**THREADS IN JAVA**

A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently. Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority. Thread is often referred to as **a lightweight process**. The process can be split down into so many threads. For example, in a browser, many tabs can be viewed as threads. MS Word uses many threads - formatting text from one thread, processing input from another thread, etc. Threads allow us to do things more quickly in Java.

There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

**Thread class:**

Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

Commonly used Constructors of Thread class:

Thread()

Thread(String name)

Thread(Runnable r)

Thread(Runnable r, String name)

**Runnable interface:**

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. Runnable interface have only one method named run().

**Examples**

1. Java Thread Example by extending Thread class

class Multi extends Thread{

public void run(){

System.out.println("thread is running...");

}

public static void main(String args[]){

Multi t1=new Multi();

t1.start();

}

}

Output: thread is running….

1. Java Thread Example by implementing Runnable interface

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);

Thread(Runnable r)

t1.start();

}

}

Output: thread is running….

Life cycle of a Thread

New: Whenever a new thread is created, it is always in the new state. For a thread in the new state, the code has not been run yet and thus has not begun its execution.

Active: When a thread invokes the start() method, it moves from the new state to the active state. The active state contains two states within it: one is runnable, and the other is running.

Blocked or Waiting: Whenever a thread is inactive for a span of time (not permanently) then, either the thread is in the blocked state or is in the waiting state.

Timed Waiting: A thread lies in a timed waiting state when it calls a method with a time-out parameter. A thread lies in this state until the timeout is completed or until a notification is received. For example, when a thread calls sleep or a conditional wait, it is moved to a timed waiting state.

**Terminated State:** A thread terminates because of either of the following reasons:

* Because it exits normally. This happens when the code of the thread has been entirely executed by the program.
* Because there occurred some unusual erroneous event, like segmentation fault or an unhandled exception.

Diagram

Description automatically generated

Single-threaded and Multi-threaded Processes

Single threaded processes contain the execution of instructions in a single sequence. In other words, one command is processes at a time.

The opposite of single threaded processes are multithreaded processes. These processes allow the execution of multiple parts of a program at the same time.

Advantages of Multithreaded Processes

Some of the advantages of multithreaded processes are given as follows

All the threads of a process share its resources such as memory, data, files etc. A single application can have different threads within the same address space using resource sharing.

It is more economical to use threads as they share the process resources. Comparatively, it is more expensive and time consuming to create processes as they require more memory and resources.

Program responsiveness allows a program to run even if part of it is blocked using multithreading. This can also be done if the process is performing a lengthy operation.

In a multiprocessor architecture, each thread can run on a different processor in parallel using multithreading. This increases concurrency of the system. This is in direct contrast to a single processor system, where only one process or thread can run on a processor at a time.

Disadvantages of Multithreaded Processes

Some of the disadvantages of multithreaded processes are given as follows −

Multithreaded processes are quite complicated. Coding for these can only be handled by expert programmers.

It is difficult to handle concurrency in multithreaded processes. This may lead to complications and future problems.

Identification and correction of errors is much more difficult in multithreaded processes as compared to single threaded processes.