**Class and object**

**Class:**

**A class in Java is like a blueprint or template for creating objects. It defines properties (attributes) and behaviors (methods) that the objects created from the class will have.**

**Example: Think of a class like the blueprint for a table. It specifies the dimensions, material, and design. However, the blueprint itself is not a table; it's just a plan for creating tables.**

**In programming terms, you are the class that gives the blueprint (instructions) to a carpenter (the program). The carpenter follows these instructions to create a real table (object).**

**Object:**

**An object is an instance of a class. It is a real-world entity created based on the blueprint (class). In Java, everything revolves around objects. Once the JVM creates an object using the class as a blueprint, the object will have the properties and behaviors defined by the class.**

**Example: If the blueprint is the class, then the actual table made by the carpenter is the object. Each table (object) may have its own state (specific size, color, etc.) but follows the general structure defined by the blueprint (class).**

**Stack And Heap Memory**

**1. Stack Memory:**

* **Stack memory is used for static memory allocation and storing local variables.**
* **It holds primitive data types (int, float, char, etc.) and references to objects (but not the objects themselves).**
* **Each time a method is called, a new block (or frame) is created on the stack for that method. This block contains all the method's local variables.**
* **When the method execution completes, its block is removed from the stack (last in, first out - LIFO structure).**

**Example of Stack Memory:**

**public class StackExample {**

**public static void main(String[] args) {**

**int a = 10; // stored in Stack (primitive)**

**int b = 20; // stored in Stack (primitive)**

**add(a, b); // method call creates a new stack frame for the method 'add'**

**}**

**public static void add(int x, int y) {**

**int sum = x + y; // x, y, and sum are stored in the stack frame for 'add'**

**System.out.println(sum);**

**}**

**}**

**Explanation:**

* **Variables a and b are stored in the main() method's stack frame.**
* **When the add() method is called, x, y, and sum are stored in the add() method's stack frame.**
* **Once the add() method finishes execution, its stack frame is destroyed, and x, y, and sum are removed.**

**2. Heap Memory:**

* **Heap memory is used for dynamic memory allocation, and it stores all objects created in Java using the new keyword.**
* **It holds the actual objects, and the references to these objects are stored in the stack.**
* **Heap memory is managed by the Java Garbage Collector, which reclaims memory used by objects that are no longer referenced.**

**Example of Heap Memory:**

**public class HeapExample {**

**public static void main(String[] args) {**

**// Person object is created in the heap, reference stored in the stack**

**Person person1 = new Person("John", 25);**

**Person person2 = new Person("Alice", 30);**

**}**

**}**

**class Person {**

**String name; // stored in heap**

**int age; // stored in heap**

**Person(String name, int age) {**

**this.name = name;**

**this.age = age;**

**}**

**}**

**Explanation:**

* **The two Person objects (person1 and person2) are created in the heap memory.**
* **However, their references (or pointers) are stored in the stack memory in the main() method's stack frame.**
* **The name and age fields of each object are also stored in the heap memory.**