CS331

Assignment II: Advanced Java Concurrent Programming

This document serves as the README file for the submission of 2nd assignment of CS331 (Jan-May 2025) by:

Adarsh Gupta

220101003

B. Tech 3rd Year CSE

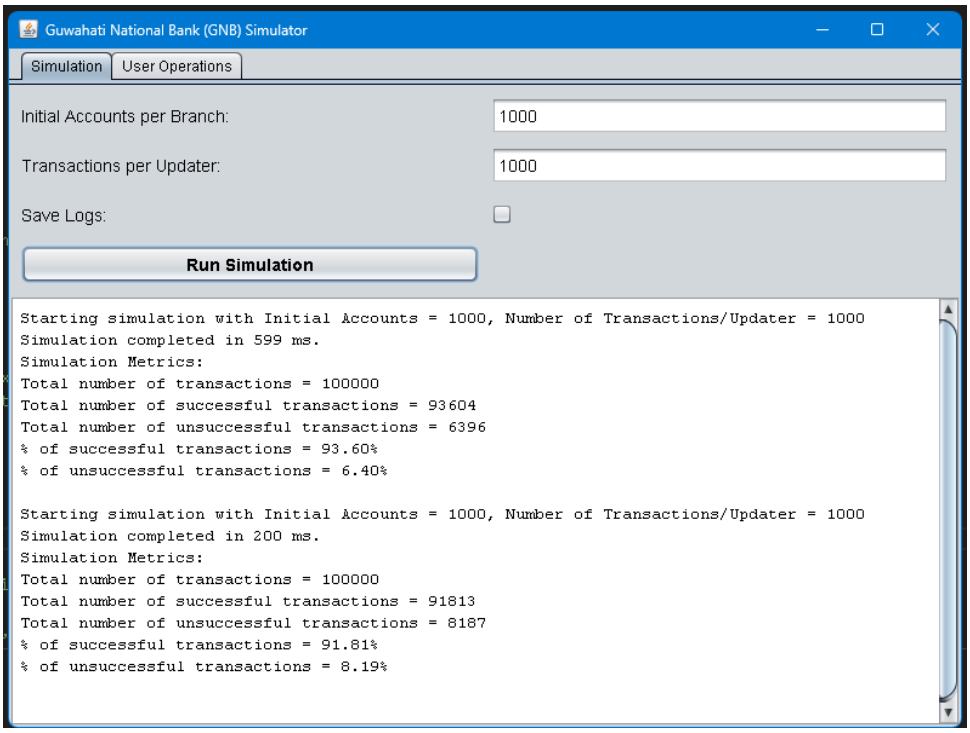
Table of Contents:

Problem Statement & Implementation Overview

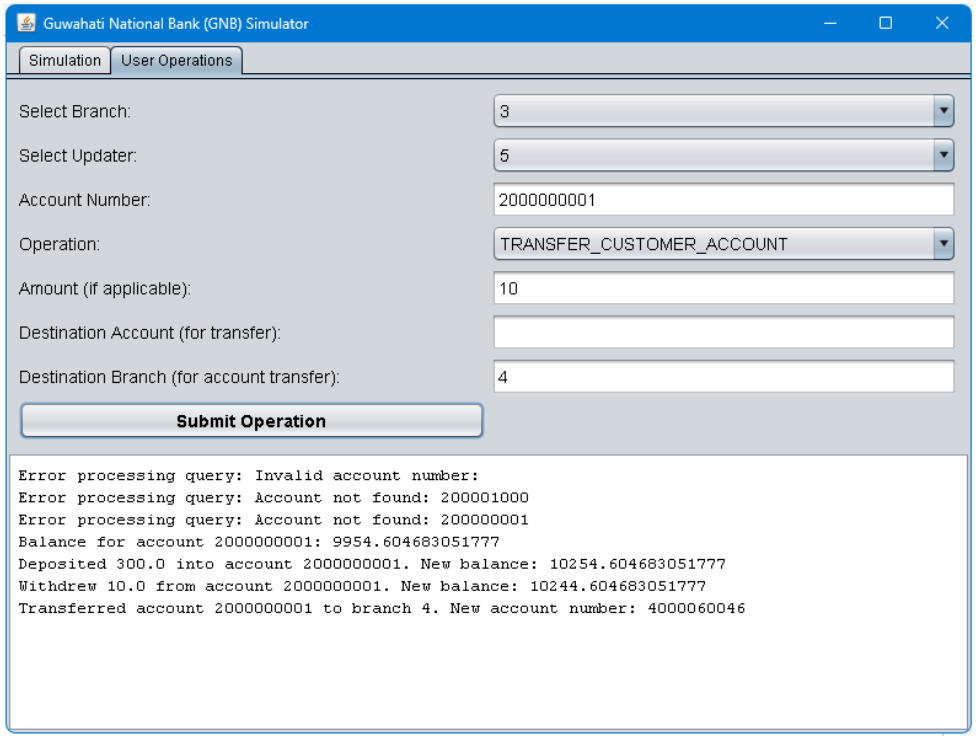
The goal of the assignment was to design and implement a Java Multithreaded program to simulate a bank money transaction system. The bank has 10 branches and 10 concurrently running updaters at each branch.

