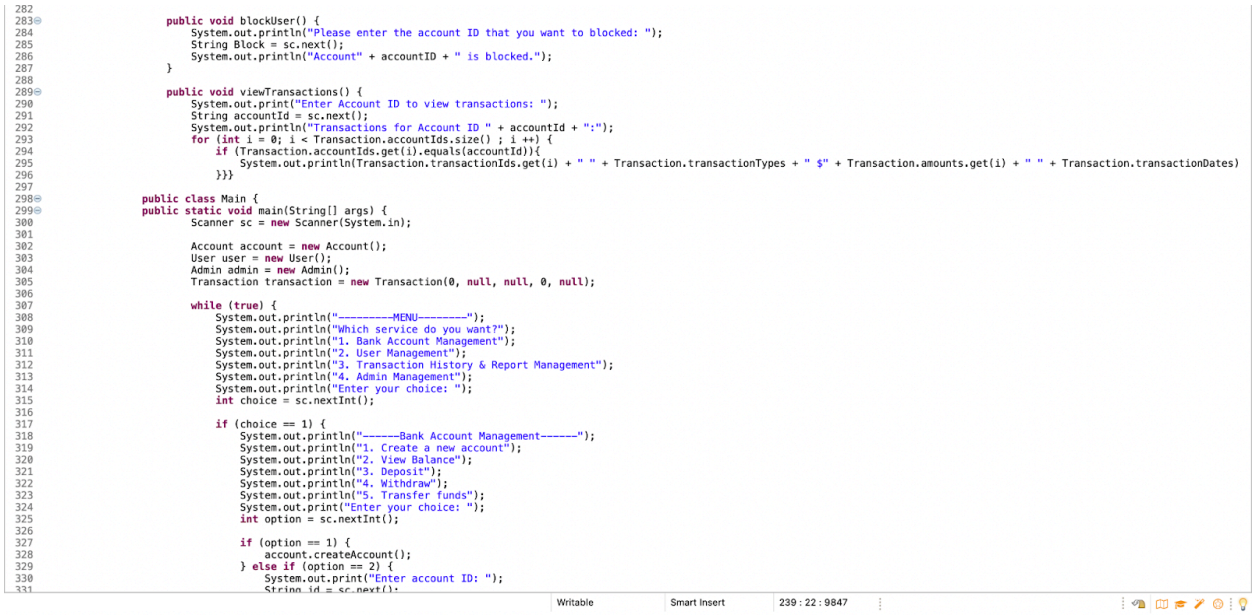
```
226
227                             int transactionId = transactionIds.size() + 1;
228                             transactionIds.add(transactionId);
229                             accountIds.add(id);
230                             transactionTypes.add(transactionType);
231                             amounts.add(amount);
232                             transactionDates.add(transactionDate);
233
234                             System.out.println("Transaction added successfully!");
235                             return;
236                         }}
237
238                     System.out.println("Account not found!");
239                 }
240
241                 public void viewTransactionHistory() {
242                     System.out.print("Enter your account ID: ");
243                     String id = Account.sc.next();
244
245                     for (int i = 0; i < accountIds.size(); i++) {
246                         if (accountIds.get(i).equals(id)) {
247                             System.out.println("Transaction ID: " + transactionIds.get(i));
248                             System.out.println("Transaction Type: " + transactionTypes.get(i));
249                             System.out.println("Amount: " + amounts.get(i));
250                             System.out.println("Date: " + transactionDates.get(i));
251                         }}}
252
253                 public void generateReport() {
254                     System.out.print("Enter your account ID: ");
255                     String checkId = Account.sc.next();
256
257                     double totalDeposit = 0;
258                     double totalWithdraw = 0;
259
260                     for (int i = 0; i < accountIds.size(); i++) {
261                         if (accountIds.get(i).equals(checkId)) {
262                             if (transactionTypes.get(i).equalsIgnoreCase("Deposit")) {
263                                 totalDeposit += amounts.get(i);
264                             } else if (transactionTypes.get(i).equalsIgnoreCase("Withdrawal")) {
265                                 totalWithdraw += amounts.get(i);
266                             }}}
267
268                     System.out.println("Total Deposits: " + totalDeposit);
269                     System.out.println("Total Withdrawals: " + totalWithdraw);
270                 }}
271
272
273             // Debbie's part
274                 class Admin extends User {
275
276                     ArrayList <String> blockedAccount= new ArrayList <>();
277
278                     public void viewAllAccounts() {
279                         for (int i = 0; i < accountID.size(); i ++) {
280                             System.out.println(accountID.get(i) + " " + accountHolderName.get(i) + " " + " Balance: " + Balance.get(i));
281                         }}
282
283                     public void blockUser() {
284                         System.out.println("Please enter the account ID that you want to blocked: ");
285                         String Block = sc.next();
```

