**Student Name:** Jayaanth S R

**SuperSet Id :** 6420871

**EXERCISE 1: LOGGING ERROR MESSAGES AND WARNING LEVELS**

**Introduction:**

This Java program, called ATMService, is a simple simulation of how an ATM works. It lets users check their balance, deposit and withdraw money, and also view a mini statement of recent transactions.

**Objective:**

* **Simulate Basic ATM Functions:** The program can handle common ATM actions like checking the current balance, deposit, and withdrawal, all with proper checks in place.
* **Log All Activities:** With the help of SLF4J, the program logs everything that happens. Info logs show successful actions, and errors show things like failed withdrawals due to insufficient funds.
* **Generate Mini Statement:** Every transaction done during the session is saved and printed at the end as a mini statement, giving a quick summary of what happened.

**Implementation :**

**ATMService.java:**

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.ArrayList;

import java.util.List;

public class ATMService {

private static final Logger logger = LoggerFactory.getLogger(ATMService.class);

private static double accountBalance = 1000.00;

private static final List<String> miniStatement = new ArrayList<>();

public static void main(String[] args) {

logger.info("=== Welcome to Java ATM ===");

checkBalance();

deposit(500);

withdraw(300);

withdraw(1500); // Insufficient funds

deposit(-50); // Invalid amount

withdraw(50);

printMiniStatement();

}

public static void checkBalance() {

logger.info("Current Balance: ${}", accountBalance);

}

public static void deposit(double amount) {

if (amount <= 0) {

logger.warn("Invalid deposit amount: {}", amount);

return;

}

accountBalance += amount;

logger.info("Deposited: ${}. New Balance: ${}", amount, accountBalance);

miniStatement.add("Deposited: $" + amount);

}

public static void withdraw(double amount) {

if (amount > accountBalance) {

logger.error("Insufficient funds for withdrawal. Requested: ${}, Available: ${}", amount, accountBalance);

miniStatement.add("Failed Withdrawal Attempt: $" + amount);

return;

}

accountBalance -= amount;

logger.info("Withdrawal successful: ${}. Remaining Balance: ${}", amount, accountBalance);

miniStatement.add("Withdrawn: $" + amount);

if (accountBalance < 200) {

logger.warn("Low balance alert! Current Balance: ${}", accountBalance);

}

}

public static void printMiniStatement() {

logger.info("=== Mini Statement ===");

if (miniStatement.isEmpty()) {

logger.info("No transactions yet.");

} else {

miniStatement.forEach(txn -> logger.info(txn));

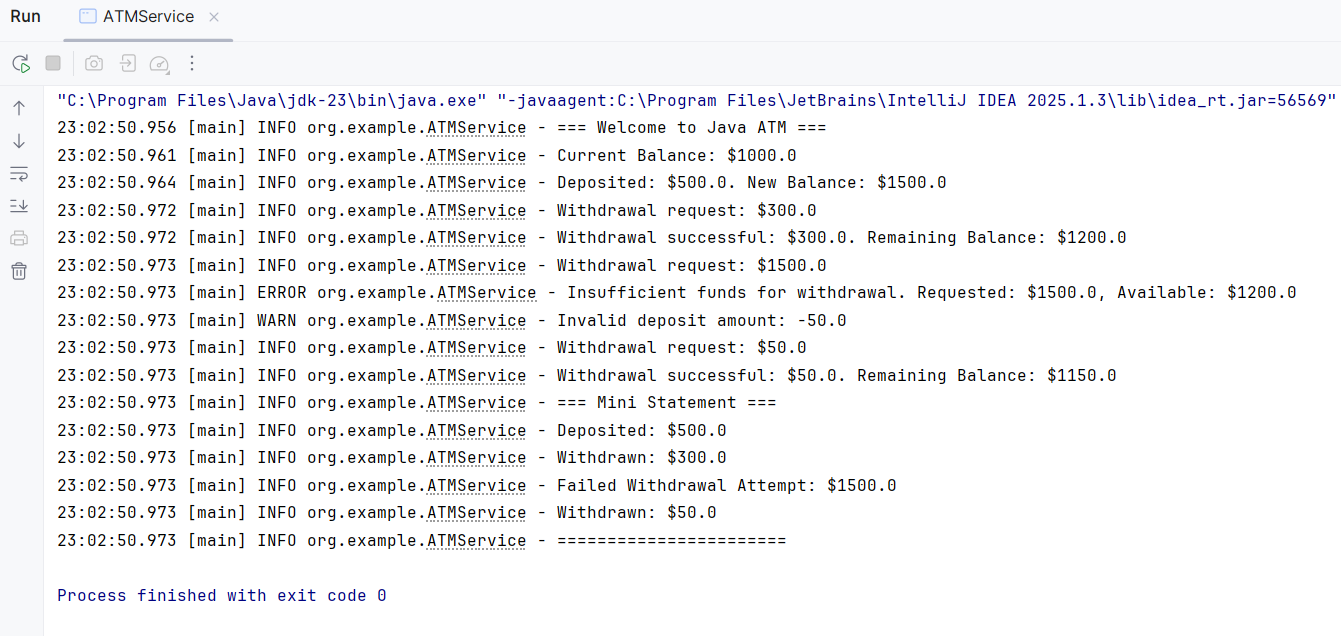
}

logger.info("=Thank You =");

}

}

**Output:**

****

**Conclusion:**

This simulation offers a concise demonstration of ATM transaction handling with logging support. It helps monitor real-time actions and ensures correct financial logic with clear visibility into operations.