Md Rahman

1.Define the following as it relates to threading:

Locks, Mutex, Semaphores, Synchronized, Volatile, Atomic

2. What is a deadlock condition?

3. What is a race condition?

4. What is a memory leak?

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

1. Define the following:

1. A lock is a mechanism that enforces a limit on access to a resource where there are multiple threads of execution.
2. A mutex (mutual exclusion object) is a locking mechanism (a binary flag) that allows multiple threads to access the same resource but not at the same time. When a resource is free for use, a thread can access it and close the lock, so other threads will have to wait for access until the current thread is complete with its work, where it will then open the lock. Then the next thread acquires the lock, and so on.
3. Mutex and Semaphores are often confused for being the same thing, but there is a difference. In a mutex, a thread will unlock the key and exit the resource after it’s completed its task, and any thread waiting will go in next. In a semaphore, the thread finishing up will signal all other threads that the resource is now free.
4. Synchronized - Threads execute in a sequential manner.
5. Volatile - A keyword in Java that makes a class thread safe. For example, if we have a volatile field, any changes made to it by one thread are immediately reflected in other threads.
6. Atomic - A algorithms designed to handle lock-and-wait algorithms in Java. In traditional locks, only one thread wins access to a resource and the other threads are suspended (i.e. in a waiting loop) until that thread has completed its work. This can become performance costly as programs get larger and more threads become suspended.

This is where Atomic operations come into play. They exploit low-level machine instructions like compare-and-swap, so when multiple threads attempt to access the same resource, one of them wins access and the other threads go back to their original work, instead of being forced to wait. We typically use an atomic variable such as AtomicInteger or AtomicBoolean. These variables have values such as get() and set(), so when one thread updates the value, that change is immediately visible to other threads.

2. What is a deadlock condition?

When thread A has hold of a resource, and thread B has hold of a different resource, and each thread wants the other’s resource, but neither are willing to release theirs.

3. What is a race condition?

When two or more threads want access to the same resource at the same time, making the resource inaccessible for all of them.

4. What is a memory leak?

A memory leak is a type of resource leak where a portion of memory that’s no longer needed is not released. Another example is when an object is improperly stored in memory and becomes unreachable, thus making it difficult to release that memory allocation.

5. What is ANR and what are some common causes?

ANR stands for Application Not Responding. This happens when a process is run on the UI (main) thread for longer than 5 seconds. During this time, the GUI the user interacts with will lock up, making it appear as if the screen froze. After 5 seconds, if the thread has not recovered, a dialogue box is shown to the user saying “Application Not Responding. Close app or Wait?”

A typical mistake novice developers make is use the main thread to access information from a database. Databases can be very large, so doing any operations will be time consuming, causing the screen to appear frozen. Another example is when a background activity is running, and it’s consuming a considerable amount of resources unknown to the user. It could potentially cause the main thread to freeze up.