SFDV3006 Concurrent Programming
Tutorial / Lab – 3

MCQs

1. Which of the following is not true about monitors?
a) monitors permit parallelism
b) monitors prevent interference and race condition
c) monitors have condition synchronization
d) monitors can notify threads which are waiting for a condition

2. Monitors must have _____
a) data encapsulation and public methods
b) private data and private methods
c) public methods and public data
d) a) and b)

3. Monitors are _____
a) active entities
b) passive entities
c) both a) and b)
d) a special kind of threads

4. Which of the following should be used if a thread needs to wait for a condition?
a) wait()
b) yield()
c) sleep()
d) ; //no-op empty statement
e) any of the above will work correctly

5. Methods of monitor need to be _____
a) private and synchronized
b) public but not synchronized
c) private but not synchronized
d) public and synchronized

6. The situation where every thread waits for every other thread and there is noprogress is called _____
a) livelock
b) spinlock
c) deadlock
d) bounded buffer problem

7. wait() and notify() methods are inherited from which class?
a) Thread class
b) Object class
c) From the Runnable interface
d) a) and c)

8. A thread calls the wait() method
a) whenever it wants to enter the wait set
b) whenever it wants to enter the entry set
c) when the lock for the object is not available
d) when it enter a monitor method and finds that it cannot continue due to some condition

9. When a thread calls wait() which of the following does not happen
a) it releases the object lock and is blocked
b) is put into the entry set
c) is put into the wait set
d) another thread will acquire the object lock

10. When a thread changes the condition of the monitor it should _____
a) do nothing
b) wait for the other thread to enter the monitor
c) call notify() or notifyAll()
d) execute its critical section

11. notify() will always select the _____ thread in the wait set
a) first
b) last
c) middle
c) any random thread

12. Which of the following happens when notify()/notifyAll() is called on a thread?
a) thread goes into RUNNING state
b) thread goes into RUNNABLE state and acquires the object lock
c) thread goes into RUNNABLE state and is put into entry set
d) thread changes the condition of the monitor

Short Answer Questions:
1) What are monitors? What are the benefits of using them?
2) How do you make a Java class a monitor?
3) Explain why using wait() is better than using Thread.yield() or Thread.sleep().
4. Give some examples of the type of problems that can happen when notify() or notifyAll() is not used inside monitors.

Tutorial Problems:
Note : For all these problems your could should not go into deadlock or livelock or starvation.

1. Write a Java monitor called *SingleSlotBuffer* where the size of buffer is one. Implement the *put()* and *take()* which will be called by the producers and consumers and show the condition synchronization. The buffer can hold any instance of Object.

2. Write a Java monitor called *UnlimitedBuffer* where is size of buffer is unlimited. Implement the *put()* and *take()* methods which will be called by the producers and consumers and show what will be the condition synchronization. The buffer can hold many instances of Object.

3. Modify the *UnlimitedBuffer* to allow the Consumer to run whenever there are at least 10 items in the buffer.

4. From the lecture modify the *Database* monitor to allow a maximum of 10 concurrent readers.

Lab Exercises:

Code from the Lecture – Use the code provided on the website

1. Run the different implementation of the *BufferInterface* - as discussed in the lecture you should see deadlock with *yield()*;
Uncomment the code for the different implementation of the *BufferInterface* and see the output.
2. Use VisualVM to look at the state of the threads for each of *BufferInterface* implementations