Object Oriented Programming System(2CS502)

ATM MANAGEMENT SYSTEM

Innovative Assignment - I

Prepared By:

Dhruvi Tanna

(23BCE507)

D1 Batch(D2D)

Introduction

An ATM (Automated Teller Machine) management system is a comprehensive software solution designed to efficiently and securely handle the operations, maintenance, and monitoring of a network of ATMs. The primary goal of such a system is to ensure the seamless functioning of ATMs, enhance user experience, and streamline administrative tasks for the financial institutions that deploy and manage these machines.

I have developed an ATM machine application using Java, providing a versatile and user-friendly solution for automated financial transactions. This Java-based ATM system seamlessly integrates core functionalities, allowing users to perform a variety of banking operations, including cash withdrawals, deposits, balance inquiries, fund transfers. The application is designed with a clean and intuitive user interface, making it accessible to a wide range of users.

Innovation:

In the context of our project, I'm pleased to highlight a series of noteworthy innovations incorporated into our codebase.As we embark on a closer examination of our code, it becomes evident that these innovations serve as important elements, elevating the project to a level of sophistication and efficiency that distinguishes it within the suitable of technological solutions.Let's look into the intricacies of these enhancements, exploring how they collectively contribute to the project's exceptional standing.

1. **HASH MAP:**

   In Java, a hash map is a data structure that implements the Map interface, allowing the storage and retrieval of key-value pairs. It is part of the Java Collections Framework and is particularly useful when you need fast access to data based on a specific key.

In this project I have used Hashmap for connecting the account of each individual with its account and perform the operations it is working as the in-memory database

2. **JAVA.TEXT PACKAGE:**

The JAVA.TEXT package in Java provides classes and interfaces for formatting and parsing text, particularly related to dates, numbers, and messages. It plays a crucial role in localization and internationalization of Java applications.

In this project i have java.text package for displaying the money format for the effective display of the money with every transaction done.

Java.text package includes several classes and interfaces like:

- Date Format
- Simple Date Format
- Number Format
- Decimal Format
- Message Format
- Choice Format

Some Basic Concepts used Are:

3. **CONSTRUCTOR:**

A constructor in Java is a special method that is called when an object of a class is created. It is used to initialize the object's state and perform any necessary setup.

Constructor help me to initialize my all variables using this keyword and i have used two parameterized constructor

4. IO EXCEPTIONS:

In Java, Input/Output (I/O) exceptions, also known as checked exceptions, are a category of exceptions that arise during input and output operations, such as reading from or writing to files, streams, or other external sources. These exceptions are checked at compile-time, and developers are required to handle or declare them in the method signature using the throws

5. GETTER AND SETTER METHODS:

In Java, getter and setter methods are commonly used to access and modify the private fields (variables) of a class. These methods follow the principles of encapsulation, providing controlled access to the internal state of an object.

Code:

