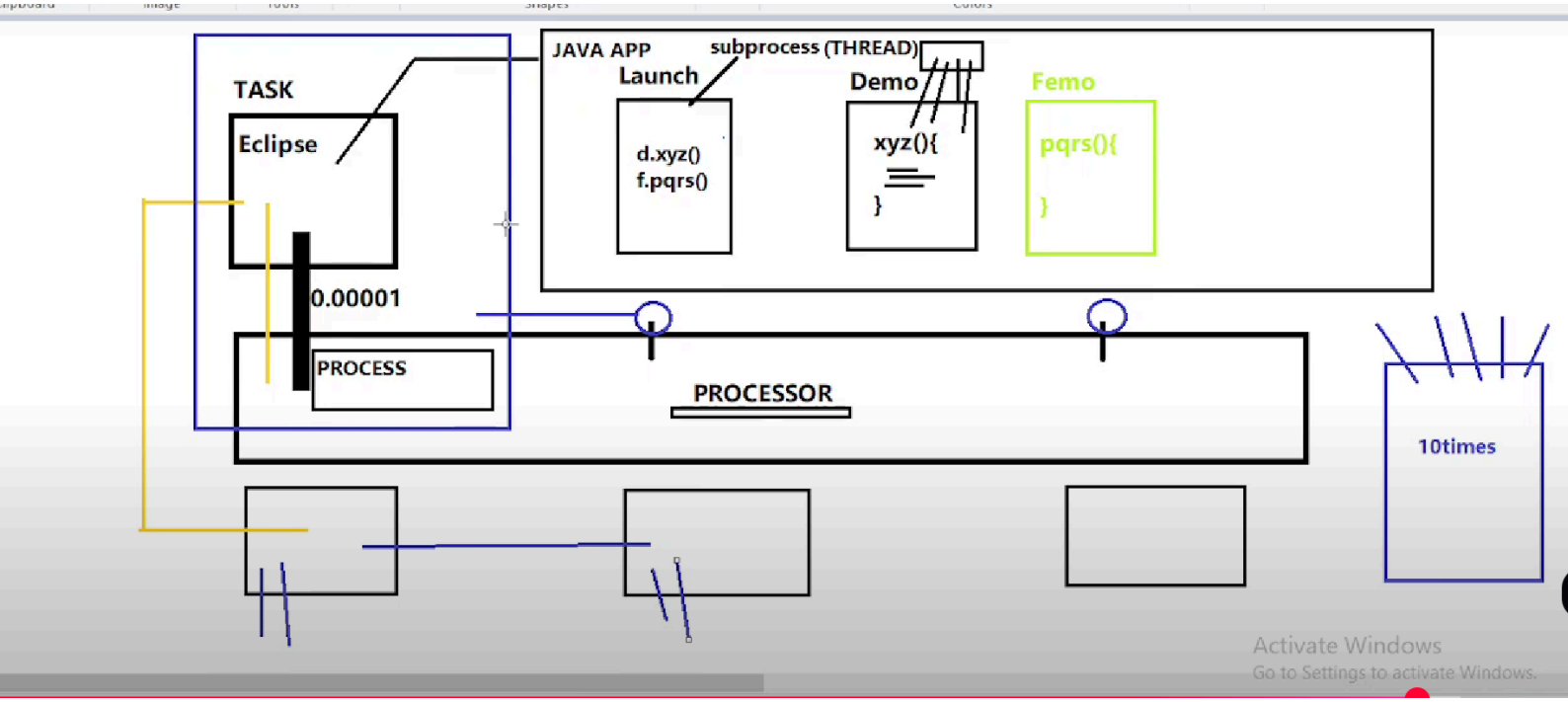
Lec 30 Multi-Threading Part 1 :

- OS is the soul of computer and hardware is the body of computer. OS manages the processing of multiple processes.
- OS manages multiple processes like Eclipse/ YouTube/Spotify. It performs Context switching / Time slicing (0.0000001s) between these applications and represents multitasking.
- Each process has its own memory. If context switches from Eclipse to YouTube, then Eclipse's data gets stored (saved till here) and it is again in motion when context is given to it.
- Process is heavy weight.

- If we talk about Eclipse only, then it is a process, but inside Eclipse, there can be multiple classes.
- Java code is Single-threaded by default.
- Whenever exception occurs in main class it referred as "Exception in thread main....." .
- The classes or code in Eclipse is single threaded and are called Sub-processes, which means that sequential flow of execution of programs will occur (one by one).
- Suppose A and B both are watching YT (MrBeast), then will they be provided separate video to them

(copied to different locations) or from same location?

