Assignment set 1 (Concurrent programming)

- **Assignment will be evaluated by the TAs.**
- **You should submit report (for answering non-code related questions), complete source codes and executable files.**
- **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 for the whole course, irrespective of your performance in the other parts of the lab.**

- **Submission deadline: 11th September, 2016**

- **Total weight = 25%**
- **Marks distribution: 30, 10, 60**

1. Consider your smart-phone. As you know, it consists of many sensors Suppose you want to design an app that accumulates raw data from various sensors, process the raw data (pre-process), performs some computation on these data (data fusion) and finally produce some output. For simplicity, let us assume the following.

   a) Each sensor returns 8-bit binary strings as raw data. Write a program to generate random binary strings.
   b) In pre-process, these strings are converted to integers. Write a program to convert the string to integer.
   c) In the data fusion stage, the various sensory inputs (in the form of integers) are fused in three ways: averaged, multiplied and added. Write programs for each of these operations.
   d) The result of each fusion operation is compared with a threshold value. If the computed value exceeds the threshold, the program outputs “state detected from (ops)”. Otherwise the program outputs “state not detected from (ops)”. Assume for average the threshold is 100, for multiplication the threshold is 100000 and for addition, the threshold is 10000.

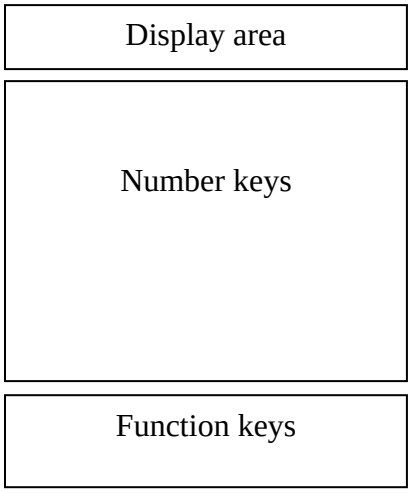
   Assume that there are 10 such sensors from which data are coming continuously. Note that the above tasks can be modeled as sequential process or as concurrent process. In this assignment, you are asked to do the following.

   i) Write a short note on the benefit of having concurrency in modeling the problem (in the report).
   ii) Propose a concurrent solution for the problem (in the report).
   iii) Implement the proposed solution using concurrency constructs in JAVA. You should show two implementations. In the first, you use the older Java constructs (Thread/Runnable & Synchronization keyword). In the second, you should use constructs from the Java.util.concurrent package.

Marks will be proportional to the degree of concurrency in your proposed solution.

2. In the above assignment, we make a small addition. In the pre-processing stage, we also sort the sensory data using merge-sort. Use the fork-join framework to perform this step. Other steps should be same as before (you can show it for any one of the two implementations).

3. 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 till the program terminates).
iv) You can have 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