**Report 5**

**Java Synchronization:**

Synchronization in java is the capability to control the access of multiple threads to any shared resource. Java Synchronization is useful when we want to allow only one thread to access the shared resource.

Synchronization in Java is an important concept since Java is a multi-threaded language where multiple threads run in parallel to complete program execution. In multi-threaded environment *synchronization of Java object or synchronization of Java class becomes extremely important*. Synchronization in Java is possible by usingJava keywords ***"synchronized"*** and ***"volatile”***.

Concurrent access of shared objects has two possible issues:

1 Thread interference

2 Memory consistency errors

To overcome the two issues we need to use the synchronized keyword properly for the objects to allow the mutual exclusive access to two critical section to two threads.

synchronized keyword involve locking and unlocking. before entering into any **synchronized method or block** thread needs to acquire the lock

Read more: <http://javarevisited.blogspot.com/2011/04/synchronization-in-java-synchronized.html#ixzz4fRcmYUfz>

Advantages:

● To prevent thread interference.

● To prevent consistency problem.

Types:

● To prevent thread interference.

● To prevent consistency problem

Synchronization in Marketplace Application:

In the marketplace application, I have added the synchronize keyword for the tow functionalities, i.e., Browse Item, Add Item, Remove Item because the synchronize keyword will enable to synchronization of the functionality that make the application to be dynamic and the will make the action performed to be the compliance with the time and the speed. The synchronization and the Concurrency will go hand in hand in this application. The change in the one state will reflect in another state.

For Instance: In the online marketplace application, if the user wants to buy a product then the he will first browses for the item and if he wants to buy and he will add the item to the shopping cart. Then the number of items in the browsing list should change as soon as the user adds the item to the shopping cart. If there are 100 units of shoes available in the marketplace then if the user selects 5 shoes and adds it to the shopping cart. Let another user ‘user 2’ from some other geographical location wants to buy shoes in the same marketplace then if the user 2 browses for the shoes then the number of shoes in the browsing list should be the 100-5 = 95 because if the user adds the shoes to the shopping cart and the 5 shoes are removed from the browsing list. If the user again dropped the shoes from the shopping cart then the shoes from the shopping cart are added back to the browsing list. The process of the action performed at the one side should reflect simultaneously the process of the state change synchronized.

The online marketplace application is a distributed system that uses the synchronization and concurrency that provides the dynamic change in the marketplace application. Since the distributed systems communicate with each other and shares the resources and information it is very important to have the synchronization for the to maintain the integrity.

Java provides to make use of the synchronization using the synchronize keyword that will enable to functionality to get synchronize in our marketplace application.

**Thread Synchronization**

There are two types of thread synchronization mutual exclusive and inter-thread communication.

1. Mutual Exclusive
   1. Synchronized method.
   2. Synchronized block.
   3. static synchronization.
2. Cooperation (Inter-thread communication in java)

### Concept of Lock in Java

Synchronization is built around an internal entity known as the lock or monitor. Every object has a lock associated with it. By convention, a thread that needs consistent access to an object's fields has to acquire the object's lock before accessing them, and then release the lock when it's done with them.

**Future Pattern**:

Future pattern doesn’t fit in my architecture. The pattern returns the value to the user when the user request for the service in the future.

Future pattern will avoid the wait for the synchronization.

The future pattern will return when its computation finished. When the client excutes without waiting or blocking the service will not avaible for he wants to use it. So in such a case the client future pattern so he can get the service whenever it is need. The future will return the data object immediately when the invokes the service it waiting time for the service execution of the client.

Thread Safe Interface Pattern: In the concurrent programs this pattern provides the thread safe access to different components such that the access to components are done through threads the components of the concurrent system that contain the critical section so in order to protect the concurrent access to the critical section it provides the lock on it. This mainly involves in the acquiring and releasing locks on the critical sections so that concurrent access doesn’t occur and it only applies the lock on the important section. This Pattern will help to separate the concerns and maintains the high coherence which will separe the locking and functionality issue. This pattern will avoid the concurrent access to the multiple thread on the critical section so it will avoid the deadlock situation such that only one thread can get access to the critical section with the acquiring the lock on the critical section and it releases the lock when it finishes it.

Scoped Locking: The scoped locking will overcome the some of the potential difficulties that we face in the thread safe pattern. The acquiring and releasing of locks are automatically done when entering and leaving the critical section that has the lock on it.

The Synchronized keyword will allow us to have the all the potential uses of the thread safe locks and the scoped locking the thread safe interface pattern. The synchronized keyword will provide the use of the pattern in form of monitor object pattern. In form of the intrinsic or mutex locking it provides the automatic obtaining and releasing locks. The acquing lock or entering the critical component is done by monitorenter and the leaving or releasing locks are done by the monitorexit.

From Java 5 the package java.util.concurrent.locks contains several lock implementations.

Domain Model:

There is no change in my domain model.

**References:**

<http://javarevisited.blogspot.com/2011/04/synchronization-in-java-synchronized.html>

Lecture Slides.