**Student Name:** JANARTHIKAN V

**Registration No:** 22CSR076

**Course/Batch:** KONGU ENGINEERING COLLEGE (B.E COMPUTER SCIENCE AND ENGINEERING)

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

**Introduction:**

The ATMService Java program simulates basic ATM functionalities such as checking balance, depositing, withdrawing money, and printing a mini statement.  
It also logs important activities and warnings using SLF4J for better traceability and error tracking.

**Objective:**

* **Transaction Simulation**: Implement deposit, withdrawal, and balance-check operations with appropriate validations.
* **Logging Integration**: Use SLF4J logging to provide clear, categorized logs (info, warn, error) for every transaction.
* **Statement Generation**: Maintain a mini statement log to track the history of user transactions during the session.

**Implementation Breakdown:**

**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 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) {

logger.info("Withdrawal request: ${}", 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miniStatement.add("Withdrawn: $" + amount);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 {

for (String txn : miniStatement) {

logger.info(txn);

}

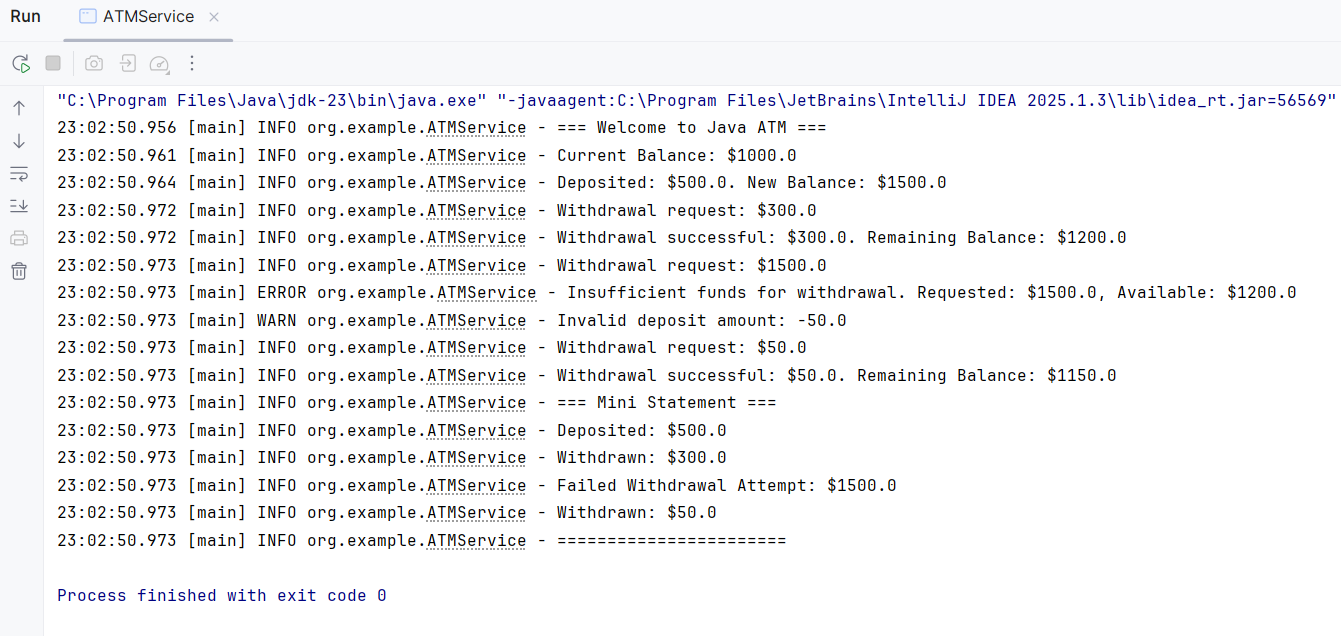
}

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

}

}

**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.