Multithreading in Java

Multithreading in Java is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

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

Advantages of Java Multithreading

1) It **doesn't block the user** because threads are independent and you can perform multiple operations at the same time.

2) You **can perform many operations together, so it saves time**.

3) Threads are **independent**, so it doesn't affect other threads if an exception occurs in a single thread.

Multitasking

Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved in two ways:

- Process-based Multitasking (Multiprocessing)
- Thread-based Multitasking (Multithreading)

1) Process-based Multitasking (Multiprocessing)

- Each process has an address in memory. In other words, each process allocates a separate memory area.
- A process is heavyweight.

- Cost of communication between the process is high.
- Switching from one process to another requires some time for saving and loading registers, memory maps, updating lists, etc.

2) Thread-based Multitasking (Multithreading)

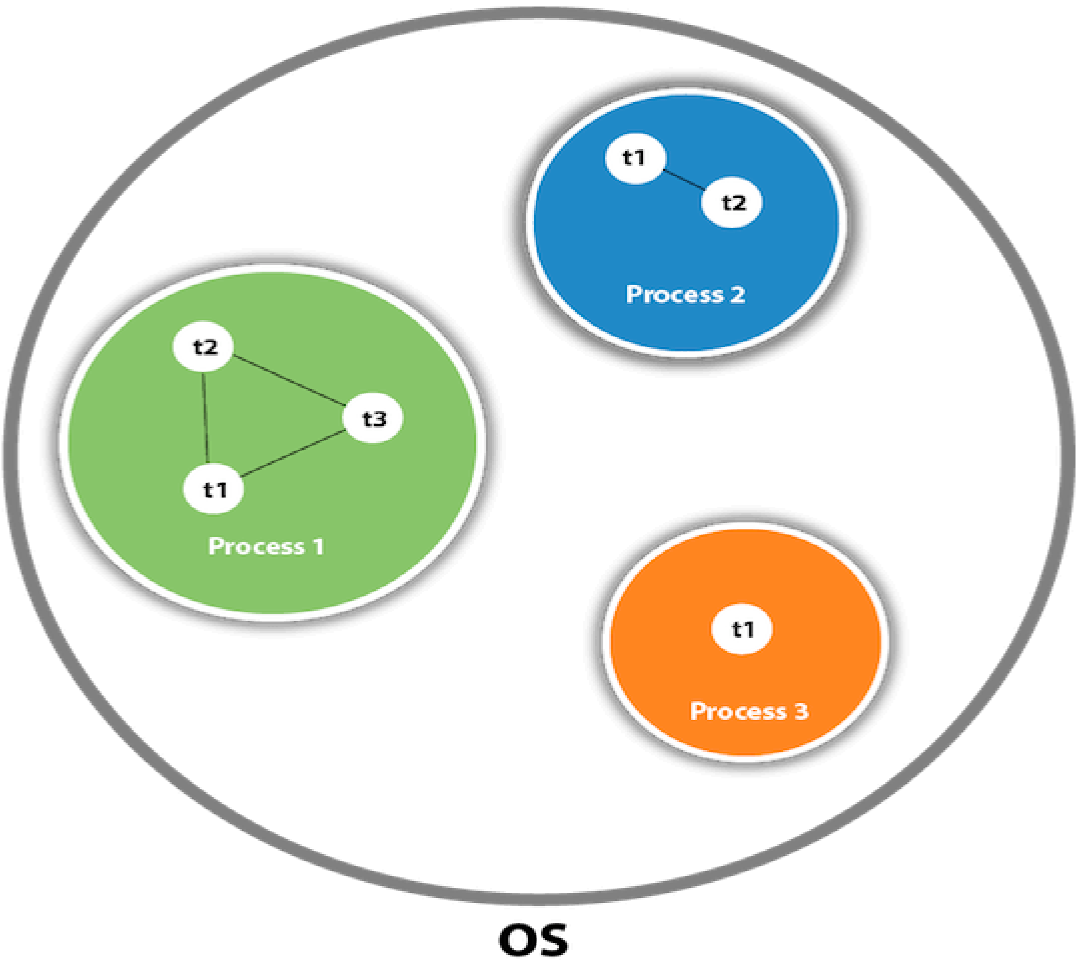
- Threads share the same address space.
- A thread is lightweight.
- Cost of communication between the thread is low.

Note: At least one process is required for each thread.

What is Thread in java

A thread is a lightweight subprocess, the smallest unit of processing. It is a separate path of execution.

Threads are independent. If there occurs exception in one thread, it doesn't affect other threads. It uses a shared memory area.



As shown in the above figure, a thread is executed inside the process. There is context-switching between the threads. There can be multiple processes inside the OS, and one process can have multiple threads.

Note: At a time one thread is executed only.

Java Thread class

Java provides **Thread class** to achieve thread programming. Thread class provides constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

Java Thread Methods

S.N.	Modifier and Type	Method	I
1)	void	start()	
2)	void	run()	
3)	static void	sleep()	
4)	static Thread	currentThread()	
5)	void	join()	
6)	int	getPriority()	

7)	void	setPriority()
8)	String	getName()
9)	void	setName()
10)	long	getId()
11)	boolean	isAlive()
12)	static void	yield()
13)	void	suspend()
14)	void	resume()
15)	void	stop()
16)	void	destroy()
17)	boolean	isDaemon()

18)	void	setDaemon()
19)	void	interrupt()
20)	boolean	isinterrupted()
21)	static boolean	interrupted()
22)	static int	activeCount()
23)	void	checkAccess()
24)	static boolean	holdLock()
25)	static void	dumpStack()

26)	StackTraceElement[]	getStackTrace()
27)	static int	enumerate()
28)	Thread.State	getState()
29)	ThreadGroup	getThreadGroup()
30)	String	toString()
31)	void	notify()
32)	void	notifyAll()

33)	void	setContextClassLoader()
34)	ClassLoader	getContextClassLoader()
35)	static Thread.UncaughtExceptionHandler	getDefaultUncaughtExceptionHandler()
36)	static void	setDefaultUncaughtExceptionHandler()

Do You Know

- How to perform two tasks by two threads?
- How to perform multithreading by anonymous class?
- What is the Thread Scheduler and what is the difference between preemptive scheduling and time slicing?
- What happens if we start a thread twice?
- What happens if we call the run() method instead of start() method?
- What is the purpose of join method?
- Why JVM terminates the daemon thread if no user threads are remaining?
- What is the shutdown hook?
- What is garbage collection?
- What is the purpose of finalize() method?
- What does the gc() method?
- What is synchronization and why use synchronization?

- What is the difference between synchronized method and synchronized block?
- What are the two ways to perform static synchronization?
- What is deadlock and when it can occur?
- What is interthread-communication or cooperation?

What will we learn in Multithreading

- Multithreading
- Life Cycle of a Thread
- Two ways to create a Thread
- How to perform multiple tasks by multiple threads
- Thread Scheduler
- Sleeping a thread
- Can we start a thread twice?
- What happens if we call the run() method instead of start() method?
- Joining a thread
- Naming a thread
- Priority of a thread
- Daemon Thread
- ShutdownHook
- Garbage collection
- Synchronization with synchronized method
- Synchronized block
- Static synchronization
- Deadlock
- Inter-thread communication