Thread Priorities, Deamon Threads, Thread Synchronization, Interthread communication and Dead locks.

-K.L.MADHAVI

Thread Priorities

Each thread has a priority. Priorities are represented by a number between 1 and 10. In most cases, the thread scheduler schedules the threads according to their priority (known as preemptive scheduling). But it is not guaranteed because it depends on JVM specification that which scheduling it chooses. Note that not only JVM a Java programmer can also assign the priorities of a thread explicitly in a Java program.

3 constants defined in Thread class:

1. public static int MIN_PRIORITY
2. public static int NORM_PRIORITY
3. public static int MAX_PRIORITY

Default priority of a thread is 5 (NORM_PRIORITY). The value of MIN_PRIORITY is 1 and the value of MAX_PRIORITY is 10.

Example Program

```
Class ThreadPriorityEx extends Thread
{
     public void run()
     {

          System.out.println(“thread running);

      }

     public static void main(String args[])

     {

          // Creating threads with the help of ThreadPriorityEx class

          ThreadPriorityEx th1 = new ThreadPriorityEx();

          ThreadPriorityEx th2 = new ThreadPriorityEx();

          ThreadPriorityEx th3 = new ThreadPriorityEx();

          System.out.println("Priority of the thread th1 is : " + th1.getPriority());

          System.out.println("Priority of the thread th2 is : " + th2.getPriority());

          System.out.println("Priority of the thread th3 is : " + th3.getPriority());
```

Continue…

```
    th1.setPriority(1);

    th2.setPriority(3);

    th3.setPriority(9);

    System.out.println("Priority of the thread th1 after changing is : " + th1.getPriority());

    System.out.println("Priority of the thread th2 after changing is : " + th2.getPriority());

    System.out.println("Priority of the thread th3 after changing  is : " + th3.getPriority());

}
```

```
Output:
Priority of the thread th1 is : 5
Priority of the thread th2 is : 5
Priority of the thread th3 is : 5
Priority of the thread th1 after changing is : 1
Priority of the thread th1 after changing is : 3
Priority of the thread th1 after changing is : 9
```

Deamon Threads

In Java, daemon threads are low-priority threads that run in the background to perform tasks such as garbage collection or provide services to user threads.The life of a daemon thread depends on the mercy of user threads, meaning that when all user threads finish their execution, the Java Virtual Machine (JVM) automatically terminates the daemon thread.

Example of Daemon Thread in Java

Some examples of daemon threads in Java include garbage collection (gc) and finalizer. These threads work silently in the background, performing tasks that support the main execution without interfering with the user’s operations.

Continue…

Properties of Java Daemon Thread

1. **Automatic Termination**: If the JVM detects a running daemon thread, it terminates the thread and subsequently shuts it down. The JVM does not check if the daemon thread is actively running; it terminates it regardless.

2. **Low Priority**: Daemon threads have the lowest priority among all threads in Java.

3. **No Preventing JVM Exit**: Daemon threads cannot prevent the JVM from exiting when all user threads finish their execution.

Thread Synchronization

Types of Synchronization

There are two synchronizations in Java mentioned below:

1.Process Synchronization: Process Synchronization is a technique used to coordinate the execution of multiple processes. It ensures that the shared resources are safe and in order.

2.Thread Synchronization: Thread Synchronization is used to coordinate and ordering of the execution of the threads in a multi-threaded program. There are two types of thread synchronization are mentioned below:

- Mutual Exclusive
- Cooperation (Inter-thread communication in Java)

Continue…

Mutual Exclusive

Mutual Exclusive helps keep threads from interfering with one another while sharing data. There are three types of Mutual Exclusive mentioned below:

- Synchronized method.
- Synchronized block.
- Static synchronization.

Continue…

Cooperation (Inter-thread communication in Java)

Inter-thread communication or **Co-operation** is all about allowing synchronized threads to communicate with each other.

Cooperation (Inter-thread communication) is a mechanism in which a thread is paused running in its critical section and another thread is allowed to enter (or lock) in the same critical section to be executed.It is implemented by following methods of **Object class**:

- wait()
- notify()
- notifyAll()

Continue…

1) wait() method

The wait() method causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

The current thread must own this object's monitor, so it must be called from the synchronized method only otherwise it will throw exception.

2) notify() method

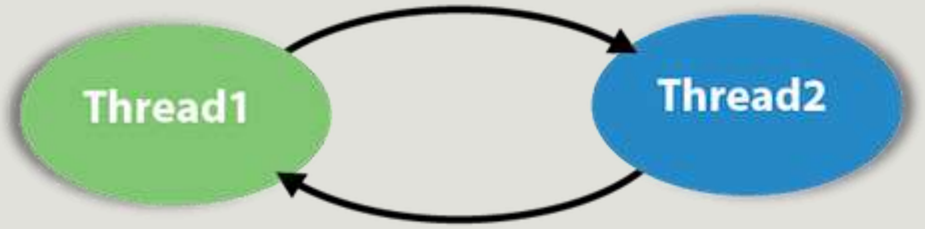
The notify() method wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened.

3) notifyAll() method

Wakes up all threads that are waiting on this object's monitor.

Dead locks

Deadlock in Java is a part of multithreading. Deadlock can occur in a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. Since, both threads are waiting for each other to release the lock, the condition is called deadlock.


Thread1
Thread2