What is tasking, multitasking, What is OS, Internal Working behavior of OS, what is process, thread, multiprocessing and multithreading.

- K.L.MADHAVI

Multitasking

Multitasking refers to handling or performing multiple tasks at the same time. Whereas, Multitasking in Java is like efficiently managing multiple cooking tasks simultaneously in the kitchen. Just as you need to handle different dishes simultaneously to keep the restaurant running smoothly, Java programs can handle multiple tasks concurrently.

Now multitasking can be done in two ways:

1. Process-based multitasking : Process-based multitasking involves running multiple independent processes simultaneously. Each process runs in its own memory space, operates independently, and has its own system resources.
2. Thread-based multitasking : Thread-based multitasking involves dividing the program's execution into multiple threads. Threads share the same memory space and resources, allowing for efficient communication and coordination.

What is OS

An operating system acts as an intermediary between the user of a computer and computer hardware. In short its an interface between computer hardware and user. The purpose of an operating system is to provide an environment in which a user can execute programs conveniently and efficiently.

Characteristics of Operating Systems:

- **Device Management:** The operating system keeps track of all the devices. So, it is also called the Input/Output controller that decides which process gets the device, when, and for how much time.

- **File Management:** It allocates and de-allocates the resources and also decides who gets the resource.

- **Job Accounting:** It keeps track of time and resources used by various jobs or users.

Internal Working behavior of OS

The **internal working behavior of an Operating System (OS)** revolves around managing and coordinating computer hardware and software resources to provide a seamless and efficient environment for users and applications.It includes following aspects:

1. Process Management

- **Purpose:** Manages the execution of processes (programs in execution).
- **Key Tasks:**

Process Scheduling: Uses algorithms like Round Robin, Priority Scheduling, or Multilevel Queue to allocate CPU time.

Multitasking: Enables multiple processes to run simultaneously through time-sharing.

Continue…

2. Memory Management

- **Purpose:** Allocates and optimizes the use of primary memory (RAM).
- **Key Tasks:**

  Memory Allocation: Dynamically allocates and deallocates memory to processes.

  Protection: Ensures processes don’t access each other’s memory.

3. File System Management

- **Purpose:** Organizes and stores data on storage devices (HDD, SSD).
- **Key Tasks:**

  File Organization: Maintains a structured directory hierarchy.

  Storage Allocation: Allocates disk blocks to files and manages free space.

What is Process,Multiprocessing

Process:

- A process is an independent program execution unit that has its own memory space and resources.
- Each process runs in its own virtual machine (JVM) instance.
- Processes are heavier-weight than threads and have higher overhead.
- Communication between processes is typically done through inter-process communication (IPC) mechanisms like pipes, sockets, or shared memory.

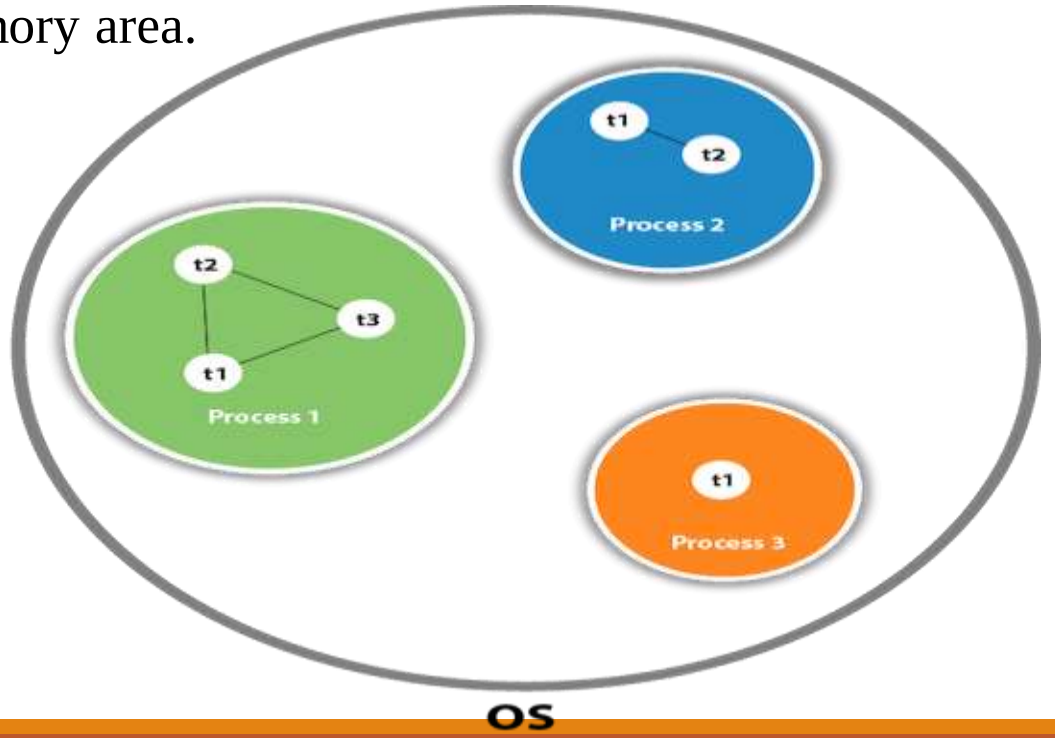
Multiprocessing:

- Multiprocessing is the execution of multiple processes concurrently, typically on different CPU cores.
- It allows you to take full advantage of the hardware capabilities of multi-core systems, improving performance and responsiveness.

What is Thread,MultiThreading

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.



Continue…

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

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.

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.