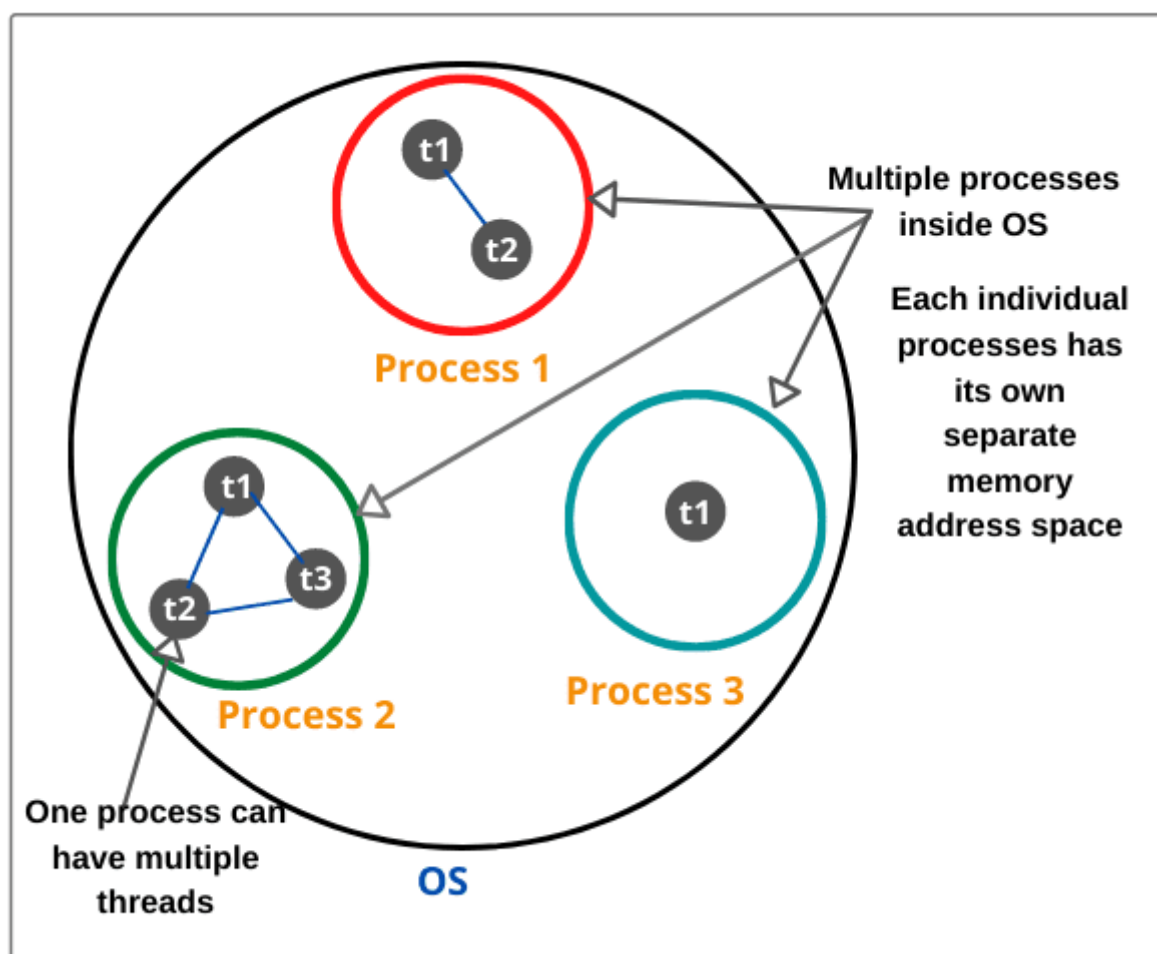


A process is a program that executes as a single thread. In other words, when an executable program is loaded into memory, it is called process.

Points to be Noted:

1. Every individual process has its own separate memory address space and can execute a different program.
2. Each process can have more than one thread.
3. Each process communicates through the operating system, files, and network.



Main Thread in Java

Every Java program has always at least one thread, even if you do not create any thread. This thread is called **main thread**.

The main thread is also called parent thread and the rest of threads that are generated from it are called child threads of the program.

Main thread is the last thread to be executed in a program. When main thread finishes the execution, the program terminates immediately.

Whenever Java program starts, main thread is created automatically.

This main thread is available in all programs. We can control the execution of the main thread by creating a Thread object and then using methods of Thread class.

