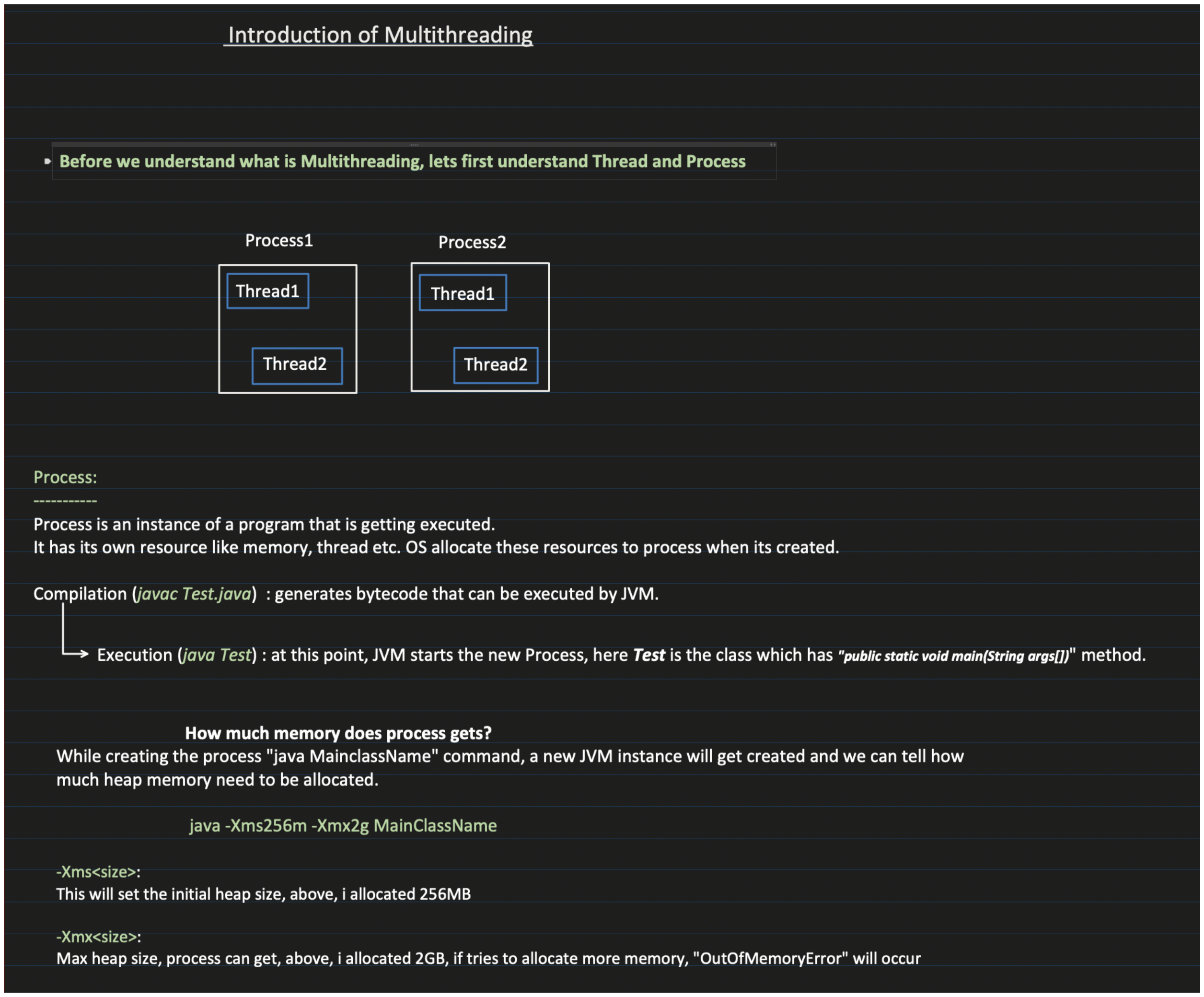
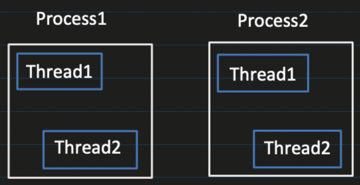
Java Multithreading Part - 1



Introduction of Multithreading

- **Before we understand what is Multithreading, lets first understand Thread and Process**



Process:

Process is an instance of a program that is getting executed.
It has its own resource like memory, thread etc. OS allocate these resources to process when its created.