- Of course same location as one video gets 10M views on average, the location from where video dispatches is made multithreaded.
- Thread shares same memory. It is light weight. Context switching is fast.
- 2 threads are independent (if i pause the same video on YT then it would not be paused for my mom watching the same video).
- Threads are created for multi-tasking.
- Example : Suppose there is a class which prints “hello” 10 times, if we create 10 threads of the class then 10 people will be able to simultaneously access that class.



This lesson explains the concepts of multi-processing and multi-threading in software, focusing on how computers handle many tasks at once. Multi-processing allows multiple programs to run at the same time using separate memory, while multi-threading lets different parts of a single program run together efficiently by sharing memory. Multi-threading saves CPU time and is key for building responsive, multitasking applications.

Introduction to Multi-threading

Multi-threading is introduced as a crucial technique for enabling multitasking in real-world applications, especially web servers and multi-user platforms.

What is Multi-threading?

Multi-threading is defined as a way to perform multiple tasks at once within an application, allowing actions like downloading and watching videos simultaneously.

Single-threaded vs. Multi-threaded Applications

Single-threaded applications execute tasks in sequence; one task must finish before the next begins, which can waste CPU time and make apps less responsive.

Multi-tasking in Software: Multi-threading vs. Multi-processing

Multi-tasking can be achieved by multi-threading (multiple threads within one process) or multi-processing (multiple processes, each with its own memory and resources).

Understanding Multi-processing

Multi-processing runs different programs (processes) at the same time, with each process having its own memory, resources, and port. The operating system manages switching between these processes, which is

called context switching and is resource-intensive (heavyweight).

Key Concepts: Memory Sharing, Context Switching, and Communication

In multi-processing, each process has its own memory and resources, making context switching and inter-process communication slow and complex.

Problems with Heavy Context Switching

If too many processes run on a weak processor, switching between them becomes slow, leading to system lag that users can notice.

Multi-processing: Memory and Communication

Each process in multi-processing has its own memory space, so sharing data or communicating between processes is difficult and slow (heavyweight IPC).

Defining a Process

A process is an instance of a program being executed by the operating system, and it is considered

"heavyweight" due to its resource needs.

The Role of Processor Cores

Multi-core processors can run multiple processes truly in parallel, while single-core processors use fast context switching to simulate multitasking.

Who Manages Process Switching?

The operating system, not the processor itself, manages which process runs and when, using scheduling algorithms.

The Problem with Sequential Code Execution

In single-threaded code, if one part waits (like for user input), the rest of the program cannot proceed, wasting CPU time.

Multi-threading: Definition and Benefits

Multi-threading is a technique to achieve multitasking within a single application, allowing different parts of code to run together and making better use of CPU time.

Multi-threading vs. Multi-processing in Applications

Multi-processing is used for running multiple applications, while multi-threading is for running multiple tasks within a single application at the same time.

Understanding Threads

A thread is a lightweight sub-process within a process, responsible for running parts of code. By default, Java programs start with a single main thread.

Single-threaded vs. Multi-threaded Java Programs

By default, Java programs are single-threaded, meaning only one thread (main thread) runs unless additional threads are created by the programmer.

Creating Multiple Threads

Developers can create multiple threads to run different parts of code simultaneously, making programs more efficient and responsive.

Thread Characteristics: Memory Sharing and Switching

Threads within the same process share memory and resources, making context switching and communication between them fast and lightweight.

Summary: Multi-processing vs. Multi-threading

Multi-processing uses separate memory for each process (heavyweight), while multi-threading uses shared memory (lightweight), making thread switching faster and more efficient.

Threads and User Experience

Multiple threads can be created for a class so that many users can access the same code independently and simultaneously, improving scalability and responsiveness.

Thread Scheduling and Management

The Java Virtual Machine (JVM) and its thread scheduler manage how threads are started, stopped, and prioritized, allowing for flexible and efficient multitasking within applications.

Thread Independence and Error Handling

Threads are independent, so if one thread encounters an error, others can continue running, which is important for reliability in multi-user or multitasking environments.

Next Steps: Advanced Thread Concepts

More advanced topics like thread synchronization, joining, sleeping, and deadlock avoidance will be covered later, as these are key for managing complex multi-threaded applications.