### **Stack:**

1. **What is Stack?**
   * The stack is a region of memory that operates in a Last-In-First-Out (LIFO) manner.
   * It is used to store primitive values and references to objects in Java.
2. **What is Stored in the Stack?**
   * Local variables: Variables declared within a method, including method parameters.
   * References to objects: When you declare an object reference variable (e.g., MyObject obj;), the reference to the actual object is stored in the stack.
   * Primitive values: Variables holding primitive data types like int, boolean, char, etc., are stored directly in the stack.
3. **Characteristics:**
   * Memory allocation and deallocation for stack variables are handled automatically by the Java Virtual Machine (JVM).
   * The stack size is generally much smaller compared to the heap.
   * Accessing variables on the stack is faster because it follows a predictable and efficient allocation/deallocation mechanism.
4. **Lifetime:**
   * Variables in the stack have a limited lifetime corresponding to the scope of the method they are declared in.
   * They are automatically removed (popped) from the stack when the method returns or completes.

**Example:**  
public class StackExample {

public static void main(String[] args) {

int x = 10; // x is stored in the stack

String name = "John"; // reference to "John" string object is stored in the stack

// Method call with parameter

calculateSum(x, 20); // x and 20 are stored in the stack during the method call

}

public static void calculateSum(int a, int b) {

int sum = a + b; // a, b, and sum are stored in the stack

System.out.println("Sum is: " + sum);

}

}

### 

### **Heap:**

1. **What is the Heap?**
   * The heap is a region of memory that stores objects created dynamically using the new keyword in Java.
   * It is a large pool of memory where Java objects live.
2. **What is Stored in the Heap?**
   * Objects: When you create an object using new, the object itself (including instance variables) is stored in the heap.
   * Arrays: Arrays in Java are also objects and are stored in the heap.
   * Class level variables (static variables): Variables declared as static belong to the class and are stored in the heap.
3. **Characteristics:**
   * Memory allocation and deallocation for objects in the heap are handled by the JVM's garbage collector.
   * The heap size is typically larger than the stack and can dynamically expand if needed (up to certain limits set by the JVM).
   * Accessing objects in the heap is slower compared to stack variables because it involves dereferencing and more complex memory access patterns.
4. **Lifetime:**
   * Objects in the heap exist beyond the scope of the method or block they were created in.
   * They remain in memory until they are no longer referenced (garbage collected) or until the program terminates.

**Example:**  
public class HeapExample {

public static void main(String[] args) {

// Creating objects in the heap

Student student1 = new Student("Alice", 20);

Student student2 = new Student("Bob", 22);

// Arrays are also stored in the heap

int[] numbers = new int[5];

}

}

class Student {

String name;

int age;

public Student(String name, int age) {

this.name = name;

this.age = age;

}

}

### **Summary:**

* **Stack**:
  + Used for local variables and method calls.
  + LIFO (Last-In-First-Out) structure.
  + Automatic memory management by the JVM.
  + Faster access.
* **Heap**:
  + Used for dynamic memory allocation (objects, arrays).
  + Managed by the garbage collector.
  + Slower access compared to stack variables.
  + Larger and more flexible in size.

Prepared by : Rishi Patel