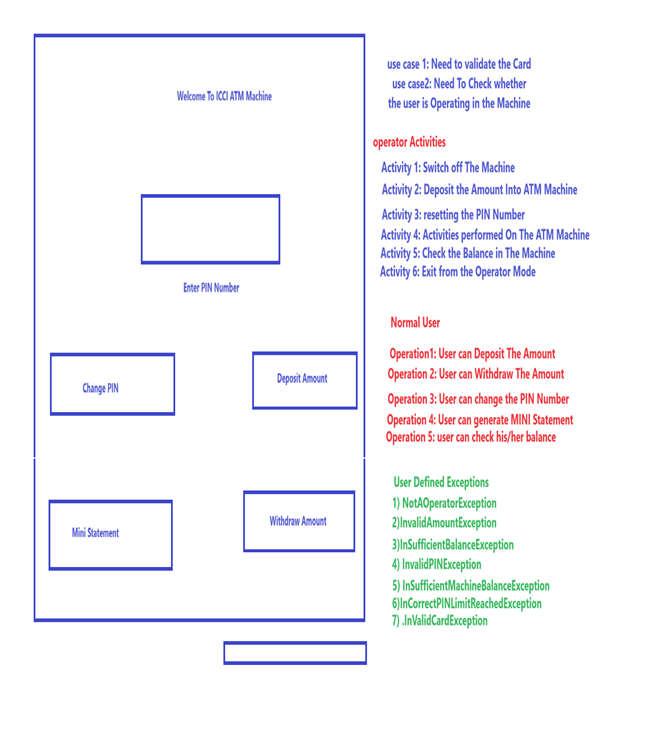
****

**IATMService.java**

**----------------------**

**package** interfaces;

**import** customExceptions.InsufficientBalanceException;

**import** customExceptions.InvalidAmountException;

**import** customExceptions.NotAOperatorException;

**import** customExceptions.InsufficientMachineBalanceException;

**public** **interface** IATMService {

// to get the user type.weather the user is operator or normaluser.

**public** **abstract** String getUserType() **throws** NotAOperatorException;

// To WithdrawAmount

// 1)will throw InvalidAmountException if the amount is not valid denomination

// 2) will throw insufficient balance Exception if the customer has insufficient

// amount in his/her account

// 3)will throw insufficient machine balance exception if the machine has in

// sufficient cash.

**public** **abstract** **double** withdrawAmount(**double** wthAmount)

**throws** InvalidAmountException, InsufficientBalanceException, InsufficientMachineBalanceException;

// to deposit amount

**public** **abstract** **void** depositAmount(**double** dptAmount) **throws** InvalidAmountException;

// to check AccountBalance

**public** **abstract** **double** checkAccountBalance();

// to change PIN Number

**public** **abstract** **void** changePinNumber(**int** pinNumber);

// to get the PIN number

**public** **abstract** **int** getPinNumber();

// to get the userName

**public** **abstract** String getUserName();

// to decrease the number of chances whiel enter the wrong pin number

**public** **abstract** **void** decreaseChances();

// to get the Chances of PIN Number

**public** **abstract** **int** getChances();

// to reset the pin number chances by the bank Operator.

