LAB-MST

Student Name: Ayush Jhalani
UID: 22BCS13289
Branch: BE/CSE
Section/Group: 22BCS_IOT-618/B
Semester: 6th
Date of Performance: 07/03/25
Subject Name: Project Based Learning in JAVA with Lab
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.

Input Example:

Enter PIN: 1234

Withdraw Amount: 5000

Output Example:

Error: Insufficient balance. Current Balance: 3000

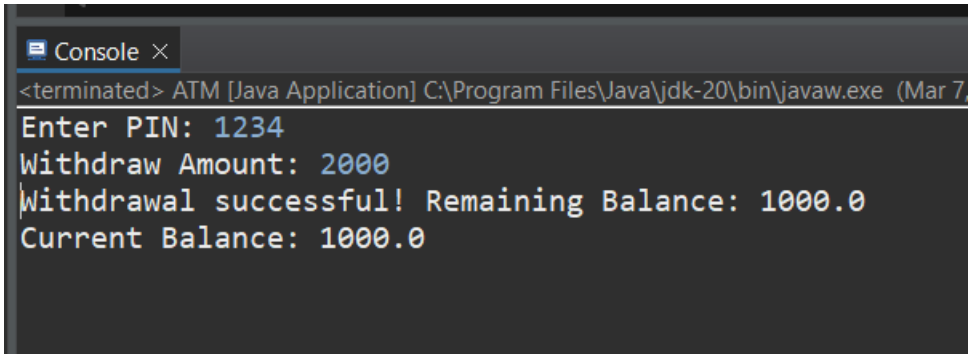
2. **Objective:** The objective of this **Java ATM Withdrawal System** is to **simulate a basic ATM transaction** while ensuring **secure and error-free operations** through proper input validation and exception handling

3. **Implementation/Code:**

```
package ayush; import java.util.Scanner; class InvalidPINException extends Exception {     public
InvalidPINException(String message) {         super(message);
    }
    } class InsufficientBalanceException extends Exception {     public
InsufficientBalanceException(String message) {         super(message);
  }
```

```
} public class ATM {     private static final int CORRECT_PIN = 1234;

private static double balance = 3000;      public static void main(String[] args)

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

      try {
         System.out.print("Enter PIN: ");            int enteredPin = scanner.nextInt();            if

(enteredPin != CORRECT_PIN) {              throw new InvalidPINException("Error: Invalid

PIN.");

         }
         System.out.print("Withdraw Amount: ");            double withdrawAmount = scanner.nextDouble();

if (withdrawAmount > balance) {              throw new InsufficientBalanceException("Error: Insufficient

balance.");           }

         balance -= withdrawAmount;
         System.out.println("Withdrawal successful! Remaining Balance: " + balance);
      } catch (InvalidPINException | InsufficientBalanceException e) {
         System.out.println(e.getMessage());
      } catch (Exception e) {
         System.out.println("Error: Invalid input.");
      } finally {
         System.out.println("Current Balance: " + balance);
      }        scanner.close();

   }
}
```

Output:-



4. Learning Outcomes:

- Understanding Serialization & Deserialization: Learners understand how Java automatically converts between objects and bytes.
- String to Integer Parsing: Demonstrates the use of Integer.parseInt() to convert string inputs into integer values.
- List Handling in Java: Shows how to store and manipulate integers using ArrayList<Integer>.
- Looping & Summation Logic: Reinforces iteration concepts using a for loop to sum up a list of numbers.
- User Input Handling: Teaches how to take space-separated user input and process it effectively using Scanner and split().