CS431 Programming Lab

Assignment 1

Priyanshu Singh 170101049

Sock Matching Robot

- *Role of Concurrency and Synchronization in above system*

Concurrency: There are multiple robotic arms in the system that simultaneously pick up socks from the heap, thus allowing concurrency in the system. Apart from that, Shelf Manager and Matching Machine is also working simultaneously with Robotic arms, which makes the system faster and concurrent.

Synchronization: Because multiple robot arms are working together to pick up socks, there must be a mechanism that prevents different robotic arms to pick up the same sock. Hence, we have to synchronize picking of socks by robotic arms. Also, sending of socks to Matching Machine's buffer should be done synchronously as Matching machine can only send one pair at a time to shelf manager.

- *How concurrency and Synchronization was handled*

Concurrency: Concurrency is achieved by **multithreading**. There is a separated Java thread for each robotic arm that runs independent of other threads and these threads are executed concurrently.

Synchronization: Synchronization is achieved by using **binary semaphores** associated with each sock. Semaphore is an abstract data type used to control access to a common resource in a multitasking system. Initially all the semaphores are at the value 0, whenever a robotic arm chooses a sock, it checks whether it can acquire it. Semaphore can be acquired if its value is 0. If some arm acquires the lock, semaphore's value becomes 1, ensuring that no other robotic arm can choose the same sock. Block synchronization was used as a synchronization method to match socks in Matching Machine's buffer.

Data Modification in Distributed System

- *Why is concurrency important here?*

We have three instructors TA1, TA2 and CC which are independent of each other and should be able to simultaneously update the marks of students. If there's no concurrency, someone would have to wait for the other instructor to complete their

work before getting on with their own work, thereby reducing the speed and efficiency of the system. Concurrency allows the whole process of marks updating to become fast.

- *What are the shared resources?*

The file "Stud_Info.txt" containing the student marks is the shared resource here which needs to be properly handled, making sure data is not corrupted in the updating process.

- *Why synchronization is necessary?*

If synchronization is not taken care of, some updates of marks would result in wrong final marks in the file. It can be seen from an example:

Let us assume student S's initial marks are = 100. Now if both TA1 and TA2 want to update the marks by +10 and +20 respectively, and they both insert their query at the same time. If both of the threads are running without synchronization, the final marks would be 110 or 120 depending on which of TA1 or TA2 thread finished last. The correct answer should have been 130(+30 increase) which would have been achieved if we used proper synchronization.

- *How synchronization and concurrency was handled?*

Concurrency: Concurrency is achieved by **multithreading**. There is a separate Java thread for each instructor that runs parallelly for updating the marks of students. Having individual threads also took into account of priority, with "CC" assigned the maximum priority.

Synchronization: **Blocked Synchronization** was used on code segment accessing the student's record, so that at one time only one of TA1, TA2 or CC can access the records.

*Note that Q1 used Semaphores while Q2 used Block synchronization.