**public** **abstract** **void** resetPinChances();

// to get the miniStatement of an account

**public** **abstract** **void** generateMiniStatement();

}

**IncorrectPinLimitReachedException.java**

**package** com.codegnan.customExceptions;

**public** **class** IncorrectPinLimitReachedException **extends** Exception{

**public** IncorrectPinLimitReachedException(String errorMsg){

**super**(errorMsg);

}

}

**InsufficientBalanceException.java**

**package** com.codegnan.customExceptions;

**public** **class** InsufficientBalanceException **extends** Exception {

**public** InsufficientBalanceException(String errorMsg) {

**super**(errorMsg);

}

}

**InsufficientMachineBalanceException.java**

**package** com.codegnan.customExceptions;

**public** **class** InsufficientMachineBalanceException **extends** Exception {

**public** InsufficientMachineBalanceException(String errorMsg) {

**super**(errorMsg);

}

}

**InvalidAmountException.java**

**package** com.codegnan.customExceptions;

**public** **class** InvalidAmountException **extends** Exception {

**public** InvalidAmountException(String errorMsg) {

**super**(errorMsg);

}

}

**InvalidCardException.java**

**package** com.codegnan.customExceptions;

**public** **class** InvalidCardException **extends** Exception{

**public** InvalidCardException(String errorMsg){

**super**(errorMsg);

}

}

**InvalidPinException.java**

**package** com.codegnan.customExceptions;

**public** **class** InvalidPinException **extends** Exception {

**public** InvalidPinException(String errorMsg) {

**super**(errorMsg);

}

}

**NotAOperatorException.java**

**package** com.codegnan.customExceptions;

**public** **class** NotAOperatorException **extends** Exception {

**public** NotAOperatorException(String errorMsg) {

**super**(errorMsg);

}

}

**AxisDebitCard.java**

**----------------------**

**package** com.codegnan.cards;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** com.codegnan.customExceptions.InsufficientBalanceException;

**import** com.codegnan.customExceptions.InsufficientMachineBalanceException;

**import** com.codegnan.customExceptions.InvalidAmountException;

**import** com.codegnan.customExceptions.NotAOperatorException;

**import** com.codegnan.interfaces.IATMService;

**public** **class** AxisDebitCard **implements** IATMService {

String name;

**long** debitCardNumber;

**double** accountBalance;

**int** pinNumber;

ArrayList<String> statement;

**final** String type = "user";

**int** chances;

**public** AxisDebitCard(**long** debitCardNumber, String name, **double** accountBalance, **int** pinNumber) {

chances = 3;

**this**.name = name;

**this**.accountBalance = accountBalance;

**this**.pinNumber = pinNumber;

statement = **new** ArrayList<>();

}

@Override

**public** String getUserType() **throws** NotAOperatorException {

// **TODO** Auto-generated method stub

**return** type;

}

@Override

**public** **double** withdrawAmount(**double** wthAmount)

**throws** InvalidAmountException, InsufficientBalanceException, InsufficientMachineBalanceException {

**if** (wthAmount <= 0) {

**throw** **new** InvalidAmountException("You can Enter Zero(0) amount to withdraw. please enter valid amount");

} **else** **if** (wthAmount % 10 != 0) {

**throw** **new** InsufficientBalanceException("Please Withdraw Multiples of 100");

} **else** **if** (wthAmount < 500) {

**throw** **new** InsufficientBalanceException("Please Withdraw More Than 500");

} **else** **if** (wthAmount > accountBalance) {

**throw** **new** InsufficientBalanceException(

"you don:t have sufficient balance to withdraw.. Please check your balanace");

} **else** {

accountBalance = accountBalance - wthAmount;

statement.add("Debited : " + wthAmount);

**return** wthAmount;

}

}

@Override

**public** **void** depositAmount(**double** dptAmount) **throws** InvalidAmountException {

**if** (dptAmount <= 0 || dptAmount % 10 != 0) {

**throw** **new** InvalidAmountException("Please Deposit Multiples of 10");

} **else** {

accountBalance = accountBalance + dptAmount;

statement.add("Creadited : " + dptAmount);

}

}

@Override

**public** **double** checkAccountBalance() {

// **TODO** Auto-generated method stub

**return** accountBalance;

}

@Override

**public** **void** changePinNumber(**int** pinNumber) {

**this**.pinNumber = pinNumber;

}

@Override

**public** **int** getPinNumber() {

**return** pinNumber;

}

@Override

**public** String getUserName() {

**return** name;

}

@Override

**public** **void** decreaseChances() {

--chances;

}

@Override

**public** **int** getChances() {

**return** chances;

}

@Override

**public** **void** resetPinChances() {

chances = 3;

}

@Override

**public** **void** generateMiniStatement() {

**int** count = 5;

**if** (statement.size() == 0) {

System.***out***.println("There are no Transactions Happend");

**return**;

}

System.***out***.println("================List 5 Transactions ========================");

Collections.*reverse*(statement);

**for** (String trans : statement) {

**if** (count == 0) {

**break**;

}

System.***out***.println(trans);

count--;

}

Collections.*reverse*(statement);

}

}

**HDFCDebitCard.java**

**------------------------**

**package** com.codegnan.cards;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** com.codegnan.customExceptions.InsufficientBalanceException;

