**Research**

**1. Define the following thread synchronization approaches:**

Thread synchronization is a mechanism that is designed to ensure that two or more concurrent processes or threads do not simultaneously execute some particular program segment called critical section.

1. Locks

A lock is a synchronization mechanism that enforces access limitation to a resource in a multithreading environment. A lock sonly allows one thread to enter the part that is locked and the lock is not shared with any other processes.

2. Mutex

Mutex is a popular thread synchronization approach to lock a shared resource prior to using it and to release the lock after using the resource. A mutex is system-wide lock which is set prior to using a resource so that no other thread can have access to the region of code.

3. Semaphores

Semaphore is a thread synchronization approach (generalized mutex) that is used to send signals between threads to avoid missed signals or to guard a critical section one would with a lock. Semaphore maintains a set of permits. Each acquire() blocks if necessary until a permit is available and then takes it. Each release() adds a permit and potentially releasing a blocking acquirer. Semaphores are often used to restrict the number of threads that can access some resources.

4. Synchronized

Synchronized only allows one thread of execution to access a resource at the same time. Synchronized is a Java native mechanism to create threads and synchronizing their tasks by using the synchronized keyword. In a synchronized block, there can only be one thread executing inside at a time. All other threads attempting to enter the synchronized block are blocked until the thread inside exits the block. Synchronized provides a protection by ensuring that a critical section of the code is never executed concurrently by two different threads. Synchronized applies to both blocks and methods.

5. Volatile

Volatile is modifier that can be applied to fields or variables. A thread usually creates its own copy on a local cache of the variable. When this value is updated, it only happens in the local cache copy first, not in the real variable. However, when a variable is declared as volatile, it will not be stored in the local cache of a thread. But, instead, each thread will have access to the same variable in the main memory. Volatile is generally useful for variables that need to be the same if one or more threads have access to the same variables. An example of a variable that would be useful is a Date variable whose data will need to be the same across all threads.

**2. Define Deadlock conditions.**

Deadlock is a common problem that happens when one thread try to another thread’s resource and vice versa but unable to do so because each thread has not given up its lock to share the resource. In other words, a deadlock occurs when a thread is in a waiting state because a resource is being held by another thread that is also in a waiting state. Since these two threads are waiting for each other to release each resource, then a deadlock occurs.

**3. Define Race conditions.**

Race condition occurs in a multi-threaded environment when two or more threads try to access a shared resource at the same time. Since multiple threads are trying to race each other to finish executing, it is called race condition. It is safe if multiple threads only try to read a shared resource as long as they’re not trying to change it at the same time. An example of this would be when there are multiple threads accessing the shared resources such as variables stored in a database. One of the ways to avoid race condition in Java is by using synchronization. The synchronized keyword in Java is designed to synchronize the access to the shared resource.

**4. What is a ORM?**

Object Relational Mapping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages, such as Java. ORM is a framework used to map an object to a relational database. In any relational database, the top level element is table and each table is divided into rows and columns. A column contains values of a particular type of data and a row represents one set of data for a particular table. In an Animal class example that has species, age, sound variables, columns in the table are species, age, and sound. A row would be the complete information of one animal.

**5. What is a memory leak?**

Memory leak is a situation that occurs when there are objects present in the heap that are no longer used, that the garbage collection process is not able to remove from the memory. This unnecessarily maintained situation is called memory leak. There are two types of objects in the heap memory: referenced and unreferenced. Referenced objects are objects who are still active, whereas unreferenced objects are those that are not active any longer. The garbage collector’s job is to remove unreferenced objects periodically, but it never collects referenced objects. Memory leak occurs when the garbage collector is not able to collect all the unreferenced objects.

**6. What is an ANR and what are some common causes?**

ANR stands for Application Non Responsive. The two common causes of ANR are when an activity stops processing events for 5 seconds and when a BroadcastReceiver do not react in 10 seconds. Android has several ways to let you know when an app has an ANR problem. One of the ways is by alerting you that your app is experiencing an ANR. This ANR dialog gives users the opportunity to force quit the app. Methods that run in the UI thread should do as little work as possible in that thread in order to avoid ANRs. Activities should set up as little as possible in onCreate() and onResume() methods. The most effective way to avoid ANRs is to create a worker thread for longer operations. This is achieved by using the AsyncTask. The doInBackground() method is designed to perform the work in the background.