The solution is implemented by modelling various thread-safe classes such as Account, BranchDatabase, BankDatabase, Query and Updater. I have also provided various modes of operation for the program:

1) Simulation Mode: The user can enter the number of initial accounts to be created, the number of queries to be simulated per updater and whether or not to save the logs. Following this everything is done automatically (a log file called output.txt is saved if you enable the `Save Logs` option)



2) User Mode: The user can even perform their own operations by selecting the Branch Number, Updater Number, Type of transactions and providing the relevant details.



How to Run

The provided solution has the following directory structure:

```
root:
  |- BankGUI.java: Contains the GUI code and callbacks to relevant
functions
  |               from other classes
  |- Simulator.java: Contains implementation of the simulator
  |- Bank: Contains the Bank JAVA Package:
      |- Account.java: Account Class
      |- BranchDatabase.java: Contains the implementation of a Branch
      |- BankDatabase.java: Exposes asynchronous functions to the updater
      |- Constants.java
      |- Query.java: Contains all data required to execute a transaction
      |- Updater.java: Concurrently running Updater class
      |- Exceptions: Contains custom defined exceptions:
            |- InsufficientFundsException.java
```

To compile the code (assuming you have Java OpenJDK installed):

In the root folder execute:

```
root> javac -d build BankGUI.java
```

Following this to start the program:

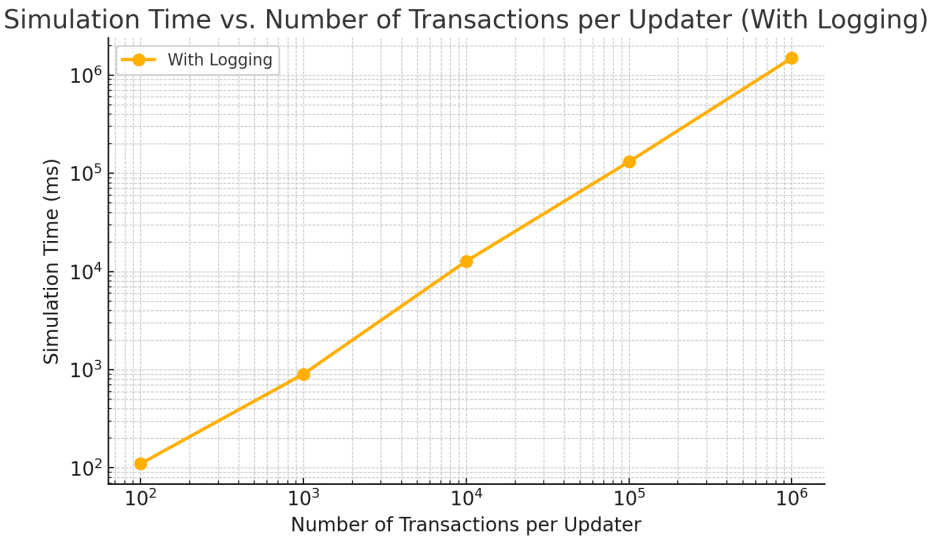
```
root> cd build
root/build> java BankGUI
```

You should now see the Guwahati National Bank Simulator popup.

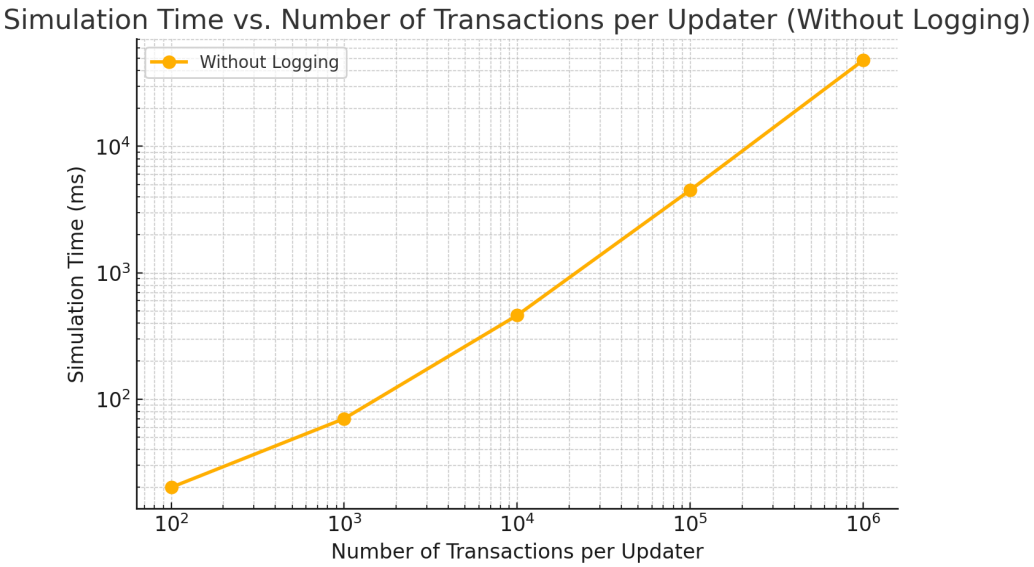
Performance Evaluation

This section is broken into 2 parts comparing performance when saving logs and when not saving logs.

Saving Logs



Without saving logs



It takes approximately 24 minutes to run 10^6 transactions per updater when saving logs. The final output file is ~8.8Gib large! There is a huge performance increase from 24 minutes → 45 seconds for the same number of operations. Indicating the bottleneck in this process is I/O.