**import** com.codegnan.customExceptions.InsufficientMachineBalanceException;

**import** com.codegnan.customExceptions.InvalidAmountException;

**import** com.codegnan.customExceptions.NotAOperatorException;

**import** com.codegnan.interfaces.IATMService;

**public** **class** HDFCDebitCard **implements** IATMService {

String name;

**long** debitCardNumber;

**double** accountBalance;

**int** pinNumber;

ArrayList<String> statement;

**final** String type = "user";

**int** chances;

**public** HDFCDebitCard(**long** debitCardNumber, String name, **double** accountBalance, **int** pinNumber) {

chances = 3;

**this**.name = name;

**this**.accountBalance = accountBalance;

**this**.pinNumber = pinNumber;

statement = **new** ArrayList<>();

}

@Override

**public** String getUserType() **throws** NotAOperatorException {

// **TODO** Auto-generated method stub

**return** type;

}

@Override

**public** **double** withdrawAmount(**double** wthAmount)

**throws** InvalidAmountException, InsufficientBalanceException, InsufficientMachineBalanceException {

**if** (wthAmount <= 0) {

**throw** **new** InvalidAmountException("You can Enter Zero(0) amount to withdraw. please enter valid amount");

} **else** **if** (wthAmount % 10 != 0) {

**throw** **new** InsufficientBalanceException("Please Withdraw Multiples of 100");

} **else** **if** (wthAmount < 500) {

**throw** **new** InsufficientBalanceException("Please Withdraw More Than 500");

} **else** **if** (wthAmount > accountBalance) {

**throw** **new** InsufficientBalanceException(

"you don:t have sufficient balance to withdraw.. Please check your balanace");

} **else** {

accountBalance = accountBalance - wthAmount;

statement.add("Debited : " + wthAmount);

**return** wthAmount;

}

}

@Override

**public** **void** depositAmount(**double** dptAmount) **throws** InvalidAmountException {

**if** (dptAmount <= 0 || dptAmount % 10 != 0) {

**throw** **new** InvalidAmountException("Please Deposit Multiples of 10");

} **else** {

accountBalance = accountBalance + dptAmount;

statement.add("Creadited : " + dptAmount);

}

}

@Override

**public** **double** checkAccountBalance() {

// **TODO** Auto-generated method stub

**return** accountBalance;

}

@Override

**public** **void** changePinNumber(**int** pinNumber) {

**this**.pinNumber = pinNumber;

}

@Override

**public** **int** getPinNumber() {

**return** pinNumber;

}

@Override

**public** String getUserName() {

**return** name;

}

@Override

**public** **void** decreaseChances() {

--chances;

}

@Override

**public** **int** getChances() {

**return** chances;

}

@Override

**public** **void** resetPinChances() {

chances = 3;

}

@Override

**public** **void** generateMiniStatement() {

**int** count = 5;

**if** (statement.size() == 0) {

System.***out***.println("There are no Transactions Happend");

**return**;

}

System.***out***.println("================List 5 Transactions ========================");

Collections.*reverse*(statement);

**for** (String trans : statement) {

**if** (count == 0) {

**break**;

}

System.***out***.println(trans);

count--;

}

Collections.*reverse*(statement);

}

}

**SBIDebitCard.java**

**----------------------**

**package** com.codegnan.cards;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** com.codegnan.customExceptions.InsufficientBalanceException;

**import** com.codegnan.customExceptions.InsufficientMachineBalanceException;

**import** com.codegnan.customExceptions.InvalidAmountException;

**import** com.codegnan.customExceptions.NotAOperatorException;

**import** com.codegnan.interfaces.IATMService;

**public** **class** SBIDebitCard **implements** IATMService {

String name;

**long** debitCardNumber;

**double** accountBalance;

**int** pinNumber;

ArrayList<String> statement;

**final** String type = "user";

**int** chances;

**public** SBIDebitCard(**long** debitCardNumber, String name, **double** accountBalance, **int** pinNumber) {

chances = 3;

**this**.name = name;

**this**.accountBalance = accountBalance;

**this**.pinNumber = pinNumber;

statement = **new** ArrayList<>();