Account.java file:

```
import java.text.*;
import java.util.*;

public class Account
{
	// variables
	int customerNumber;
	int pinNumber;
	double checkingBalance = 0;
	double savingBalance = 0;

	Scanner input = new Scanner(System.in);
	DecimalFormat moneyFormat = new DecimalFormat("'$'###,##0.00"); //output money
format

// constructor:
	public Account(int customerNumber, int pinNumber)
	{
		this.customerNumber = customerNumber;
		this.pinNumber = pinNumber;
	}

	//parameterized constructor
	public Account(int customerNumber, int pinNumber, double checkingBalance, double
savingBalance)
	{
		this.customerNumber = customerNumber;
		this.pinNumber = pinNumber;
		this.checkingBalance = checkingBalance;
		this.savingBalance = savingBalance;
	}

	//getter and setter method
	public int setCustomerNumber(int customerNumber)
	{
		this.customerNumber = customerNumber;
		return customerNumber;
	}

	public int getCustomerNumber()
	{
		return customerNumber;
```

```
}

public int setPinNumber(int pinNumber)
{
	this.pinNumber = pinNumber;
	return pinNumber;
}

public int getPinNumber()
{
	return pinNumber;
}

public double getCheckingBalance()
{
	return checkingBalance;
}

public double getSavingBalance()
{
	return savingBalance;
}

//method for withdrawing money and counting balance after it
public double calcCheckingWithdraw(double amount)
{
	checkingBalance = (checkingBalance - amount);
	return checkingBalance;
}

//method calculating savingbalance after savingwithdraw
public double calcSavingWithdraw(double amount)
{
	savingBalance = (savingBalance - amount);
	return savingBalance;
}

//method for calculating balance after deposit
public double calcCheckingDeposit(double amount)
{
	checkingBalance = (checkingBalance + amount);
	return checkingBalance;
}

//calculating saving after depositsaving
public double calcSavingDeposit(double amount)
{
```

```
		savingBalance = (savingBalance + amount);
		return savingBalance;
	}

	//method for transfering money from account
	public void calcCheckTransfer(double amount)
	{
		checkingBalance = checkingBalance - amount;
		savingBalance = savingBalance + amount;
	}

	//transfering money from saving account
	public void calcSavingTransfer(double amount)
	{
		savingBalance = savingBalance - amount;
		checkingBalance = checkingBalance + amount;
	}

	//displaying and checking the input format and balance(withdraw)
	public void getCheckingWithdrawInput()
	{
		boolean end = false;
		while (!end)
		{
			try
			{
				System.out.println("\nCurrent Checkings Account Balance: " +
moneyFormat.format(checkingBalance));
				System.out.print("\nAmount you want to withdraw from
Checkings Account: ");
				double amount = input.nextDouble();
				if ((checkingBalance - amount) >= 0 && amount >= 0)
				{
					calcCheckingWithdraw(amount);
					System.out.println("\nCurrent Checkings Account
Balance: " + moneyFormat.format(checkingBalance));
					end = true;
				} else
				{
					System.out.println("\nBalance Cannot be Negative.");
				}
			}
			catch (InputMismatchException e)
			{
				System.out.println("\nInvalid Choice.");
				input.next();
			}
```

```
		}
	}

	//displaying and checking the input format and balance(saving)
	public void getsavingWithdrawInput()
	{
		boolean end = false;
		while (!end)
		{
			try
			{
				System.out.println("\nCurrent Savings Account Balance: " +
moneyFormat.format(savingBalance));
				System.out.print("\nAmount you want to withdraw from Savings
Account: ");
				double amount = input.nextDouble();
				if ((savingBalance - amount) >= 0 && amount >= 0)
				{
					calcSavingWithdraw(amount);
					System.out.println("\nCurrent Savings Account Balance:
" + moneyFormat.format(savingBalance));
					end = true;
				}
				else
				{
					System.out.println("\nBalance Cannot Be Negative.");
				}
			}
			catch (InputMismatchException e)
			{
				System.out.println("\nInvalid Choice.");
				input.next();
			}
		}
	}

	//displaying and checking the input format and balance(deposit)
	public void getCheckingDepositInput()
	{
		boolean end = false;
		while (!end)
		{
			try
			{
				System.out.println("\nCurrent Checkings Account Balance: " +
moneyFormat.format(checkingBalance));
				System.out.print("\nAmount you want to deposit from
```

```
Checkings Account: ");
                                double amount = input.nextDouble();
                                if ((checkingBalance + amount) >= 0 && amount >= 0)
                                {
                                        calcCheckingDeposit(amount);
                                        System.out.println("\nCurrent Checkings Account
Balance: " + moneyFormat.format(checkingBalance));
                                        end = true;
                                }
                                else
                                {
                                        System.out.println("\nBalance Cannot Be Negative.");
                                }
                        }
                        catch (InputMismatchException e)
                        {
                                System.out.println("\nInvalid Choice.");
                                input.next();
                        }
                }
        }

        public void getSavingDepositInput()
        {
                boolean end = false;
                while (!end)
                {
                        try
                        {
                                System.out.println("\nCurrent Savings Account Balance: " +
moneyFormat.format(savingBalance));
                                System.out.print("\nAmount you want to deposit into your
Savings Account: ");
                                double amount = input.nextDouble();

                                if ((savingBalance + amount) >= 0 && amount >= 0)
