1. Ways to define a Thread
2. By extending Thread Class
3. By implementing Runnable Interface
4. Method to prevent Thread execution
5. Yield ()
6. Join ()
7. Sleep ()
8. Inter Thread Communication
9. Wait
10. Notify
11. Notify All

**User and Daemon Thread**

1. **User threads** are created by the application (**user**) to perform some specific task. Whereas

**daemon threads** are mostly created by the JVM to perform some background tasks like garbage collection.

1. JVM will wait for **user threads** to finish their tasks. JVM will not exit until all **user threads**

finish their tasks.

**Multitasking:**

Executing several tasks simultaneously is the concept of Multitasking. There are 2 types of multitasking.

1. **Process Based**: Executing several tasks simultaneously where each task is separate independent program(process) is called process based multitasking.

**Ex**: 1. Typing a Java Program in editor.

2. Listening audio songs from system.

3. Downloading a file from internet.

All these tasks executed simultaneously independent of each other. This type of multitasking is applicable in OS level.

1. **Thread Based:** Executing several tasks simultaneously, where each task is separate independent part of same program. Each independent part is called a thread.

Thread based multitasking is best suitable at programmatic level.

**Advantage of Multitasking:** Weather it is process or thread based, the main objective of multitasking is to reduce the response time of system and to improve performance.

**Where we can use Multithreading concepts:**

To develop multimedia graphics

To develop animation

To develop Video Games

To develop Web servers and application servers etc.

When compared with old languages developing multithreaded applications in java is very easy because Java provides inbuild support for multithreading support by having rich API support (Thread, Runnable, Thread Group…).

Thread is a separate flow of execution. For every thread, a separate independent job is there.

**Defining a thread**

We can define a Thread in the following 2 ways:

1. By Extending Thread Class
2. By Implementing the Runnable Interface

**By Extending Thread Class**

Class MyThread extends Thread {

Public void run () {

----------------------------Job of thread------------

}

}

Every Java program contains only one normal thread i.e. Main Thread

To start the thread:

Class ThreadDemo {

Psvm (String[] args) {

**--In this point 1 thread--**

MyThread t = new MyThread ();🡪 **Thread Instantiation. Main thread creates a child thread.**

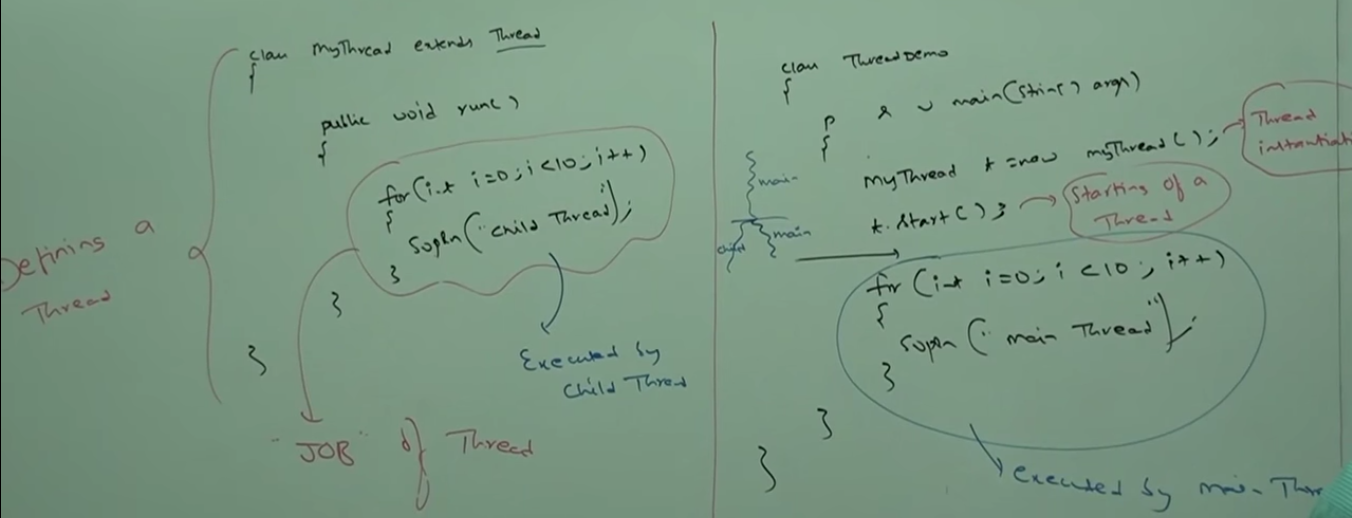
t.start();🡪**main thread starting of the child thread**

**--After t.start () In this point 2 thread---**

}

}

Main thread starts the child thread. Main method is executed by Main thread.



If there is a dependency do not go for Multithreading.

**Thread Scheduler:**

It is the part of JVM. It is responsible to schedule thread executions.

Algorithm followed by scheduler is varied from system to system.