

Let's break down the difference between a **Java process** and **Java threads** clearly:

1. Java Process

- A **process** is an instance of a running Java program.
- Each process has its **own memory space** (heap, stack, method area - its own instance of JVM).
- Processes are **independent**; one process crashing does not affect others.
- **Communication** between processes requires inter-process communication (IPC) like files, sockets, or shared memory.
- Example: Running `java MyApp` in two separate terminal windows creates **two separate processes. Each has its own instance of JVM.**

Characteristics:

| Feature | Java Process |
|---------------|--|
| Memory | Separate memory for each process |
| Isolation | High (processes don't share memory) |
| Communication | IPC needed |
| Overhead | Higher (each JVM instance consumes resources) |
| Crash impact | One process crash usually does not affect others |

2. Java Threads

- A **thread** is the smallest unit of execution in Java.
- All threads in a process **share the same memory space** (heap), but each thread has its own **stack**.

- Threads are **dependent**: if one thread crashes due to an exception and is unhandled, it may terminate the whole process.
- Communication between threads is **easy**, since they share memory.
- Example: Using `new Thread(() -> { ... }).start();` in a Java program creates a thread **inside the running process**.

Characteristics:

| Feature | Java Thread |
|---------------|--|
| Memory | Shares process memory; each has its own stack |
| Isolation | Low (threads share memory, so synchronization is needed) |
| Communication | Simple, via shared variables |
| Overhead | Low (lighter than full processes) |
| Crash impact | Can affect the whole process if exception is unhandled |

Key Differences

| Aspect | Process | Thread |
|-----------------|------------------------------|----------------------|
| Memory | Separate | Shared |
| Creation cost | High | Low |
| Communication | Difficult (IPC) | Easy (shared memory) |
| Fault tolerance | Independent | Dependent |
| JVM | Each process has its own JVM | Shares the same JVM |

💡 Analogy:

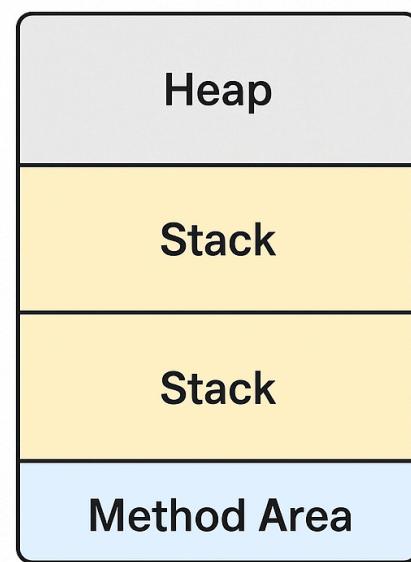
- Process = **house**
- Threads = **people in the house**

- They share the same house (memory) but have their own rooms (stack).
 - People can interact easily, but if the house catches fire, everyone is affected.
-

Java Process vs Java Threads



Java Process



Java Threads