SLF4J Logging Example

# Introduction

Logging is an essential part of application development. It helps developers track the flow of execution, identify issues, and analyze behavior. SLF4J (Simple Logging Facade for Java) provides a simple facade or abstraction for various logging frameworks like Logback, Log4j, and java.util.logging. In this example, we will demonstrate how to log error messages and warning levels using SLF4J with Logback.

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miniStatement.add("Withdrawn: $" + amount);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

}

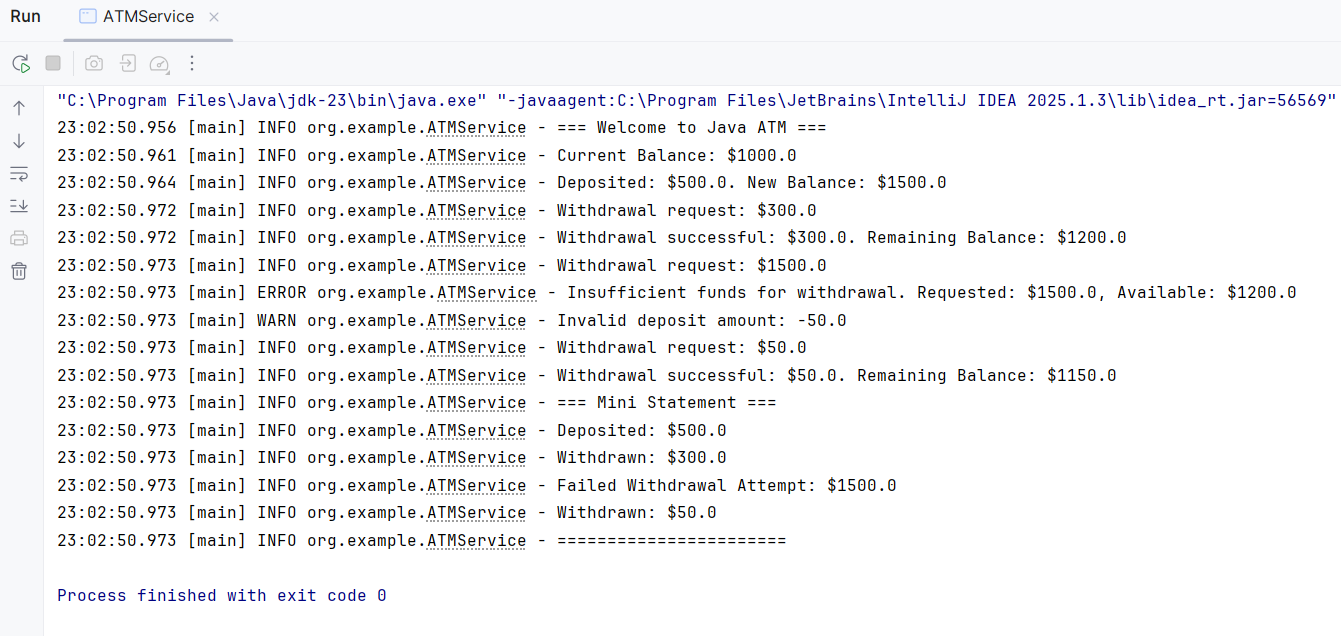
}

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.