ATM Management System Using Java and OOPS

Introduction

The ATM Machine Simulation project is designed to mimic the fundamental operations of a real-world Automated Teller Machine (ATM). This project provides a simple yet functional console-based application that allows users to perform essential banking operations such as account creation, balance inquiry, cash withdrawal, cash deposit, and mini statement viewing. The project is built using Java, leveraging object-oriented principles to ensure modularity and ease of maintenance. It aims to provide a clear understanding of how ATM systems operate, including user authentication and transaction handling. This project simulates an ATM machine application that allows users to perform various banking operations such as checking their balance, withdrawing money, depositing money, and viewing mini statements. The application supports user authentication via an ATM number and PIN.

Key Components

1.Main Class (App)

- Entry point of the application.
- Manages user authentication and menu navigation.
- Provides options to either create a new account or log in with an existing account.

2.ATM Class (ATM)

- Stores ATM numbers, PINs, and account balances.
- Provides methods to set and get these values.
- ATM Operations Interface (ATMOpInterface)

Defines methods for ATM operations like viewing balance, withdrawing, depositing, and viewing mini statements.

3.ATM Operations Implementation (ATMOpImple)

- Implements the ATMOpInterface methods.
- Contains logic for performing the ATM operations.

Step-by-Step Breakdown

Step 1: Main Method (App.main)

Initialize Components:

- Create an instance of ATMOpInterface and Scanner.
- Display a welcome message.
- Prompt the user to choose between logging in or creating a new account.

User Choice:

If the user chooses to log in (presses 1):

- Prompt for ATM number and PIN.
- Validate credentials.

If the user chooses to create a new account (presses 2):

- Prompt for new ATM number and PIN.
- Store the new credentials.
- Validate the new credentials.

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

public class App {
    public static void main(String[] args) throws Exception {
       ATMOpInterface op=new ATMOpImple();
       Scanner scan=new Scanner(System.in);
       System.out.println("Welcome to ATM Machine!.....");
       System.out.println("Already have an account press 1 to continue
OR to create new Account press 2");
       int i=scan.nextInt();
       ATM atm=new ATM();
       ArrayList<Integer> atmNum=atm.getATMnum();
       ArrayList<Integer> atmPin=atm.getATMpin();
    if(i==1){
       System.out.println("Enter ATM Number:");
       int atmInput=scan.nextInt();
       System.out.println("Enter ATM Pin Number");
       int pinInput=scan.nextInt();
       ValidationSuccess(atmInput, pinInput, atmNum, atmPin);
    }
    else if(i==2){
        System.out.println("Enter new ATM Number to create New
Account");
        int num=scan.nextInt();
        System.out.println("Enter new ATM pin to create new Account");
        int pin=scan.nextInt();
        atm.setATMnum(num);
        atm.setATMpin(pin);
        System.out.println("Account Created SuccessFully!...");
        ValidationSuccess(num, pin, atmNum, atmPin);
      }
    }
    public static void ValidationSuccess(int atmInput,int
pinInput,ArrayList<Integer> atmNum,ArrayList<Integer> atmPin){
        ATMOpInterface op=new ATMOpImple();
        if(atmNum.contains(atmInput) && atmPin.contains(pinInput)){
            Scanner scan=new Scanner(System.in);
            while(true) {
                System.out.println("1.View Available balance");
                System.out.println("2.With Draw the amount");
                System.out.println("3.Deposit Amount");
                System.out.println("4.View Mini Statement");
                System.out.println("5.Exit the Application");
                System.out.println("Enter Choice: ");
                int ch=scan.nextInt();
                if(ch==1){
                    op.viewBalance();
```

