Programming Languages Lab

Assignment 1 - Concurrent Programming

Sahithya Vemuri (170101077)

Sock Matching Robot:

The role of concurrency and synchronization in the above system?

- All the robotic arms will select the socks at the same time and this is where the role of concurrency comes in.
- All the robotic arms are using the same matching machine and the socks which will come under the shared variables. This is where the role of synchronization comes in.
- Sock needs to be accessed such that it will not be accessed by more than one thread.
- Also all the socks must be added into the matching machine in a synchronized fashion.

How did you handle it?

- I've used threads for each of the robotic arms inorder to ensure concurrency.
- For doing this **Thread class** of JAVA has been extended in the classes.
- Just like the working of robotic arms in the system, all these threads will also process independently of each other.
- I've used **synchronized keyword** for the matching machine while adding the socks to the machine and I've also used the **Reentrant lock** system for the socks.
- Locks are used such that only a single thread can use the sock array and choose the sock and all the other threads cannot access the same object simultaneously.
- Just in case if they try to access the same object then their access will be denied.

Data Modification in Distributed System:

Why is concurrency important here?

- Concurrency is important here because we need to ensure that Course Coordinator, Teaching Assistants 1 and 2 can work simultaneously.

What are the shared resources?

- Shared resources here are the records of the Stud_Info.txt file.
- All the objects formed using this file information are also shared resources.

What may happen if synchronization is not taken care of? Give examples

- If synchronization is not taken care of then the marks that are modified by the teachers will not get updated properly in the file i.e., the result won't be correct.
- Let's consider an example where the Stud_info.txt is as follows

Roll No.	Name	Mail Id	Marks	Teacher
174101055	Amit Kumar Sharma	amit55@iitg.ac.in	75	TA1
174101012	Abdul Najim	abdul12@iitg.ac.in	29	TA2
174101058	Kunal Kishore	kunal58@iitg.ac.in	67	TA2
174101033	Subhra Shivani	subhra33@iitg.ac.in	53	CC
174101035	Savnam Khatam	savanam35@iig.ac.in	88	TA1

- Let's say TA1 wants to decrease the marks of 174101055 by 10. TA2 wants to increase the marks of 174101055 by 15.
- If synchronization is not taken care of then the final marks after updation will be either 65 by TA1 or 90 by TA2.
- If synchronization is taken care of then the final marks of 174101055 will be correct and that will be 80 (75 - 10 + 15).

How you handled concurrency and synchronization?

- I've used **multithreading** to handle the concurrency. I've created threads for each of TA1, TA2 and CC by extending the Thread class of java in the Class I've created to modify the records, thus they can work concurrently now.
- To ensure synchronization I've created locks, **Reentrant locks** for the record level locks and normal semaphore locks which I've implemented for the file level locks.

Calculator For Differently Abled Persons:

Implementation

- I’ve implemented the calculator in two different stages as mentioned in the problem description.
- The packages used to build this calculator are **swing** and **awt** for the implementation of user interface (UI).
- I’ve used **Java keylistener** which is in awt for the sake of keyboard events.
- And mainly **SwingWorker** for handling concurrency.
- SwingWorker is a multi threading utility class for the Swing library. It enables proper use of the event dispatching thread.
- Buttons need to be highlighted periodically after some specified time intervals, as the UI will not respond in between, the timer cannot run on the Event Dispatch Thread.
- Hence highlighting the numbers and functions periodically was done by using separate threads from the main Event Dispatch Thread.
- Instead of running the timer, thread is temporarily suspended for a specified time which I’ve fixed it to 1.2s and doing this is more efficient.
- The user interface was created using a **Jframe** object of swing. **JLabel, JTextField** and **JButton** classes were used for labelling, text areas and buttons respectively.
- Logic was implemented by using different states of input in the program and it moves from one state to another based on the user input.
- For handling the input numbers java **BigDecimal** objects are used.
- For key press events, keyListener interface was implemented and it runs on the main Event Dispatch Thread as it was added to the Jframe object.