}

@Override

**public** String getUserType() **throws** NotAOperatorException {

// **TODO** Auto-generated method stub

**return** type;

}

@Override

**public** **double** withdrawAmount(**double** wthAmount)

**throws** InvalidAmountException, InsufficientBalanceException, InsufficientMachineBalanceException {

**if** (wthAmount <= 0) {

**throw** **new** InvalidAmountException("You can Enter Zero(0) amount to withdraw. please enter valid amount");

} **else** **if** (wthAmount % 10 != 0) {

**throw** **new** InsufficientBalanceException("Please Withdraw Multiples of 100");

} **else** **if** (wthAmount < 500) {

**throw** **new** InsufficientBalanceException("Please Withdraw More Than 500");

} **else** **if** (wthAmount > accountBalance) {

**throw** **new** InsufficientBalanceException(

"you don:t have sufficient balance to withdraw.. Please check your balanace");

} **else** {

accountBalance = accountBalance - wthAmount;

statement.add("Debited : " + wthAmount);

**return** wthAmount;

}

}

@Override

**public** **void** depositAmount(**double** dptAmount) **throws** InvalidAmountException {

**if** (dptAmount <= 0 || dptAmount % 10 != 0) {

**throw** **new** InvalidAmountException("Please Deposit Multiples of 10");

} **else** {

accountBalance = accountBalance + dptAmount;

statement.add("Creadited : " + dptAmount);

}

}

@Override

**public** **double** checkAccountBalance() {

// **TODO** Auto-generated method stub

**return** accountBalance;

}

@Override

**public** **void** changePinNumber(**int** pinNumber) {

**this**.pinNumber = pinNumber;

}

@Override

**public** **int** getPinNumber() {

**return** pinNumber;

}

@Override

**public** String getUserName() {

**return** name;

}

@Override

**public** **void** decreaseChances() {

--chances;

}

@Override

**public** **int** getChances() {

**return** chances;

}

@Override

**public** **void** resetPinChances() {

chances = 3;

}

@Override

**public** **void** generateMiniStatement() {

**int** count = 5;

**if** (statement.size() == 0) {

System.***out***.println("There are no Transactions Happend");

**return**;

}

System.***out***.println("================List 5 Transactions ========================");

Collections.*reverse*(statement);

**for** (String trans : statement) {

**if** (count == 0) {

**break**;

}

System.***out***.println(trans);

count--;

}

Collections.*reverse*(statement);

}

}

**OperatorCard.java**

**------------------**

**package** com.codegnan.cards;

**import** com.codegnan.customExceptions.InsufficientBalanceException;

**import** com.codegnan.customExceptions.InsufficientMachineBalanceException;

**import** com.codegnan.customExceptions.InvalidAmountException;

**import** com.codegnan.customExceptions.NotAOperatorException;

**import** com.codegnan.interfaces.IATMService;

**public** **class** OperatorCard **implements** IATMService {

**private** **int** pinNumber;

**private** **long** id;

**private** String name;

**private** **final** String type = "operator";

**public** OperatorCard(**long** idn, **int** pin, String name) {

id = idn;

pinNumber = pin;