Compilation (*javac Test.java*) : generates bytecode that can be executed by JVM.

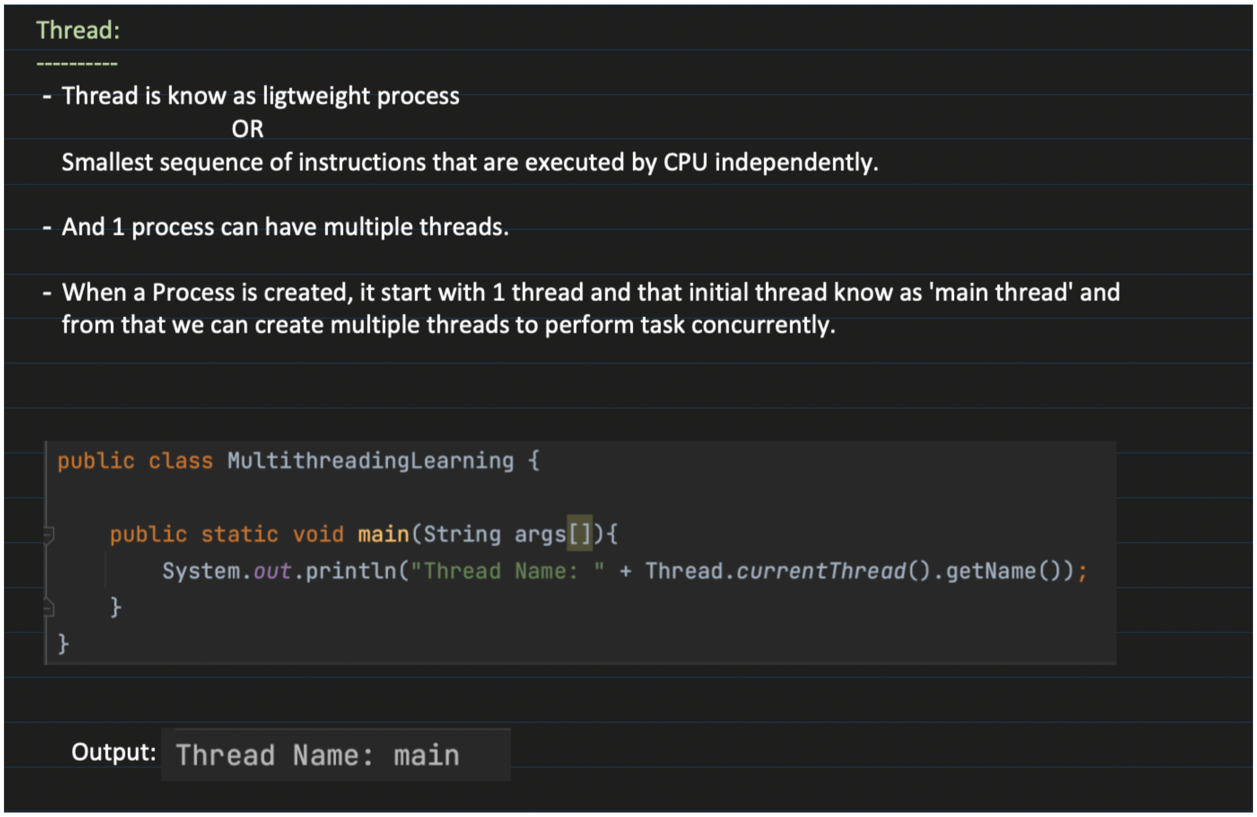
→ Execution (*java Test*) : at this point, JVM starts the new Process, here ***Test*** is the class which has ***"public static void main(String args[])"*** method.

How much memory does process gets?
While creating the process "java MainclassName" command, a new JVM instance will get created and we can tell how much heap memory need to be allocated.

```
java -Xms256m -Xmx2g MainClassName
```

-Xms<size>:
This will set the initial heap size, above, i allocated 256MB

-Xmx<size>:
Max heap size, process can get, above, i allocated 2GB, if tries to allocate more memory, "OutOfMemoryError" will occur



Thread:

- Thread is know as ligtweight process

  OR

  Smallest sequence of instructions that are executed by CPU independently.