Writable | Smart Insert | 303 : 45 : 12759



```
                    }}}

                public void updateAccountInfo() {
                    System.out.println("Enter the user ID: ");

                    String id = sc.nextLine();

                    for (int i = 0; i < USERID.size(); i++) {
                        if (id.equals(USERID.get(i))) {
                            System.out.print("Enter new address: ");
                            String newAddress = sc.nextLine();
                            addresses.set(i, newAddress);

                            System.out.print("Enter new contact info: ");
                            String newContactInfo = sc.nextLine();
                            contacts.set(i, newContactInfo);
                }}}

            //Sienna's part
             class Transaction {
                    private int transactionid;
                    private String accountid;
                    private String transactiontype;
                    private double total;
                    private String date;

                    static ArrayList<Integer> transactionIds = new ArrayList<>();
                    static ArrayList<String> accountIds = new ArrayList<>();
                    static ArrayList<String> transactionTypes = new ArrayList<>();
                    static ArrayList<Double> amounts = new ArrayList<>();
                    static ArrayList<String> transactionDates = new ArrayList<>();

                    public Transaction(int transactionId, String accountId, String transactionType, double amount, String transactionDate) {
                        this.transactionid = transactionId;
                        this.accountid = accountId;
                        this.transactiontype = transactionType;
                        this.total = amount;
                        this.date = transactionDate;
                    }

                    public void addTransaction() {
                        System.out.print("Enter Account ID: ");
                        String id = Account.sc.next();

                        for (int i = 0; i < Account.accountID.size(); i++) {
                            if (Account.accountID.get(i).equals(id)) {
                                System.out.print("Enter Transaction Type (Deposit or Withdrawal): ");
                                String transactionType = sc.next();
                                System.out.print("Enter Transaction Date (YYYY-MM-DD): ");
                                String transactionDate = sc.next();
                                System.out.print("Enter Amount: ");
                                double amount = sc.nextDouble();

                                int transactionId = transactionIds.size() + 1;
                                transactionIds.add(transactionId);
                                accountIds.add(id);
                                transactionTypes.add(transactionType);
                                amounts.add(amount);
                                transactionDates.add(transactionDate);
```

Writable | Smart Insert | 303 : 45 : 12759



```

                    public void blockUser() {
                        System.out.println("Please enter the account ID that you want to blocked: ");
                        String Block = sc.next();
                        System.out.println("Account" + accountID + " is blocked.");
                    }

                    public void viewTransactions() {
                        System.out.print("Enter Account ID to view transactions: ");
                        String accountId = sc.next();
                        System.out.println("Transactions for Account ID " + accountId + ":");
                        for (int i = 0; i < Transaction.accountIds.size() ; i ++) {
                            if (Transaction.accountIds.get(i).equals(accountId)){
                                System.out.println(Transaction.transactionIds.get(i) + " " + Transaction.transactionTypes + " $" + Transaction.amounts.get(i) + " " + Transaction.transactionDates)
                            }}}

                public class Main {
                public static void main(String[] args) {
                        Scanner sc = new Scanner(System.in);

                        Account account = new Account();
                        User user = new User();
                        Admin admin = new Admin();
                        Transaction transaction = new Transaction(0, null, null, 0, null);

                        while (true) {
                            System.out.println("---------MENU--------");
                            System.out.println("Which service do you want?");
                            System.out.println("1. Bank Account Management");
                            System.out.println("2. User Management");
                            System.out.println("3. Transaction History & Report Management");
                            System.out.println("4. Admin Management");
                            System.out.println("Enter your choice: ");
                            int choice = sc.nextInt();

                            if (choice == 1) {
                                System.out.println("------Bank Account Management------");
                                System.out.println("1. Create a new account");
                                System.out.println("2. View Balance");
                                System.out.println("3. Deposit");
                                System.out.println("4. Withdraw");
                                System.out.println("5. Transfer funds");
                                System.out.print("Enter your choice: ");
                                int option = sc.nextInt();

                                if (option == 1) {
                                    account.createAccount();
                                } else if (option == 2) {
                                    System.out.print("Enter account ID: ");
                                    String id = sc.next();
```