**this**.name = name;

}

@Override

**public** String getUserType() **throws** NotAOperatorException {

// **TODO** Auto-generated method stub

**return** type;

}

@Override

**public** **double** withdrawAmount(**double** wthAmount)

**throws** InvalidAmountException, InsufficientBalanceException, InsufficientMachineBalanceException {

**return** 0;

}

@Override

**public** **void** depositAmount(**double** dptAmount) **throws** InvalidAmountException {

}

@Override

**public** **double** checkAccountBalance() {

**return** 0;

}

@Override

**public** **void** changePinNumber(**int** pinNumber) {

}

@Override

**public** **int** getPinNumber() {

**return** pinNumber;

}

@Override

**public** String getUserName() {

**return** name;

}

@Override

**public** **void** decreaseChances() {

// **TODO** Auto-generated method stub

}

@Override

**public** **int** getChances() {

// **TODO** Auto-generated method stub

**return** 0;

}

@Override

**public** **void** resetPinChances() {

}

@Override

**public** **void** generateMiniStatement() {

// **TODO** Auto-generated method stub

}

}

**ATMOperations.java**

**-----------------------**

**package** com.codegnan.operations;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Scanner;

**import** com.codegnan.cards.AxisDebitCard;

**import** com.codegnan.cards.HDFCDebitCard;

**import** com.codegnan.cards.OperatorCard;

**import** com.codegnan.cards.SBIDebitCard;

**import** com.codegnan.customExceptions.IncorrectPinLimitReachedException;

**import** com.codegnan.customExceptions.InsufficientBalanceException;

**import** com.codegnan.customExceptions.InsufficientMachineBalanceException;

**import** com.codegnan.customExceptions.InvalidAmountException;

**import** com.codegnan.customExceptions.InvalidCardException;

**import** com.codegnan.customExceptions.InvalidPinException;

**import** com.codegnan.customExceptions.NotAOperatorException;

**import** com.codegnan.interfaces.IATMService;

**public** **class** ATMOperations {

// Initial ATM machine balance

**public** **static** **double** *ATM\_MACHINE\_BALANCE* = 100000.0;

// List to keep track of all activities performed on the ATM

**public** **static** ArrayList<String> *ACTIVITY* = **new** ArrayList<>();

// Database to map card numbers to card objects

**public** **static** HashMap<Long, IATMService> *dataBase* = **new** HashMap<>();

// Flag to indicate if the ATM machine is on or off

**public** **static** **boolean** *MACHINE\_ON* = **true**;

// Reference to the current card in use

**public** **static** IATMService *card*;

// Validate the inserted card by checking against the database

**public** **static** IATMService validateCard(**long** cardNumber) **throws** InvalidCardException {

**if** (*dataBase*.containsKey(cardNumber)) {

**return** *dataBase*.get(cardNumber);

} **else** {

*ACTIVITY*.add("Accessed by: " + cardNumber + " is Not Compatible");

**throw** **new** InvalidCardException("This is Not A valid Card");

}

}

// Display the activities performed on the ATM

**public** **static** **void** checkATMMachineActivities() {

System.***out***.println("=================== Activities Performed ===================");

**for** (String activity : *ACTIVITY*) {

System.***out***.println("==========================================================");

System.***out***.println(activity);

System.***out***.println("==========================================================");

}

}

// Reset the number of PIN attempts for a user

**public** **static** **void** resetUserAttempts(IATMService operatorCard) {

IATMService card = **null**;

**long** number;

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println("Enter your CARD Number:");

number = scanner.nextLong();

**try** {

card = *validateCard*(number);

card.resetPinChances(); // Resetting PIN attempts for the specified card

*ACTIVITY*.add("Accessed By: " + operatorCard.getUserName() + " to reset number of chances for user.");

} **catch** (InvalidCardException ive) {

System.***out***.println(ive.getMessage());

}

}

// Validate user credentials including PIN verification

**public** **static** IATMService validateCredentials(**long** cardNumber, **int** pinNumber)

**throws** InvalidCardException, IncorrectPinLimitReachedException, InvalidPinException {

**if** (*dataBase*.containsKey(cardNumber)) {

*card* = *dataBase*.get(cardNumber);

} **else** {

**throw** **new** InvalidCardException("This card is Not A valid Card");

}

**try** {

**if** (*card*.getUserType().equals("operator")) {

// Operators have a different PIN validation process

**if** (*card*.getPinNumber() != pinNumber) {

**throw** **new** InvalidPinException("Dear operator, Please enter Correct PIN Number");

} **else** {

**return** *card*;

}

}

} **catch** (NotAOperatorException noe) {

noe.printStackTrace();

}

// Validate PIN and handle incorrect attempts

**if** (*card*.getChances() <= 0) {

**throw** **new** IncorrectPinLimitReachedException(

"You have Reached Wrong Limit of PIN Number, which is 3 attempts");

