JAVA

INTRODUCTION TO MULTI-THREADING IN JAVA

INTRODUCTION

Process and Threads are two basic units of Java program execution.

- Process: A process is a self contained execution environment and it can be seen as a program or application.
- Thread: It can be called lightweight process
- Thread requires less resources to create and exists in the process
- Thread shares the same address space and process resources

MULTITHREADING

Multithreading in java is a process of executing multiple processes simultaneously ·

A program is divided into two or more subprograms, which can be implemented at the same time in parallel.

· Multiprocessing and multithreading, both are used to achieve multitasking.

· Java Multithreading is mostly used in games, animation etc.

ADVANTAGE:

- It doesn't block the user
- Can perform many operations together so it saves time.
- Threads are independent so it doesn't affect other threads

CREATING THREAD

- Threads are implemented in the form of objects.
- The run() and start() are two inbuilt methods which helps to thread implementation
- The run() method is the heart and soul of any thread
  - It makes up the entire body of a thread
- The run() method can be initiating with the help of start() method.

By Extending Thread class	By implementing Runnable interface

class Multi extends Thread
{
public void run() // run() method declared
{ System.out.println("thread is running...");
}
public static void main(String args[]) {
Multi t1=new Multi();
t1.start(); } }
Output: thread is running…
	class Multi3 implements Runnable
	{
	public void run()
	{
	System.out.println("thread is running..."); }
	public static void main(String args[]) {
	Multi3 m1=new Multi3();
	Thread t1 =new Thread(m1); t1.start(); } }
	Output: thread is running…

LIFE cycle of a thread

- Newborn state -The thread is born and is said to be in newborn state. The thread is not yet scheduled for running. At this state, we can do only one of the following: • Schedule it for running using start() method. • Kill it using stop() method.
- Runnable state - The thread is ready for execution Waiting for the availability of the processor. The thread has joined the queue
- Running state – The processor has given its time to the thread for its execution. • The thread runs until it gives up control on its own or taken over by other threads.
- Blocked state - • A thread is prevented to entering into the runnable and the running state. • This happens when the thread is suspended, sleeping, or waiting in order to satisfy certain requirements. A blocked thread is considered "not runnable" but not dead and therefore fully qualified to run again This state is achieved when we Invoke suspend() or sleep() or wait() methods.
- Dead state - • Every thread has a life cycle.A running thread ends its life when it has completed executing its run( ) method • A thread can be killed in born, or in running, or even in "not runnable" (blocked) condition. • This state is achieved when we invoke stop() method or the thread completes it execution.

➢ Thread priority

• Each thread is assigned a priority, which affects the order in which it is scheduled for running.

• Java permits us to set the priority of a thread using the setPriority() method as follows:
ThreadName.setPriority(int Number);

➢ Deadlock

Deadlock describes a situation where two or more threads are blocked forever, waiting for each other.

• when two or more threads are waiting to gain control on a resource.

For example, assume that the thread A must access Method1 before it can release Method2, but the thread B cannot release Method1 until it gets holds of Method2.

➢ Java synchronization

Generally threads use their own data and methods provided inside their run() methods.

• But if we wish to use data and methods outside the thread's run() method, they may compete for the same resources and may lead to serious problems.

• Java enables us to overcome this problem using a technique known as Synchronization.

For ex.: One thread may try to read a record from a file while another is still writing to the same file.