                                {
                                        calcSavingDeposit(amount);
                                        System.out.println("\nCurrent Savings Account Balance:
" + moneyFormat.format(savingBalance));
                                        end = true;
                                }
                                else
                                {
                                        System.out.println("\nBalance Cannot Be Negative.");
                                }
                        }
```

```
                        catch (InputMismatchException e)
                        {
                                System.out.println("\nInvalid Choice.");
                                input.next();
                        }
                }
        }

        public void getTransferInput(String accType)
        {
                boolean end = false;
                while (!end)
                {
                        try
                        {
                                if (accType.equals("Checkings"))
                                {
                                        System.out.println("\nSelect an account you wish to
tranfers funds to:");
                                        System.out.println("1. Savings");
                                        System.out.println("2. Exit");
                                        System.out.print("\nChoice: ");
                                        int choice = input.nextInt();
                                        switch (choice)
                                        {
                                        case 1:
                                                System.out.println("\nCurrent Checkings Account
Balance: " + moneyFormat.format(checkingBalance));
                                                System.out.print("\nAmount you want to deposit
into your Savings Account: ");
                                                double amount = input.nextDouble();
                                                if ((savingBalance + amount) >= 0 &&
(checkingBalance - amount) >= 0 && amount >= 0)
                                                {
                                                        calcCheckTransfer(amount);
                                                        System.out.println("\nCurrent Savings
Account Balance: " + moneyFormat.format(savingBalance));
                                                        System.out.println(
                                                                        "\nCurrent Checkings
Account Balance: " + moneyFormat.format(checkingBalance));
                                                        end = true;
                                                }
                                                else
                                                {
                                                        System.out.println("\nBalance Cannot Be
Negative.");
                                                }
```

```
                                        break;
                                case 2:
                                        return;
                                default:
                                        System.out.println("\nInvalid Choice.");
                                        break;
                                }
                        }
                        else if (accType.equals("Savings"))
                        {
                                System.out.println("\nSelect an account you wish to
tranfers funds to: ");
                                System.out.println("1. Checkings");
                                System.out.println("2. Exit");
                                System.out.print("\nChoice: ");
                                int choice = input.nextInt();
                                switch (choice)
                                {
                                case 1:
                                        System.out.println("\nCurrent Savings Account
Balance: " + moneyFormat.format(savingBalance));
                                        System.out.print("\nAmount you want to deposit
into your savings account: ");
                                        double amount = input.nextDouble();
                                        if ((checkingBalance + amount) >= 0 &&
(savingBalance - amount) >= 0 && amount >= 0)
                                        {
                                                calcSavingTransfer(amount);
                                                System.out.println("\nCurrent checkings
account balance: " + moneyFormat.format(checkingBalance));
                                                System.out.println("\nCurrent savings
account balance: " + moneyFormat.format(savingBalance));
                                                end = true;
                                        }
                                        else
                                        {
                                                System.out.println("\nBalance Cannot Be
Negative.");
                                        }
                                        break;
                                case 2:
                                        return;
                                default:
                                        System.out.println("\nInvalid Choice.");
                                        break;
                                }
                        }
```

```
                        }
                        catch (InputMismatchException e)
                        {
                                System.out.println("\nInvalid Choice.");
                                input.next();
                        }
                }
        }
}
```