Writable | Smart Insert | 239 : 22 : 9847

```
                    if (option == 1) {
                        account.createAccount();
                    } else if (option == 2) {
                        System.out.print("Enter account ID: ");
                        String id = sc.next();
                        account.viewBalance(id);
                    } else if (option == 3) {
                        account.deposit();
                    } else if (option == 4) {
                        account.withdraw();
                    } else if (option == 5) {
                        account.transferFunds();
                    }
                } else if (choice == 2) {
                    System.out.println("------User Management------");
                    System.out.println("1. Register User");
                    System.out.println("2. Login");
                    System.out.println("3. View Account Details");
                    System.out.println("4. Update Account Info");
                    System.out.print("Enter your choice: ");
                    int option = sc.nextInt();

                    if (option == 1) {
                        user.registerUser();
                    } else if (option == 2) {
                        user.login();
                    } else if (option == 3) {
                        user.viewAccountDetails();
                    } else if (option == 4) {
                     user.updateAccountInfo();
                    }
                } else if (choice == 3) {
                    System.out.println("------Transaction History & Report Management------");
                    System.out.println("1. Add Transaction");
                    System.out.println("2. View Transaction History");
                    System.out.println("3. Generate Report");
                    System.out.print("Enter your choice: ");
                    int option = sc.nextInt();

                    if (option == 1) {
                        transaction.addTransaction();
                    } else if (option == 2) {
                        transaction.viewTransactionHistory();
                    } else if (option == 3) {
                        transaction.generateReport();
                    }
```

```
                    }
                  } else if (choice == 4) {
                        System.out.println("------Admin Management------");
                        System.out.println("1. View All Accounts");
                        System.out.println("2. Block User");
                        System.out.println("3. View Transactions");
                        System.out.print("Enter your choice: ");
                        int option = sc.nextInt();

                        if (option == 1) {
                            admin.viewAllAccounts();
                        } else if (option == 2) {
                            admin.blockUser();
                        } else if (option == 3) {
                            admin.viewTransactions();
                        }
                  }
              }
          }
      }}}}
```

Chapter-6

Conclusion

In conclusion, the program using Java, HTML, and Eclipse, creates a bank management system with essential banking functions. We use Java to implement terminal logic to handle the management of accounts, users and transactions. HTML used to build front-end web pages, providing user interfaces and decorate with tables, images, links, and background colors. Moreover, Eclipse is used for writing, debugging, and running Java code.

The bank management system core functionalities include:

- Account management: creating accounts, deposit and withdraw funds, view balances and transferring funds.
- User management: registering, logging in, viewing and updating account information.
- Transaction management: recording and viewing transaction history, as well as generating transaction reports.
- Admin management: view all accounts, block users, and manage transactions.

Overall, the program provides an efficient and friendly method to help users and administrators to handle their accounts smoothly.

Chapter-7

Project Allocation

- Coding — Part 1 – Julie
  Part 2 – Shuhan
  Part 3 – Mingxi
  Part 4 – Debbie
  All -- Debbie

- HTML — Mingxi

- Report — Julie(Chapter 2, Chapter 6)
  Shuhan(Format, declaration, abstract, table of content, Chapter 1, Chapter2)
  Debbie(Chapter2, Chapter 3, Chapter 4, Chapter 5)
  Mingxi(Chapter 2, Chapter 3, Chapter 4)

- PPT — Shuhan

References