}

**if** (*card*.getPinNumber() != pinNumber) {

*card*.decreaseChances(); // Decrease the number of remaining chances

**throw** **new** InvalidPinException("You Have Entered A Wrong PIN Number");

} **else** {

**return** *card*;

}

}

// Validate the amount for withdrawal to ensure sufficient machine balance

**public** **static** **void** validateAmount(**double** amount) **throws** InsufficientMachineBalanceException {

**if** (amount > *ATM\_MACHINE\_BALANCE*) {

**throw** **new** InsufficientMachineBalanceException("Insufficient cash in the Machine");

}

}

// Validate deposit amount to ensure it meets machine requirements

**public** **static** **void** validateDepositAmount(**double** amount)

**throws** InsufficientMachineBalanceException, InvalidAmountException {

// Ensure deposit amount is a multiple of 100

**if** (amount % 100 != 0) {

**throw** **new** InvalidAmountException("Please deposit amounts in multiples of 100.");

}

// Check if deposit will exceed machine capacity

**if** (amount + *ATM\_MACHINE\_BALANCE* > 200000.0d) {

*ACTIVITY*.add("Unable to deposit cash in the machine...");

**throw** **new** InsufficientMachineBalanceException(

"You can't deposit cash as the limit of the machine is reached.");

}

}

// Operations available in operator mode

**public** **static** **void** operatorMode(IATMService card) {

Scanner scanner = **new** Scanner(System.***in***);

**double** amount;

**boolean** flag = **true**;

**while** (flag) {

System.***out***.println("OPERATOR MODE: Operator Name: " + card.getUserName());

System.***out***.println("===================================================");

System.***out***.println("|| 0. Switch Off The Machine ||");

System.***out***.println("|| 1. To Check The ATM Machine Balance ||");

System.***out***.println("|| 2. Deposit Cash In The Machine ||");

System.***out***.println("|| 3. Reset The User PIN Attempts ||");

System.***out***.println("|| 4. To Check Activities Performed In the Machine ||");

System.***out***.println("|| 5. Exit Operator Mode ||");

System.***out***.println("Please Enter Your Choice: ");

**int** option = scanner.nextInt();

**switch** (option) {

**case** 0:

*MACHINE\_ON* = **false**; // Turn off the machine

*ACTIVITY*.add(

"Accessed By " + card.getUserName() + " Activity Performed: Switching Off The ATM Machine");

flag = **false**;

**break**;

**case** 1:

*ACTIVITY*.add("Accessed By " + card.getUserName() + " Activity Performed: Checking ATM Machine Balance");

System.***out***.println("The Balance Of ATM Machine Is: " + *ATM\_MACHINE\_BALANCE* + " Is Available");

**break**;

**case** 2:

System.***out***.println("Enter The Amount To Deposit: ");

amount = scanner.nextDouble();

**try** {

*validateDepositAmount*(amount); // Validate deposit amount

*ATM\_MACHINE\_BALANCE* += amount; // Update ATM balance

*ACTIVITY*.add("Accessed By " + card.getUserName()

+ " Activity Performed: Depositing Cash in The ATM Machine");

System.***out***.println("==================================================================");

System.***out***.println(

"========================== Cash Added In The ATM Machine =========================");

System.***out***.println("==================================================================");

} **catch** (InvalidAmountException | InsufficientMachineBalanceException e) {

System.***out***.println(e.getMessage());

}

**break**;

**case** 3:

*resetUserAttempts*(card); // Reset user's PIN attempts

System.***out***.println("==================================================================");

System.***out***.println("========================== User Attempts Are Reset ========================");

System.***out***.println("==================================================================");

*ACTIVITY*.add("Accessed By " + card.getUserName()

+ " Activity Performed: Resetting The PIN Attempts Of User");

**break**;

**case** 4:

*checkATMMachineActivities*(); // Display ATM activities

**break**;

**case** 5:

flag = **false**; // Exit operator mode

**break**;

**default**:

System.***out***.println("You Have Entered A Wrong Option");

}

}

}

**public** **static** **void** main(String[] args) **throws** NotAOperatorException {

// Initialize the database with some sample card data

*dataBase*.put(222222221L, **new** AxisDebitCard(222222221L, "yashas", 50000.0, 2222));

*dataBase*.put(3333333331L, **new** SBIDebitCard(3333333331L, "Akshay", 55000.0, 3333));

*dataBase*.put(4444444441L, **new** AxisDebitCard(4444444441L, "Das", 32500.0, 4444));

*dataBase*.put(5555555551L, **new** HDFCDebitCard(5555555551L, "Aravind", 71000.0, 5555));