OptionMenu.java file:

```
import java.io.*;
import java.text.*;
import java.util.*;

public class OptionMenu
{
        Scanner menuInput = new Scanner(System.in);
        DecimalFormat moneyFormat = new DecimalFormat("'$'###,##0.00");
        //HashMap used for storing each coustomer number and its account detail it works in
place of database
        //works as an in memory database
        HashMap<Integer, Account> data = new HashMap<Integer, Account>();

        public void getLogin() throws IOException
        {
                boolean end = false;
                int customerNumber = 0;
                int pinNumber = 0;
                while (!end)
                {
                        try {
                                System.out.print("\nEnter your customer number: ");
                                customerNumber = menuInput.nextInt();
                                System.out.print("\nEnter your PIN number: ");
                                pinNumber = menuInput.nextInt();
                                //it traverse to the map for finding the entered coustomer
number and pin number
                                Iterator it = data.entrySet().iterator();
                                //this is the iterator function check till the map contains some
value
                                while (it.hasNext())
                                {
```

```
                                        //this code is used to check if the entry exists in the hash
map or not
                                        Map.Entry pair = (Map.Entry) it.next();
                                        Account acc = (Account) pair.getValue();
                                        if (data.containsKey(customerNumber) && pinNumber
== acc.getPinNumber())
                                        {
                                                getAccountType(acc);
                                                end = true;
                                                break;
                                        }
                                }
                                if (!end)
                                {
                                        System.out.println("\nWrong Customer Number or Pin
Number");
                                }
                        }
                        catch (InputMismatchException e)
                        {
                                System.out.println("\nInvalid Character(s). Only Numbers.");
                        }
                }
        }

        public void getAccountType(Account acc)
        {
                boolean end = false;
                while (!end)
                {
                        try
                        {
                                System.out.println("\nSelect the account you want to access: ");
                                System.out.println(" Type 1 - Checkings Account");
                                System.out.println(" Type 2 - Savings Account");
                                System.out.println(" Type 3 - Exit");
                                System.out.print("\nChoice: ");

                                int selection = menuInput.nextInt();

                                switch (selection)
                                {
                                case 1:
                                        getChecking(acc);
                                        break;
                                case 2:
                                        getSaving(acc);
```

```
                                        break;
                                case 3:
                                        end = true;
                                        break;
                                default:
                                        System.out.println("\nInvalid Choice.");
                                }
                        } catch (InputMismatchException e)
                        {
                                System.out.println("\nInvalid Choice.");
                                menuInput.next();
                        }
                }
        }

        public void getChecking(Account acc)
        {
                boolean end = false;
                while (!end)
                {
                        try
                        {
                                System.out.println("\nCheckings Account: ");
                                System.out.println(" Type 1 - View Balance");
                                System.out.println(" Type 2 - Withdraw Funds");
                                System.out.println(" Type 3 - Deposit Funds");
                                System.out.println(" Type 4 - Transfer Funds");
                                System.out.println(" Type 5 - Exit");
                                System.out.print("\nChoice: ");

                                int selection = menuInput.nextInt();

                                switch (selection)
                                {
                                case 1:
                                        System.out.println("\nCheckings Account Balance: " +
moneyFormat.format(acc.getCheckingBalance()));
                                        break;
                                case 2:
                                        acc.getCheckingWithdrawInput();
                                        break;
                                case 3:
                                        acc.getCheckingDepositInput();
                                        break;

                                case 4:
                                        acc.getTransferInput("Checkings");
```

```
                                        break;
                                case 5:
                                        end = true;
                                        break;
                                default:
                                        System.out.println("\nInvalid Choice.");
                                }
                        }
                        catch (InputMismatchException e)
                        {
                                System.out.println("\nInvalid Choice.");
                                menuInput.next();
                        }
                }
        }

        public void getSaving(Account acc)
        {
                boolean end = false;
                while (!end)
                {
                        try
                        {
                                System.out.println("\nSavings Account: ");
                                System.out.println(" Type 1 - View Balance");
                                System.out.println(" Type 2 - Withdraw Funds");
                                System.out.println(" Type 3 - Deposit Funds");
                                System.out.println(" Type 4 - Transfer Funds");
                                System.out.println(" Type 5 - Exit");
                                System.out.print("Choice: ");
                                int selection = menuInput.nextInt();
                                switch (selection)
                                {
                                case 1:
                                        System.out.println("\nSavings Account Balance: " +
moneyFormat.format(acc.getSavingBalance()));
                                        break;
                                case 2:
                                        acc.getsavingWithdrawInput();
                                        break;
                                case 3:
                                        acc.getSavingDepositInput();
                                        break;
                                case 4:
                                        acc.getTransferInput("Savings");
                                        break;
                                case 5:
```

```
                                        end = true;
                                        break;
                                default:
                                        System.out.println("\nInvalid Choice.");
                                }
                        } catch (InputMismatchException e) {
                                System.out.println("\nInvalid Choice.");
                                menuInput.next();
                        }
                }
        }

        public void createAccount() throws IOException
        {
                int cst_no = 0;
                boolean end = false;
                while (!end)
                {
                        try
                        {
                                System.out.println("\nEnter your customer number ");
                                cst_no = menuInput.nextInt();
                                Iterator it = data.entrySet().iterator();
                                while (it.hasNext()) {
                                        Map.Entry pair = (Map.Entry) it.next();
                                        if (!data.containsKey(cst_no))
                                        {
                                                end = true;
                                        }
                                }
                                if (!end)
                                {
                                        System.out.println("\nThis customer number is already
registered");
                                }
                        }
                        catch (InputMismatchException e)
                        {
                                System.out.println("\nInvalid Choice.");
                                menuInput.next();
                        }
                }
                System.out.println("\nEnter PIN to be registered");
                int pin = menuInput.nextInt();
                data.put(cst_no, new Account(cst_no, pin));
                System.out.println("\nYour new account has been successfuly registered!");
                System.out.println("\nRedirecting to login.....");
```