To do so, we will have to create a Thread object by calling `currentThread()` method of class Thread.

```
public class MainThread
{
    public static void main(String[] args)
    {
        // Create a Thread object by calling currentThread() method of class Thread.
        Thread obj = Thread.currentThread();

        System.out.println("Current thread is " +obj);
        System.out.println("Name of current thread is " +obj.getName());

        obj.setName("New Thread"); // Changing name of main thread.
        System.out.println("Name of current thread after changing name is " +obj);
        System.out.println("Main thread existing");
    }
}
```

Use of Thread in Java

Threads can be used for multiple purposes. Some advantages of using threads are as follows:

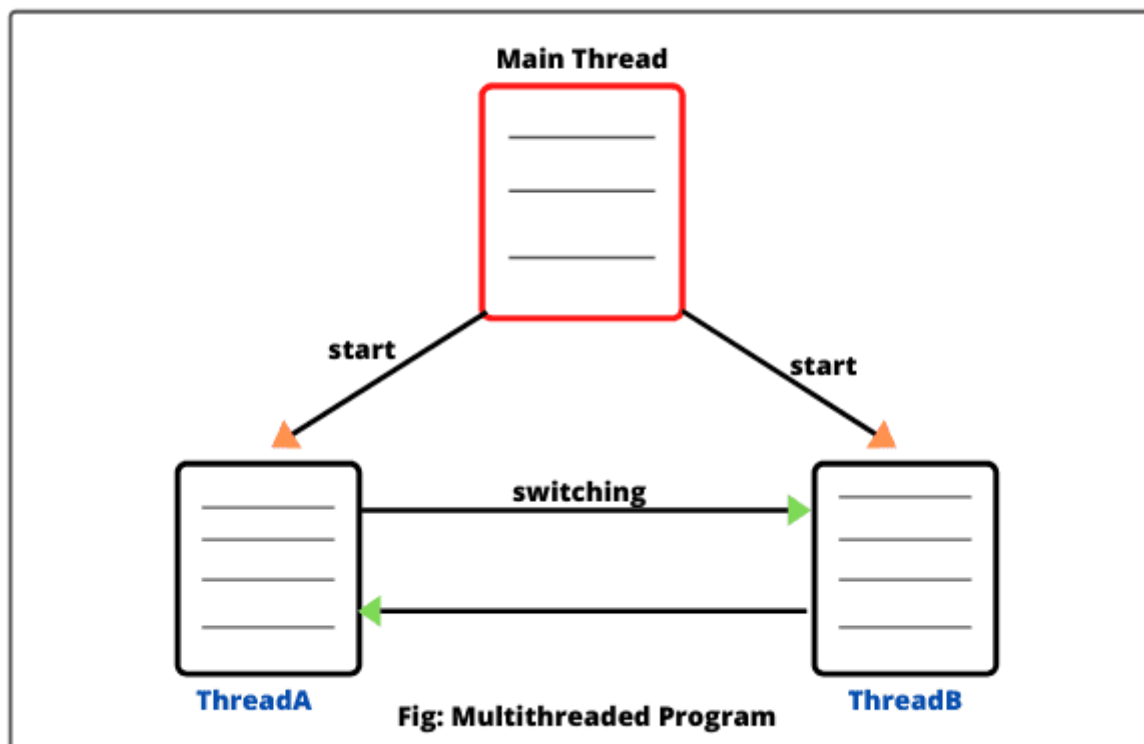
1. We mainly used threads in server-side programs where we need to handle multiple clients on network or internet simultaneously.
2. Another important use of threads is in creating games and animations.
3. Generally, threads can be used to perform more than one task simultaneously.

What is Multithreading in Java

Multithreading means multiple threads of execution concurrently. The process of executing

multiple threads simultaneously (concurrently) is called **multithreading in Java**.

In other words, multithreading is a technique or programming concept in which a program (process) is divided into two or more subprograms (subprocesses), each of which can perform different tasks simultaneously (at the same time and in parallel manner). Each subprogram of a program is called thread in Java.



Context Switch

When a program contains more than one thread, the CPU can switch between two threads to execute them at the same time. The switching between two threads is known as **context switch**

The switching from one thread to another thread occurs so fast that it appears to the users that all threads are being executed at the same time. But in reality, only one thread is executed at a time.

Time-Sharing

This technique is useful for those applications which need multiple tasks to be done simultaneously. In a single processor system, multiple threads share CPU time that is known as **time-sharing**.

The operating system is responsible for allocating and scheduling resources to them. Thus, multithreading improves the performance of CPU by maximum utilization and keeping the idle time of CPU to minimum.

Multitasking in Java

The process of executing one or more tasks concurrently or at the same time is called **multitasking**.