*dataBase*.put(1111111111L, **new** OperatorCard(1111111111L, 1111, "Operator 1"));

Scanner scanner = **new** Scanner(System.***in***);

**long** cardNumber = 0;

**double** depositAmount = 0.0;

**double** withdrawAmount = 0.0;

**int** pin = 0;

// Main loop for ATM operations

**while** (*MACHINE\_ON*) {

System.***out***.println("Please Enter the Debit Card Number:");

cardNumber = scanner.nextLong();

**try** {

System.***out***.println("Please Enter PIN Number:");

pin = scanner.nextInt();

*card* = *validateCredentials*(cardNumber, pin); // Validate card and PIN

**if** (*card* == **null**) {

System.***out***.println("Card validation failed.");

**continue**;

}

*ACTIVITY*.add("Accessed By: " + *card*.getUserName() + " Status: Access Approved");

**if** (*card*.getUserType().equals("operator")) {

*operatorMode*(*card*); // Enter operator mode

**continue**;

}

**while** (**true**) {

System.***out***.println("USER MODE: " + *card*.getUserName());

System.***out***.println("===================================================");

System.***out***.println("|| 1. Withdraw Amount ||");

System.***out***.println("|| 2. Deposit Amount ||");

System.***out***.println("|| 3. Check Balance ||");

System.***out***.println("|| 4. Change PIN ||");

System.***out***.println("|| 5. Mini Statement ||");

System.***out***.println("===================================================");

System.***out***.println("Enter Your Choice:");

**int** option = scanner.nextInt();

**try** {

**switch** (option) {

**case** 1:

System.***out***.println("Please Enter The Amount to Withdraw: ");

withdrawAmount = scanner.nextDouble();

*validateAmount*(withdrawAmount); // Validate withdrawal amount

*card*.withdrawAmount(withdrawAmount); // Withdraw amount

*ATM\_MACHINE\_BALANCE* -= withdrawAmount; // Update ATM balance

*ACTIVITY*.add("Accessed By " + *card*.getUserName() + " Activity: Amount Withdrawn "

+ withdrawAmount + " From Machine");

**break**;

**case** 2:

System.***out***.println("Please Enter The Amount to Deposit: ");

depositAmount = scanner.nextDouble();

*validateDepositAmount*(depositAmount); // Validate deposit amount

*ATM\_MACHINE\_BALANCE* += depositAmount; // Update ATM balance

*card*.depositAmount(depositAmount); // Deposit amount

*ACTIVITY*.add("Accessed By " + *card*.getUserName() + " Activity: Amount Deposited "

+ depositAmount + " in the Machine");

**break**;

**case** 3:

System.***out***.println("Your Account Balance is: " + *card*.checkAccountBalance()); // Check

// balance

*ACTIVITY*.add("Accessed By " + *card*.getUserName() + " Activity: Checking The Balance");

**break**;

**case** 4:

System.***out***.println("Enter A New PIN:");

pin = scanner.nextInt();

*card*.changePinNumber(pin); // Change PIN

*ACTIVITY*.add("Accessed By " + *card*.getUserName() + " Activity: Changed PIN Number");

**break**;

**case** 5:

*ACTIVITY*.add("Accessed By " + *card*.getUserName() + " Activity: Generating MINI Statement");

*card*.generateMiniStatement(); // Generate mini statement

**break**;

**default**:

System.***out***.println("You Have Entered A Wrong Option");

**break**;

}

System.***out***.println("Do You Want To Continue? (Y/N):");

String nextOption = scanner.next();

**if** (nextOption.equalsIgnoreCase("N")) {

**break**; // Exit user mode

}

} **catch** (InvalidAmountException | InsufficientBalanceException

| InsufficientMachineBalanceException e) {

System.***out***.println(e.getMessage());

}

}

} **catch** (InvalidPinException | InvalidCardException | IncorrectPinLimitReachedException e) {

*ACTIVITY*.add("Accessed By: " + *card*.getUserName() + " Status: Access Denied");

System.***out***.println(e.getMessage());

}

}

// Display a message when the ATM machine is turned off

System.***out***.println("=====================================================");

System.***out***.println("============== Thanks For Using ICCI ATM Machine ==============");

System.***out***.println("===================================================================");

}

}