Assignment 01 Concurrent Programming in Java

By Shivam Bansal, Roll No. 170101063

1. Sock Matching Robot

How to run

```
make answer-01
```

Role of concurrency

- In the described system all the machines — multiple Robot arms, the Matching machine, the Shelf manager robot — are working along one another, i.e. they are all working concurrently.
- The Robot arms concurrently pick socks from the heap and place them in front of the Matching machine, which is concurrently matching pairs of socks whenever possible.

Role of Synchronisation

Synchronisation is very important in the described system as there are various objects that are being shared between concurrently running tasks.

- The Socks stored in the heap are shared objects between the Robot arms
- The Matching machines buffer is shared between the Robot arms and the Matching machine
- The Shelf manager robots buffer is shared between the Matching machine and the Shelf manager robot

Hence, synchronisation is crucial to prevent race conditions and other problems that can arise if multiple threads work on the same object simultaneously.

Solution

Flow of system:

Heap of socks → (Robot arms) → Matching buffer → (Matching machine) → Shelf buffer → (Shelf manager robot) → White, black, blue, grey shelves

- All machines — Robot arms, Matching machine, Shelf manager robot — were run as separate threads that do as needed if socks are available in their buffers
- `synchronized` modifier was used to control access to buffers that are shared between multiple machines — Heap of socks is shared by robot arms; Matching buffer is used by Robot arms and Matching machine; Shelf buffer is used by Matching machine and Shelf manager robot

2. Evaluation System

How to run

```
make answer-02
```

Importance of concurrency

- Multiple agents are working on a set of common assets
- Course coordinator (CC), Teaching Assistant 1,2 (TA1, TA2) are the multiple agents, they may be working simultaneously
- Stud_Info.txt, Sorted_Roll.txt, Sorted_Name.txt are the common assets, these are shared between all agents
- Concurrency is important to allow everyone to edit the file at the same time
- If there were no concurrency, each agent would have to wait for everyone else to complete their work before starting themselves

Shared resources

There are three files that are shared resources —

1. Stud_Info.txt — Agents edit marks in this, simultaneously
2. Sorted_Name.txt — Agents may generate this file, simultaneously
3. Sorted_Roll.txt — Agents may generate this file, simultaneously

If we don't do synchronization

We may face many problems if synchronization is not taken care of —

- Data read may be stale, e.g. one agent makes an edit but simultaneously another agent reads the file, then data read may be stale and not reflect edit made by first agent.
- Simultaneous edits may be lost, e.g. two agents edit file, then one agent writes to file, then second agent writes to file. Edit done by first agent is lost because neither agent knows of edit made by the other.
- Data may be malformed in writing, e.g. two agents edit different records of file. Both write their changes simultaneously, so length of file changes unexpectedly and their writer's seek position is not where they think it is.

Solution

Concurrency

- To enable concurrency, any number of programme instances can be run simultaneously, working on the same files
- Each user — CC, TA1, TA2 — can run their own instances of the programme and edit as they wish

Synchronization

- `FileLock` (reference) functionality was used to acquire lock on file before reading or writing
- Whenever reading from file, programme acquires a shared lock such that simultaneous reads are not blocked
- Whenever writing to file, programme acquires an exclusive lock such that data in file is not malformed due to simultaneous writing

3. Different Calculator

How to run

```
make answer-03-a    # Run variant (a)
make answer-03-b    # Run variant (b)
```

Concurrency

- Since program is running with GUI, there is a main thread managing drawing of GUI
- Child threads (callbacks) are spawned whenever any event occurs, in our case whenever user presses a button using Enter key
- Timer runs in it's own thread, spawning callbacks at regular intervals to enable highlighting of next button

Synchronization

- All shared state is managed by Java SWING library
- All synchronization is handled by Java SWING library

Solution

- Press 'Enter' or 'Space' to press highlighted button
- Press '=' button to calculate result
- See calculated result on screen