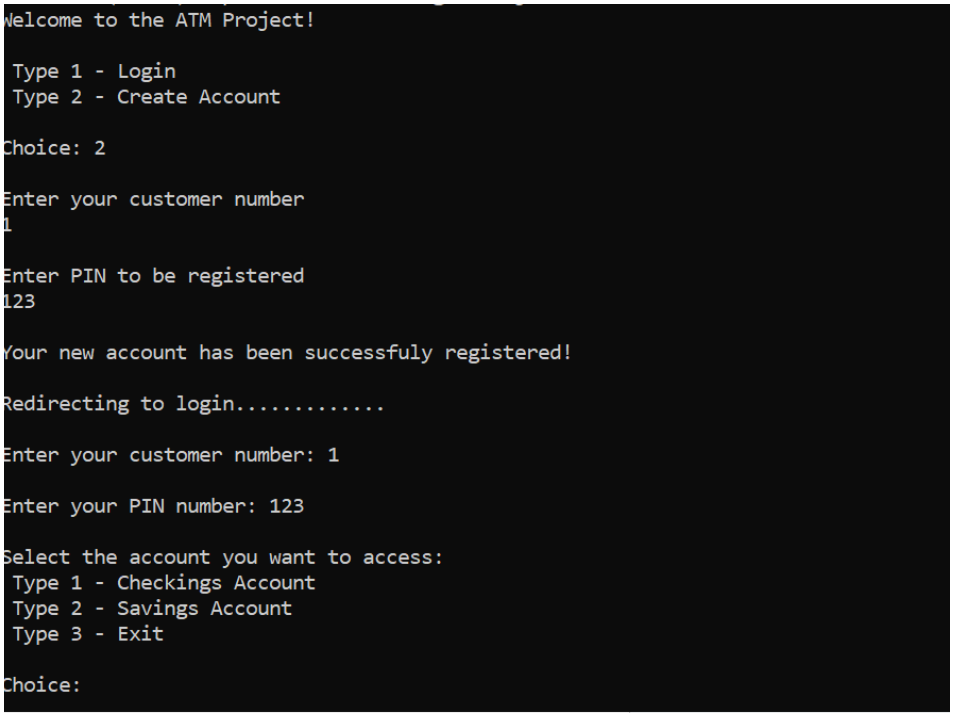
```
            getLogin();
        }

        public void mainMenu() throws IOException
        {
            data.put(952141, new Account(952141, 191904, 1000, 5000));
            data.put(123, new Account(123, 123, 20000, 50000));
            boolean end = false;
            while (!end)
            {
                try
                {
                    System.out.println("\n Type 1 - Login");
                    System.out.println(" Type 2 - Create Account");
                    System.out.print("\nChoice: ");
                    int choice = menuInput.nextInt();
                    switch (choice)
                    {
                    case 1:
                        getLogin();
                        end = true;
                        break;
                    case 2:
                        createAccount();
                        end = true;
                        break;
                    default:
                        System.out.println("\nInvalid Choice.");
                    }
                }
                catch (InputMismatchException e)
                {
                    System.out.println("\nInvalid Choice.");
                    menuInput.next();
                }
            }
            System.out.println("\nThank You for using this ATM.\n");
            menuInput.close();
            System.exit(0);
        }
}
```

