**Multi-threading**

Life cycle of a Thread (Thread States)

In Java, a thread always exists in any one of the following states. These states are:

1. New
2. Active
3. Blocked / Waiting
4. Timed Waiting
5. Terminated

Q. Difference b/w Wait() and Sleep()

|  |  |
| --- | --- |
| ***Wait()*** | ***Sleep()*** |
| The Wait() method is related to the Object class. | The Sleep () method is related to the Thread class.  The Sleep () method does not release the lock on the object during Synchronization. |
| It is not a static method. | It is a static method. |
| At the time of the Synchronization, the Wait() method releases obj. | At the time of the Synchronization, the Sleep() method doesn't release the obj, i.e., lock. |
| We can call the Wait () method only from the Synchronized context. | We can call the Sleep () method from outside the Synchronized context. |
| The Sleep() method has two overloaded methods, which are as follows:   * sleep(long milliseconds, int nanoseconds) * sleep(long milliseconds) | The Sleep() method has three overloaded methods, which are as follows:   * Wait() * wait(long timeout, int nanoseconds) * wait(long timeout) |
| The constructor of the Wait() method is defined in the following way: public final void Wait(long timeout) | The constructor of the Sleep () method in the following way: public static void Sleep (long millis) throws Interrupted\_Execption |

**Q. What is Race Condition in Multi-threading?**

A race condition occurs **when two or more threads can access shared data and they try to** change it at the same time.

Race conditions are most commonly associated with computer science and programming. They occur when two computer program processes, or threads, attempt to access the same resource at the same time and cause problems in the system. Race conditions are considered a common issue for multithreaded applications.

**Q. What is Slipped Condition in Multi-threading?**

**Slipped Condition** is a special type of [race condition](https://www.geeksforgeeks.org/operating-system-process-synchronization/) that can occur in a multithreaded application. In this, a thread is suspended **after reading a condition** and **before performing the activities** related to it. It rarely occurs, however, one must look for it if the outcome is not as expected.

## **Q. Difference between synchronization and volatile keyword**

Volatile keyword is not a substitute of a synchronized keyword, but it can be used as an alternative in certain cases. There are the following differences are as follows:

|  |  |
| --- | --- |
| **Volatile Keyword** | **Synchronization Keyword** |
| Volatile keyword is a field modifier. | Synchronized keyword modifies code blocks and methods. |
| The thread cannot be blocked for waiting in case of volatile. | Threads can be blocked for waiting in case of synchronized. |
| It improves thread performance. | Synchronized methods degrade the thread performance. |
| It synchronizes the value of one variable at a time between thread memory and main memory. | It synchronizes the value of all variables between thread memory and main memory. |
| Volatile fields are not subject to compiler optimization. | Synchronize is subject to compiler optimization. |

**Q: The Major Difference between User and Daemon Threads:**

|  |  |
| --- | --- |
| User Thread | Daemon Thread |
| User threads are foreground threads. | Daemon threads are background threads. |
| User threads are high priority threads. | Daemon threads are low priority threads. |
| Its life independent. | Its life depends on user threads. |
| JVM wait until user threads to finish their work. It never exit until all user threads finish their work. | The JVM will’t wait for daemon threads to finish their work. The JVM will exit as soon as all user threads finish their work. |
| JVM will not force to user threads for terminating, so JVM will wait for user threads to terminate themselves. | If all user threads have finished their work JVM will force the daemon threads to terminate |
| User threads are created by the application. | Mostly Daemon threads created by the JVM. |
| Mainly user threads are designed to do some specific task. | Daemon threads are design as to support the user threads. |