CS431: Programming Languages Lab (2020)

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Assignment Set I

- **Assignments will be evaluated by the TAs.**
- **You should submit report (for answering non-code related questions, if any), complete source codes and executable files (whichever applicable).**
- **All codes must be properly documented and good code writing practice should be followed (carry marks).**
- **Copying is strictly prohibited. Any case of copying will automatically result in F grade for the whole course, irrespective of your performance in the other parts of the lab.**
- **Submission deadline: October 08, 2020.**
- **Marks distribution: 20, 30, 50.**

1. Sock Matching Robot: (20 Marks)

There is a heap of new labelled socks. Each sock is having one of four colors: white, black, blue and grey. There are several robotic arms which can pick up a single sock at a time and pass it to a *matching machine*. The matching machine is able to find two socks of the same color and pass the pair of socks to a *shelf manager* robot. The shelf manager robot then puts the pair of socks to the appropriate shelf.

Write a java program to simulate the sock sorting system. Write a brief report highlighting the following.

a) The role of concurrency and synchronization in the above system.

b) How did you handle it?

2. Data Modification in Distributed System: (30 Marks)

The evaluation process of the B. Tech final year students is going to be conducted for the Programming Languages Lab. The evaluation will be done by the course coordinator (CC) and two teaching assistants (TA1, TA2) of the department. A file (named Stud_Info.txt) is used to record the students' data with the following fields (shown along with some sample data).

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

File entries are made based on the following assumptions.

Assumptions:

1. The file can be updated by TA1, TA2 or CC.
2. CC is having higher priority than TA1 and TA2.
3. TA1 and TA2 are having equal priorities, i.e. if TA1 is modifying a field of a record, it can be modified by TA2 later and vice versa.
4. If CC is modifying any data, that can only be modified by CC later and not by any TAs.
5. The file can be assessed by TA1, TA2 or CC simultaneously.
6. There can be negative marks for students.

You are asked to write a Java program for the evaluation system. The system should be able to perform the following.

(a) Allow the CC, TA1 and TA2 to update the Stud_Info.txt.
(b) Should be able to generate a file containing all the students data in sorted order based on the Roll No. Call this file Sorted_Roll.txt.
(c) Should be able to generate a file containing all the students data in sorted order based on the Name. Call this file Sorted_Name.txt.

Your program should allow the updating of Stud_Infor.txt at two levels: record level and file level.

Record level modification example: If the mark of Amit Kumar Sharma is modified from 75 to 80 by TA2 and TA1 want to decrease the mark by 3, the sorted files should contain the 77 mark for Amit.

File level modification example: If the mark of Kunal Kishore is modified from 67 to 70 by TA1 and TA2 want to decrease the value by 5 mark for Savnam (88 to 83), both should be reflected in the sorted files.

Along with the Java program, submit a report answering the following.

I. Why concurrency is important here?

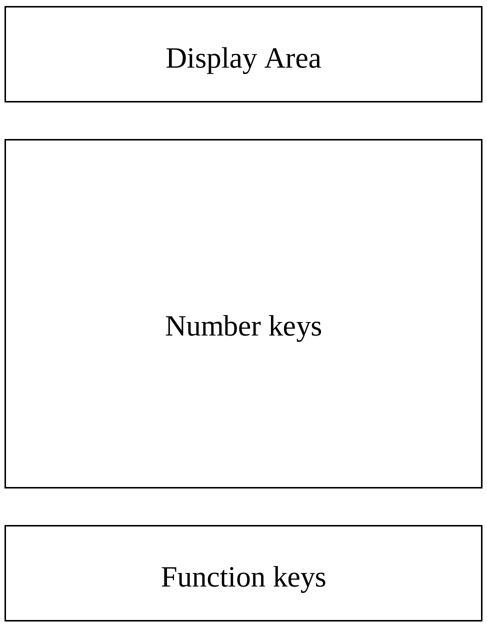
II. What are the shared resources?

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

IV. How you handled concurrency and synchronization? (**You must use a different synchronization technique than you used in Q1**)?

3. Calculator for Differently Abled Persons: (50 Marks)

Consider a simple calculator interface. The interface looks as follows.



The "number keys" area contains the numeric keypad (0-9). The "function keys" contain four function keys: add, subtract, multiply and divide. To perform any function, the user has to "select" one number from the number keys, "select" the function key and finally "select" the other number (e.g. select 5, followed by "+" followed by 6 to perform 5+6). The result is displayed in the "display area".

We want to design the calculator with a different input (selection) mechanism: instead of using mouse to select directly from the respective areas, we'll design a "scanning input" mechanism; an alternative input technique useful for physically disabled users. In this, the selectable on-screen elements are periodically *highlighted* (i.e., change in color of the key). To select an element, the user has to wait for the element to be highlighted and then press "enter" key to select it. We'll try to implement following variations of the above scheme.

(a) The number keys will be periodically highlighted first. Once a number is selected (using the "enter" key), the function keys will be periodically highlighted. After a function key is selected, the periodic highlighting returns to the number keys for the selection of the second number. Thus, to perform the calculation 3+2, the user makes the following sequence of operation: wait for 3 to be highlighted → press 'enter' → wait for '+' to be highlighted → press 'enter' → wait for 2 to be highlighted → press 'enter'.

(b) Note that in the above input mechanism, the user cannot select multiple numbers before a function is selected. To avoid this, we can have both the number keys and the function keys periodically highlighted simultaneously. To select a number, the user has to press "enter" key. To select a function, the user has to press the "space" key. Thus, to perform the calculation 33+25, the user makes the following sequence of operations: wait for 3 to be highlighted → press 'enter' → wait for 3 to be highlighted → press 'enter' → wait for '+' to be highlighted → press 'space' → wait for 2 to be highlighted → press 'enter' → wait for 5 to be highlighted → press 'enter'.

Implement the calculator with the above input methods using the JAVA concurrency constructs and the SWING library for the interface. Note the following.

i) Highlighting can be implemented by changing color of the key
ii) A key should be highlighted for a specific time interval (you can fix it. However, it should be long enough to allow the user time to select the key).
iii) The highlighting is periodic (i.e., highlighter moves from one key to the next until the program terminates).
iv) You can have an explicit “stop” mechanism (a button) to indicate the end of selection before the result is displayed. Note that such buttons will also be selected through scanning input only.