```
                }
                else if(ch==2){
                    System.out.println("Enter Amount to withdraw ");
                    double wd=scan.nextDouble();
                    op.withDrawAmount(wd);
                }
                else if(ch==3){
                     System.out.println("Enter Amount to Deposit");
                     double depAmo=scan.nextDouble(); //5000
                     op.depositAmount(depAmo);
                }
                else if(ch==4){
                    System.out.println("-----Your Mini Statement-----
");
                     op.viewMiniStatement();
                }
                else if(ch==5){
                  System.out.println("Collect your ATM Card");
                  System.out.println("Thank you for using ATM
Machine");
                  System.exit(0);
                }
                else{
                   System.out.println("Please enter valid choice");
                }
            }
          }
          else{
              System.out.println("Incorrect ATM Number and Pin...Enter
Correct Number OR Create New One");
          }
    }
}
```

Step 2: ATM Class (ATM)

Attributes:

- balance: Stores the account balance.
- depositAmount: Stores the deposit amount.
- withDrawAmount: Stores the withdrawal amount.
- atmNum: Stores the list of ATM numbers.
- atmPin: Stores the list of ATM PINs.

Methods:

- Getters and setters for atmNum, atmPin, balance, depositAmount, and withDrawAmount.

```
import java.util.*;
public class ATM {
     private double balance;
     private double depositAmount;
     private double withDrawAmount;
     private ArrayList<Integer> atmNum=new ArrayList<>();
     private ArrayList<Integer> atmPin=new ArrayList<>();
     //default construtor
     public ATM(){

     }
     //getter and setter Methods
     public void setATMnum(int atmN){
          atmNum.add(atmN);
     }
     public void setATMpin(int atmP){
        atmPin.add(atmP);
     }
     public ArrayList<Integer> getATMnum(){
        return atmNum;
     }
     public ArrayList<Integer> getATMpin(){
        return atmPin;
     }
     public double getBalance(){
        return balance;
     }
     public void setBalance(double balance){
         this.balance=balance;
     }
     public double getDepositAmount(){
        return depositAmount;
     }
     public void setDepositAmount(double depositAmount){
         this.depositAmount=depositAmount;
     }
     public double getWithDrawAmount(){
        return withDrawAmount;
     }
     public void setWithDrawAmount(double withDrawAmount){
         this.withDrawAmount=withDrawAmount;
     }
}
```

Step 3: ATM Operations Implementation (ATMOpImple)

Attributes:

- atm: An instance of the ATM class.
- 'l' An ArrayList to store mini statements.

Methods:

- viewBalance(): Displays the current balance.
- withDrawAmount(double withDrawAmount): Withdraws money if the amount is a multiple of 500 and sufficient balance is available.
- depositAmount(double depositAmount): Deposits money into the account.
- viewMiniStatement(): Displays the mini statement of transactions.

```
import java.util.*;
public class ATMOpImple implements ATMOpInterface {
     ATM atm=new ATM();
     ArrayList<String> l=new ArrayList<>();
      @Override
      public void viewBalance(){
            System.out.println("Available balance is
:"+atm.getBalance());
       }
      @Override
      public void withDrawAmount(double withDrawAmount){
        if(withDrawAmount%500==0){
         if(atm.getBalance()-withDrawAmount<0){
            l.add("Insufficient balance error for Rs."+withDrawAmount);
            System.out.println("Insufficient Balance...You have only
Rs."+" "+atm.getBalance());
          }
          else if(atm.getBalance()-withDrawAmount>=0){
            l.add("Rs."+withDrawAmount+" "+"Amount withdrawn");
           System.out.println("Rs."+ withDrawAmount +" "+" withdrawed
sucessfully ... Collect the cash!");
            atm.setBalance(atm.getBalance()-withDrawAmount);
            viewBalance();
          }
         }
```

