**Java Memory Model — Important Points**

**1. Stack Memory**

* Stores **method calls**, **local variables**, and **reference variables**.
* **Continuous memory allocation** (fixed-size blocks).
* **Faster access** compared to heap.
* Automatically cleared when the method finishes.
* Variables stored here have **short life** (only till the method runs).

**Example:**

void method() {

int a = 5; // stored in stack

}

When method() finishes, a is removed from memory.

**2. Heap Memory**

* Stores **objects** and **arrays**.
* **Non-continuous memory allocation** (managed by JVM).
* Slower than stack because of dynamic allocation.
* Cleared by **Garbage Collector** (automatic cleanup).
* Objects can be accessed from anywhere if you have their reference.

**Example:**

String s = new String("Hello");

// 's' (reference) → stack

// actual object "Hello" → heap

**3. Arrays in Memory**

* **Stored in heap** (because arrays are objects in Java).
* The **reference** to the array is stored in the stack.
* Memory for an array is **continuous** in the heap (helps fast access by index).

**Example:**

int[] arr = {1, 2, 3};

// arr → stack (reference)

// {1, 2, 3} → heap (continuous memory)

**4. Objects in Memory**

* **Created in heap**.
* Reference is stored in the stack.
* Multiple references can point to the same object.

**Example:**

Person p1 = new Person(); // object in heap

Person p2 = p1; // p2 points to same object

✅ **Key Differences: Stack vs Heap**

| **Feature** | **Stack** | **Heap** |
| --- | --- | --- |
| **Allocation** | Continuous | Non-continuous |
| **Speed** | Fast | Slower |
| **Stores** | Method calls, references, local vars | Objects, arrays |
| **Cleanup** | Method end | Garbage Collector |
| **Lifetime** | Short | Until no reference exists |