ATM.java file:

```
import java.io.*;

public class ATM
{

	public static void main(String[] args) throws IOException
	{
		OptionMenu optionMenu = new OptionMenu();
		System.out.println("Welcome to the ATM Project!");
		optionMenu.mainMenu();
	}


}
```

OUTPUT:

TEST CASE 1: LOGIN AND CREATING ACCOUNT



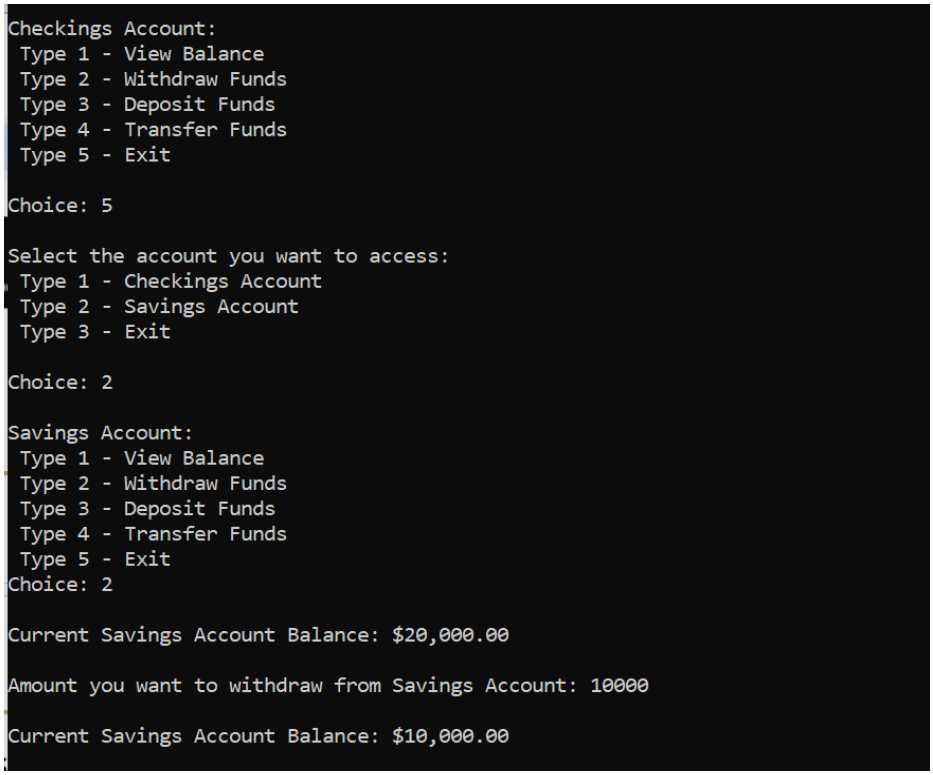
TEST CASE 2: DEPOSIT INTO CHECKING ACCOUNT

```
Select the account you want to access:
 Type 1 - Checkings Account
 Type 2 - Savings Account
 Type 3 - Exit

Choice: 1

Checkings Account:
 Type 1 - View Balance
 Type 2 - Withdraw Funds
 Type 3 - Deposit Funds
 Type 4 - Transfer Funds
 Type 5 - Exit

Choice: 3

Current Checkings Account Balance: $0.00

Amount you want to deposit from Checkings Account: 12000000

Current Checkings Account Balance: $12,000,000.00
```

TEST CASE 3: TRANSFER FROM CHECKINGS TO SAVINGS

```
Checkings Account:
 Type 1 - View Balance
 Type 2 - Withdraw Funds
 Type 3 - Deposit Funds
 Type 4 - Transfer Funds
 Type 5 - Exit

Choice: 4

Select an account you wish to tranfers funds to:
1. Savings
2. Exit

Choice: 1

Current Checkings Account Balance: $12,000,000.00

Amount you want to deposit into your Savings Account: 20000

Current Savings Account Balance: $20,000.00

Current Checkings Account Balance: $11,980,000.00
```

TEST CASE 4: WITHDRAW FROM SAVINGS



Conclusion:

In conclusion, the Java-based Console ATM Management System offers a seamless and secure platform for managing both savings and checking accounts through a simple yet effective console interface. Through meticulous Java programming, the system ensures the integrity of diverse transactions, including withdrawals, deposits, transfers, and balance inquiries. Its user-friendly design, coupled with robust security measures, positions it as a reliable solution for financial management, demonstrating the power of console-based programming in delivering efficient and accessible banking services.