



SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai

Accredited by NAAC-UGC with 'A++' Grade (Cycle III) &

Accredited by NBA (B.E - CSE, EEE, ECE, Mech&B.Tech.IT)

COIMBATORE-641 035, TAMIL NADU



UNIT V

MULTITHREADING IN JAVA

Multithreading

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)

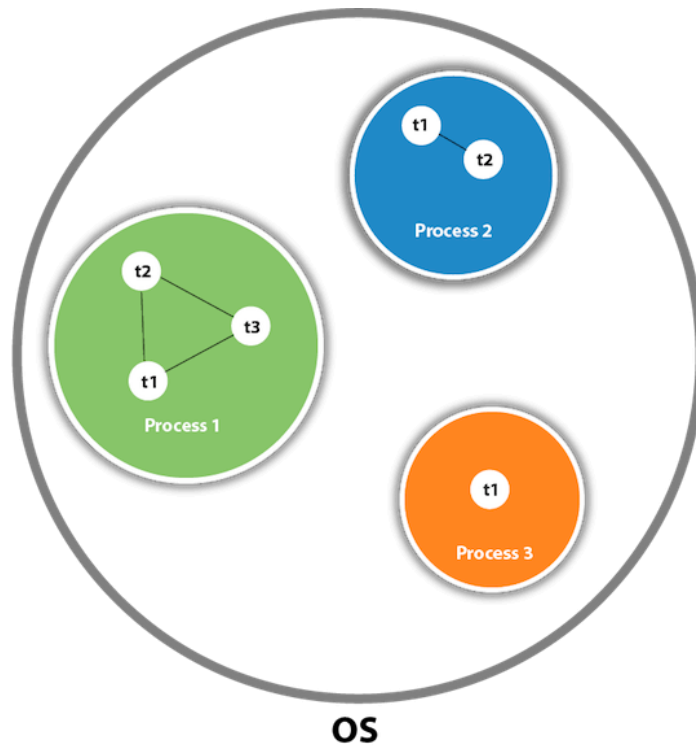
- Threads share the same address space.
- A thread is lightweight.
- Cost of communication between the thread is low.

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	Description
-------	-------------------	--------	-------------

1)	void	<u>start()</u>	It is used to start the execution of the thread.
2)	void	<u>run()</u>	It is used to do an action for a thread.
3)	static void	<u>sleep()</u>	It sleeps a thread for the specified amount of time.
4)	static Thread	<u>currentThread()</u>	It returns a reference to the currently executing thread object.
5)	void	<u>join()</u>	It waits for a thread to die.

6)	int	<u>getPriority()</u>	It returns the priority of the thread.
7)	void	<u>setPriority()</u>	It changes the priority of the thread.
8)	String	<u>getName()</u>	It returns the name of the thread.
9)	void	<u>setName()</u>	It changes the name of the thread.
10)	long	<u>getId()</u>	It returns the id of the thread.
11)	boolean	<u>isAlive()</u>	It tests if the thread is alive.
12)	static void	<u>yield()</u>	It causes the currently executing thread object to

			pause and allow other threads to execute temporarily.
13)	void	<u>suspend()</u>	It is used to suspend the thread.
14)	void	<u>resume()</u>	It is used to resume the suspended thread.
15)	void	<u>stop()</u>	It is used to stop the thread.
16)	void	<u>destroy()</u>	It is used to destroy the thread group and all of its subgroups.
17)	boolean	<u>isDaemon()</u>	It tests if the thread is a

			daemon thread.
18)	void	<u>setDaemon()</u>	It marks the thread as daemon or user thread.
19)	void	<u>interrupt()</u>	It interrupts the thread.
20)	boolean	<u>isinterrupted()</u>	It tests whether the thread has been interrupted .
21)	static boolean	<u>interrupted()</u>	It tests whether the current thread has been interrupted .
22)	static int	<u>activeCount()</u>	It returns the number of active

			threads in the current thread's thread group.
23)	void	<u>checkAccess()</u>	It determines if the currently running thread has permission to modify the thread.
24)	static boolean	<u>holdLock()</u>	It returns true if and only if the current thread holds the monitor lock on the specified object.
25)	static void	<u>dumpStack()</u>	It is used to print a stack trace

			of the current thread to the standard error stream.
26)	StackTraceElement[]	<u>getStackTrace()</u>	It returns an array of stack trace elements representing the stack dump of the thread.
27)	static int	<u>enumerate()</u>	It is used to copy every active thread's thread group and its subgroup into the specified array.

28)	Thread.State	<u>getState()</u>	It is used to return the state of the thread.
29)	ThreadGroup	<u>getThreadGroup()</u>	It is used to return the thread group to which this thread belongs
30)	String	<u>toString()</u>	It is used to return a string representation of this thread, including the thread's name, priority, and thread group.
31)	void	<u>notify()</u>	It is used to give the notification for only

			one thread which is waiting for a particular object.
32)	void	<u>notifyAll()</u>	It is used to give the notification to all waiting threads of a particular object.
33)	void	<u>setContextClassLoader()</u>	It sets the context ClassLoader for the Thread.
34)	ClassLoader	<u>getContextClassLoader()</u>	It returns the context ClassLoader for the thread.
35)	static Thread.UncaughtExceptionHandler	<u>getDefaultUncaughtExceptionHandler()</u>	It returns the default handler invoked

			when a thread abruptly terminates due to an uncaught exception.
36)	static void	<u>setDefaultUncaughtExceptionHandler()</u>	It sets the default handler invoked when a thread abruptly terminates due to an uncaught exception.