Data Structures – Foundation

Motivation: How can you be a better coder, if you don't know how good your code is? 😊

Memory Areas: The below diagram describes the different memory areas in **Java**:

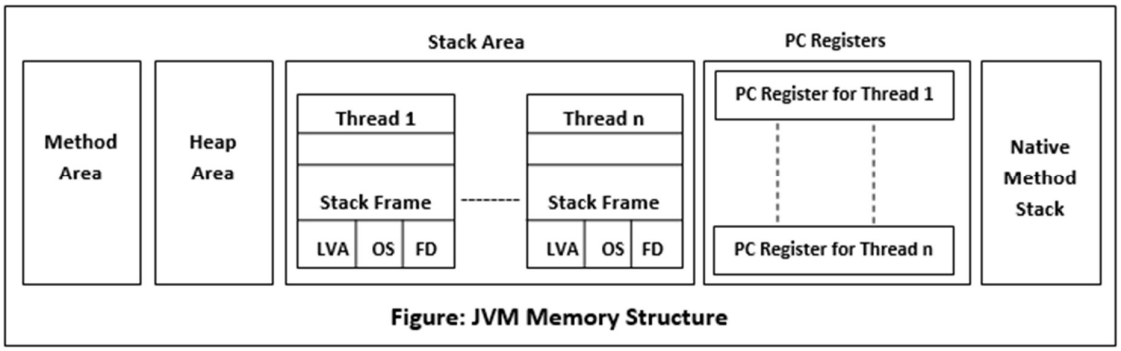


Figure: JVM Memory Structure

Important points to remember:

Method Area: Global variables are stored in Method Area.

Heap Area: Objects are stored in Heap.

Stack Area: Local variables are stored in Stack.

Native method stack: Stack stores the data of the methods in non-java language. (also called as C-Stack)

PC Register:

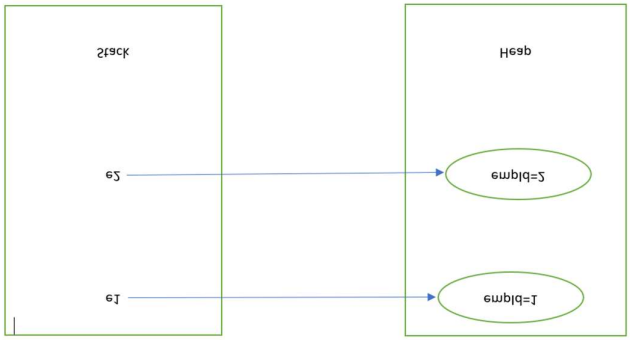
- For non – native methods: JVM thread has a program counter (PC) associated with it. PC stores the available JVM instructions.
- For Native methods: PC value is undefined. PC Register stores the return address of the native pointer.

Simple code:

```
public class Employee {
	int empId;
}

public class Test {
	static String Company;
	public static void main(String[] args) {
		Employee e1 = new Employee();
		Employee e2 = new Employee();
		}
}
```

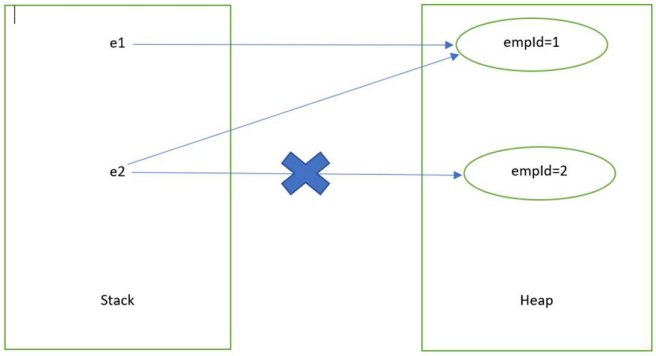
Mercy Technologies



Tweaking the above code to see the change in the memory organization:

```
public class Test {
	static String Company;

	public static void main(String[] args) {
		Employee e1 = new Employee();
		Employee e2 = new Employee();
		e1.empId = 10;
		e2 = e1;
	}
}
```



Important points to remember:

1. For every method call, there will be an activation records that gets created.
2. Activation record gets destroyed, once the method call gets completed
3. The stack gets destroyed when the associated thread gets get destroyed.

Sample code:

```
public class MethodCallExample {
	public static void main(String[] args) {
		m1();
	}
	private static void m1() {
		int a;
		m2();
	}
	private static void m2() {
		int b;
		m3();
		m4();
	}
	private static void m3() {
		int c;
	}
	private static void m4() {
		int d;
	}
}
```