It is the ability of an operating system to execute multiple tasks at once. The main purpose of multitasking is to use the idle time of CPU.

Multitasking can be implemented in two ways:

1. Process-based multitasking (Multiprocessing)
2. Thread-based multitasking (Multithreading)

Process-based Multitasking (Multiprocessing)

The process of executing multiple programs or processes at the same time (concurrently) is called process-based multitasking or program-based multitasking. In process-based multitasking, several programs are executed at the same time by the microprocessor.

Therefore, it is also called multiprocessing in Java. It is a heavyweight. A process-based multitasking feature allows to execute two or more programs concurrently on the computer.

A good example is, running spreadsheet program while also working with word-processor.



































Each program (process) has its own address space in the memory. In other words, each program is allocated in a separate memory area.

Thread-based Multitasking (Multithreading)

A thread is a separate path of execution of code for each task within a program. In thread-based multitasking, a program uses multiple threads to perform one or more tasks at the same time by a processor.

That is, thread-based multitasking feature allows you to execute several parts of the same program at once. Each thread has a different path of

execution.

Multithreading vs Multiprocessing	
Multithreading	Multiprocessing
 Multithreading helps to create multiple threads within a single process to increase processing throughput.	 Basic Multiprocessing helps to increase processing throughput.
 Multiple threads of a single process are executed concurrently.	 Allows to create and execute multiple processes simultaneously.
 CPU has to switch among multiple threads to give execute them simultaneously.	 CPU switches among multiple processes and executes them according to job scheduling.
 Creation of the thread is based on parent and its execution space.	 Creation of the process happens on the resources.
 Threads belonging to the same process share the same memory and access the same data of the process.	 Multiprocessing assigns separate memory and resources for each of the processes.
 Multithreading system creates multiple threads of the same process/program.	 It allows executing multiple programs and tasks.
 Multithreading is not divided into subgraphs.	 It can be subdivided or segmented.
 Threads belonging to the same process share the same memory and resources within of the process.	 Multiprocessing allocates separate memory and resources for each process or program.
 An average amount of time is taken for job processing.	 Taken less time for executing the program.
 Execution of multiple thread is concurrent.	 Execution of program is concurrent.
 The process is divided into small subtasks and resources are shared among the subtasks.	 Availability of more than one processor per system, which has the ability to execute more than one operation in parallel.
 Processing unit can be more than one, but not necessary.	 System consists more than one processing unit.
 Various components of the same program are being executed at a time.	 Multiple processes can be executed at a time.
 Throughput is much higher.	 Throughput is average to less.
 Setup of the system is more sophisticated and less expensive.	 Setup of the system is expensive.
 Multiple threads are created for the same program, which increases the speed of the execution.	 Multiple CPUs are added to the one or multiple systems.
 Threads created from the single process is more rapid in terms of efficient execution of the process.	 Process created in a time consuming way.

Multithreading	Multiprocessing
Multithreading helps to create multiple threads inside a single process to increase computing throughput.	Basic Multiprocessing helps to increase computing throughput.
Multiple threads of a single process are executed concurrently.	Allows to create and execute multiple processes simultaneously.
CPU has to switch among multiple threads or can execute them simultaneously.	CPU switches among multiple processes simultaneously (according to job scheduling).
Creation of the thread is based on function and it is optimization specific.	Creation of the process depends on the resources.
Threads belonging to the same process share the same memory and resources as that of the process.	Multiprocessing assigned separate memory and resources for each of the processes.
Multithreading systems executes multiple threads of the same simultaneously.	It allows executing multiple programs and tasks.
Multithreading is not divided into categories.	It can be symmetric or asymmetric.
Threads belonging to the same process share the same memory and resources as that of the process.	Multiprocessing allocates separate memory and resources for each process or program.
An average amount of time is taken for job processing.	Takes less time for executing the job/programs.
Execution of thread/child thread is concurrent.	Execution is job/program is concurrent.
The process is divided into several different sub-processes called threads, which have their own way of execution.	Availability of more than one processor per system, which has the ability to execute more than one operation in parallel.
Processing unit can be more than one but not necessary.	System contains more than one processing unit.
Various components of the same process are being executed at a time.	Multiple process can be executed at a time.
Throughput is maximum/optimal.	Throughput is average to less.
Setup of the system is more optimized and less expensive.	Setup of the system is expensive.
Multiple threads are created for the same process, which increases the speed of the execution.	Multiple CPUs are added to the one or multiple systems.
Threads creation from the one-single process is more optimal in terms of efficient execution of the process.	Process creation is a time consuming task.