**Multithreading in Java**

|  |
| --- |
| Multithreading is a process of executing **multiple threads simultaneously**. |
| Thread is basically a **lightweight sub process**, **a smallest unit of processing**. **Multiprocessing** and **multithreading**, both are used to achieve **multitasking**. But we use multithreading than multiprocessing because **threads share a common memory area**. They **don't allocate separate memory area so save memory**, and **context-switching** between the threads takes **less time** than processes. |
| Multithreading is mostly used in **games**, **animation** etc. |

**Multitasking**

|  |
| --- |
| Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved by two ways:   * Process-based Multitasking(Multiprocessing) * Thread-based Multitasking(Multithreading) |

**1)Process-based Multitasking (Multiprocessing)**

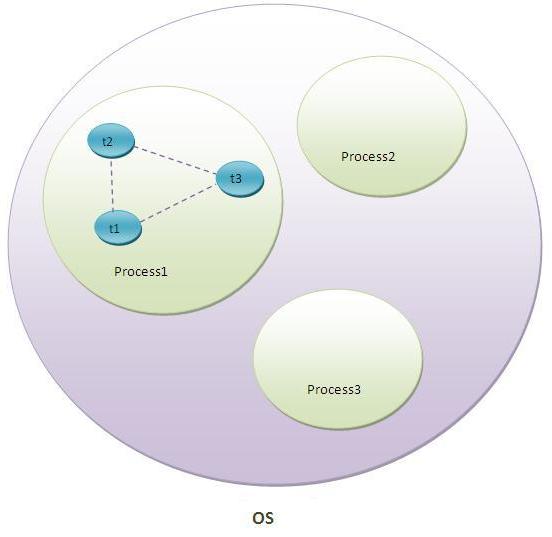
|  |
| --- |
| * Each process have its own address in memory i.e. each process allocates separate memory area. * Process is heavyweight. * Cost of communication between the process is high. * Switching from one process to another require some time for saving and loading registers, memory maps, updating lists etc. |

**2)Thread-based Multitasking (Multithreading)**

|  |
| --- |
| * Threads share the same address space. * Thread is lightweight. * Cost of communication between the thread is low. * **Note:** At least one process is required for each thread. |

**What is Thread?**

|  |
| --- |
| A thread is a lightweight sub process, a smallest unit of processing. It is a separate path of execution. It shares the memory area of process. |



|  |
| --- |
| As shown in the above figure, 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 only one thread is executed.** |