- And 1 process can have multiple threads.

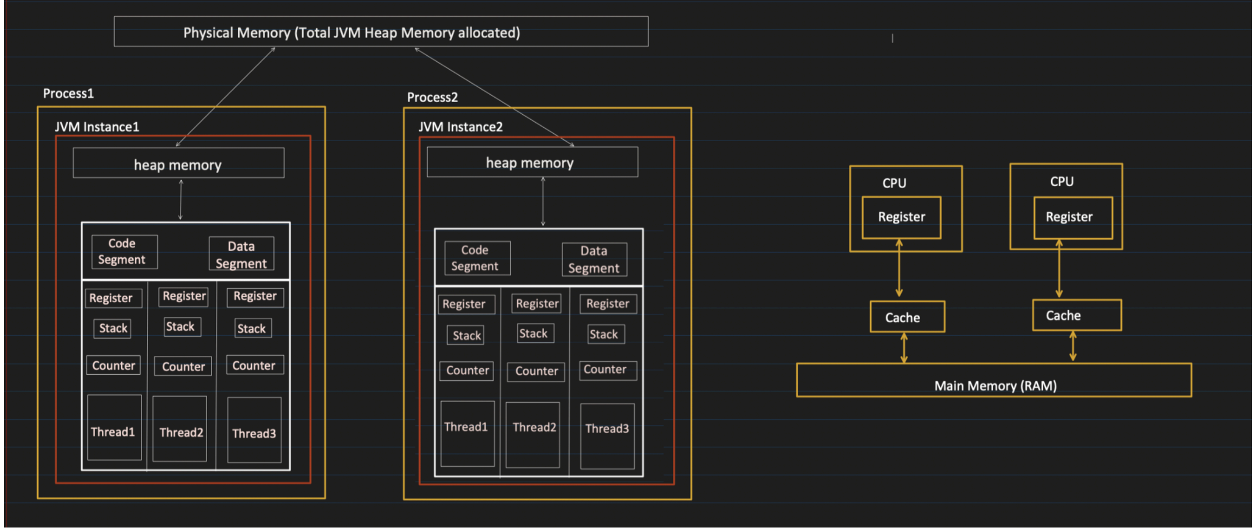
- When a Process is created, it start with 1 thread and that initial thread know as 'main thread' and from that we can create multiple threads to perform task concurrently.

```java
public class MultithreadingLearning {

    public static void main(String args[]){
        System.out.println("Thread Name: " + Thread.currentThread().getName());
    }
}
```

Output:
```
Thread Name: main
```

Let's understand little bit more about Process and Threads:



Code Segment:

- Contains the compiled **BYTECODE** *(i.e machine code)* of the Java Program.
- Its read only.
- All threads within the same process, share the same code segment.

Data Segment:

- Contains the GLOBAL and STATIC variables.
- All threads within the same process, share the same data segment.
- Threads can read and modify the same data.
- Synchronization is required between multiple threads.

Heap :

- Objects created at runtime using "new" keyword are allocated in the heap.

- Heap is shared among all the threads of the same process. (but NOT WITHIN PROCESS)
  (let say in Process1, X8000 heap memory pointing to some location in physical memory, same X8000 heap memory point to differet location for Process2)

- Threads can read and modiy the heap data.