```
          else{
              l.add("Withdrawn amount error in terms of 500 only for
Rs."+ withDrawAmount);
              System.out.println("You can withdraw amount in terms of 500
only!....");
          }
      }
      @Override
      public void depositAmount(double depositAmount){
          l.add("Rs."+depositAmount+" "+ "Amount Deposited");
          System.out.println("Rs." +depositAmount + "   " +"Deposited
SucessFully!");
          atm.setBalance(atm.getBalance()+depositAmount);
          viewBalance();
      }
      @Override
      public void viewMiniStatement(){
         for(int i=0;i<l.size();i++){
             System.out.println(l.get(i));
         }
         System.out.println("Now Available balance is"+" "+
atm.getBalance());
      }
}
```

Step 4: ATM Operations Interface (ATMOpInterface)

Methods:

- viewBalance(): To view the current balance.
- withDrawAmount(double withDrawAmount): To withdraw a specified amount.
- depositAmount(double depositAmount): To deposit a specified amount.
- viewMiniStatement(): To view the mini statement of transactions

```
public interface ATMOpInterface {
      public void viewBalance();
      public void withDrawAmount(double withDrawAmount);
      public void depositAmount(double depositAmount);
      public void viewMiniStatement();
}
```

Demonstration of the project

Welcome to ATM Machine!.....
Already have an account press 1 to continue OR to create new Account press 2
2
Enter new ATM Number to create New Account
1234
Enter new ATM pin to create new Account
1001
Account Created SuccessFully!...
1.View Available balance
2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application
Enter Choice:
1
Available balance is :0.0
1.View Available balance
2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application
Enter Choice:
3
Enter Amount to Deposit
10000
Rs.10000.0 Deposited SucessFully!
Available balance is :10000.0
1.View Available balance
2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application
Enter Choice:
2
Enter Amount to withdraw
5500
Rs.5500.0 withdrawed sucessfully ... Collect the cash!
Available balance is :4500.0
1.View Available balance

2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application
Enter Choice:
2
Enter Amount to withdraw
5000
Insufficient Balance...You have only Rs. 4500.0
1.View Available balance
2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application
Enter Choice:
300
Please enter valid choice
1.View Available balance
2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application
Enter Choice:
2
Enter Amount to withdraw
300
You can withdraw amount in terms of 500 only!....
1.View Available balance
2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application
Enter Choice:
2
Enter Amount to withdraw
4000
Rs.4000.0 withdrawed sucessfully ... Collect the cash!
Available balance is :500.0
1.View Available balance
2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application

```
Enter Choice:
2
Enter Amount to withdraw
500
Rs.500.0  withdrawed sucessfully ... Collect the cash!
Available balance is :0.0
1.View Available balance
2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application
Enter Choice:
4
-----Your Mini Statement-----
Rs.10000.0 Amount Deposited
Rs.5500.0 Amount withdrawn
Insufficient balance error for Rs.5000.0
Withdrawn amount error in terms of 500 only for Rs.300.0
Rs.4000.0 Amount withdrawn
Rs.500.0 Amount withdrawn
Now Available balance is 0.0
1.View Available balance
2.With Draw the amount
3.Deposit Amount
4.View Mini Statement
5.Exit the Application
Enter Choice:
5
Collect your ATM Card
Thank you for using ATM Machine
```

Summary

Main Class (App):

Handles user interaction for login or account creation.Manages the validation of user credentials and navigation through the ATM menu.

ATM Class (ATM):

Stores ATM numbers, PINs, and account balances. Provides methods to set and get these values.

ATM Operations Interface (ATMOpInterface):

Defines the methods that need to be implemented for ATM operations.

ATM Operations Implementation (ATMOpImple):

Implements the interface methods. Handles the logic for viewing balance, withdrawing, depositing, and viewing mini statements. This comprehensive structure ensures a clear separation of concerns, making the application easy to understand, maintain, and extend.

Conclusion

The ATM Machine Simulation project successfully demonstrates the core functionalities of an ATM system through a console-based Java application. By implementing essential features such as account creation, balance inquiry, cash withdrawal, cash deposit, and mini statement generation, the project provides a practical insight into the workings of ATM software. The use of interfaces and classes to segregate different operations ensures a clean and maintainable code structure. This project not only serves as an educational tool for understanding basic banking operations but also lays the groundwork for more complex ATM functionalities and enhancements in the future.