Experiment 3

Student Name: Sahil Phogat **UID: 22BCS11636**
Branch: B.E CSE **Section/Group: IOT_635 B**
Semester: 6th **Date of Performance:**
Subject Name: PBLJ-II **Subject Code: 22CSH-359**

1. **Aim:** Write a Java program to simulate an ATM withdrawal system. The program should:
   Ask the user to enter their PIN.
   Allow withdrawal if the PIN is correct and the balance is sufficient.
   Throw exceptions for invalid PIN or insufficient balance.
   Ensure the system always shows the remaining balance, even if an exception occurs.

2. **Objective:** To Simulate an ATM withdrawal system with user authentication.
   Implement exception handling for:
   Invalid PIN entry.
   Insufficient balance.
   Ensure the remaining balance is displayed, even if an exception occurs.

3. **Algorithm/Approach:**
   Start

   Define the ATM class with:
   Attributes: pin (preset PIN) and balance (initial balance).
   Methods:
   verifyPin(int enteredPin): Checks if the entered PIN is correct.
   withdraw(double amount): Withdraws money if the balance is sufficient.
   displayBalance(): Displays the remaining balance.

   Create custom exceptions:

InvalidPinException for incorrect PIN entry.
InsufficientBalanceException for low balance.

In the main method:
Prompt the user to enter a PIN.
Verify the PIN; if incorrect, throw an InvalidPinException.
Ask for withdrawal amount.
If balance is sufficient, process the withdrawal; otherwise, throw InsufficientBalanceException.

Handle exceptions:
If an exception occurs, print an appropriate error message.
Always display the remaining balance using a finally block.

End.

4. **Implementation/Code:**

```
package package_Noobie;


import java.util.Scanner;

//Custom Exception for Invalid PIN
@SuppressWarnings("serial")
class InvalidPinException extends Exception {
 public InvalidPinException(String message) {
    super(message);
 }
}
```

```
//Custom Exception for Insufficient Balance
@SuppressWarnings("serial")
class InsufficientBalanceException extends Exception {
 public InsufficientBalanceException(String message) {
    super(message);
 }
}

//ATM class to handle transactions
class ATM {
 private int pin = 1234; // Preset PIN
 private double balance = 5000.0; // Initial balance

 // Method to authenticate PIN
 public void verifyPin(int enteredPin) throws InvalidPinException {
    if (enteredPin != pin) {
       throw new InvalidPinException("Incorrect PIN! Access Denied.");
    }
 }

 // Method to withdraw money
 public void withdraw(double amount) throws
InsufficientBalanceException {
    if (amount > balance) {
       throw new InsufficientBalanceException("Insufficient balance!");
    }
    balance -= amount;
    System.out.println("Withdrawal successful. Amount withdrawn: Rs." +
amount);
 }

 // Method to display balance
```

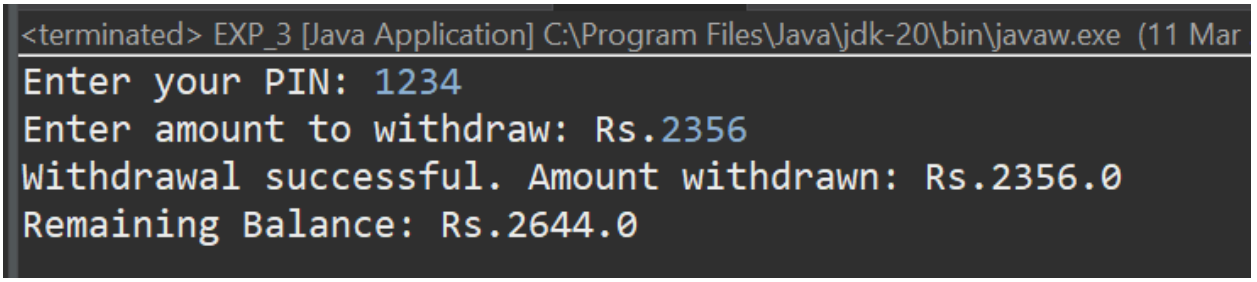
```
 public void displayBalance() {
    System.out.println("Remaining Balance: Rs." + balance);
 }
}

//Main class to run the ATM system
public class EXP_3 {
 public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    ATM atm = new ATM();

    try {
       // Asking user for PIN
       System.out.print("Enter your PIN: ");
       int enteredPin = scanner.nextInt();
       atm.verifyPin(enteredPin);

       // Asking user for withdrawal amount
       System.out.print("Enter amount to withdraw: Rs.");
       double amount = scanner.nextDouble();
       atm.withdraw(amount);
    } catch (InvalidPinException | InsufficientBalanceException e) {
       System.out.println("Error: " + e.getMessage());
    } finally {
       // Always display the balance
       atm.displayBalance();
       scanner.close();
    }
 }
}
```

5. Output



6. Learning Outcome:

- How to create and throw custom exceptions for specific errors.
- Using methods for authentication and transaction processing.
- Implementing a fail-safe mechanism to always show the remaining balance.