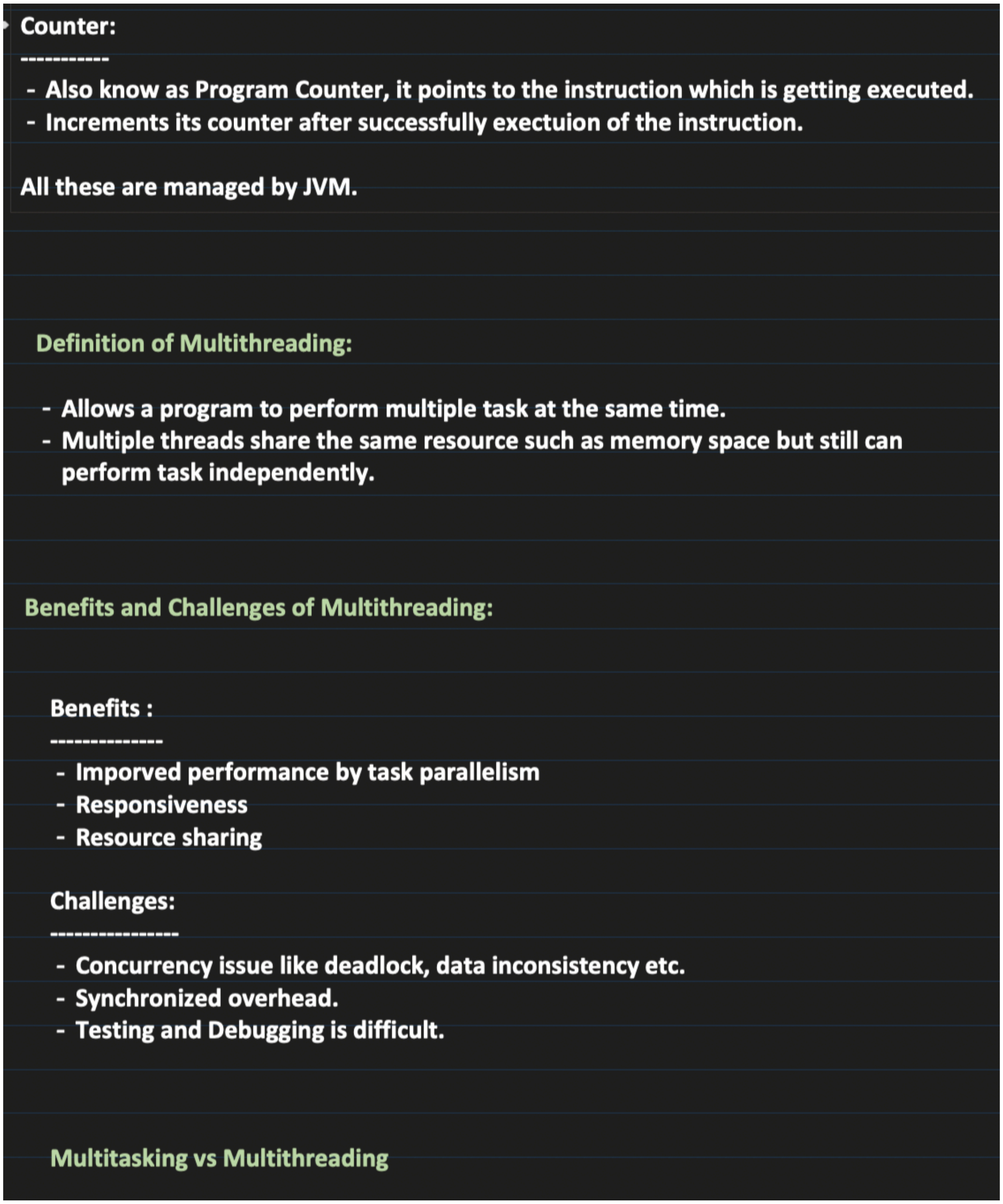
- Synchronization is required between multiple threads.

Stack:

- **Each thread has its own STACK.**
- **It manages, method calls, local variables.**

Register:

- **When JIT (Just-in time) compiles converts the Bytecode into native machine code, its uses register to optimized the generated machine code.**
- **Also helps in context switching.**
- **Each thread has its own Register.**



Counter:

- **Also know as Program Counter, it points to the instruction which is getting executed.**
- **Increments its counter after successfully exectuion of the instruction.**

All these are managed by JVM.

Definition of Multithreading:

- **Allows a program to perform multiple task at the same time.**
- **Multiple threads share the same resource such as memory space but still can perform task independently.**

Benefits and Challenges of Multithreading:

Benefits :

- **Imporved performance by task parallelism**
- **Responsiveness**
- **Resource sharing**

Challenges:

- **Concurrency issue like deadlock, data inconsistency etc.**
- **Synchronized overhead.**
- **Testing and Debugging is difficult.**

Multitasking vs Multithreading