# CLASS

**You:** A class is a fundamental building block in Object-Oriented Programming (OOP). It's essentially a user-defined data type that acts as a blueprint for creating objects. A class encapsulates data members (attributes) and member functions (methods) that operate on these data members. By defining a class, you can create objects that share common properties and behaviors specified in the class.

**Example:** Consider a class Car:

* **Data Members:** These could include attributes like make, model, year, and color.
* **Member Functions:** These could be methods like start(), stop(), and accelerate().

Using this Car class, you can create multiple car objects, each with specific values for make, model, year, and color, but all having the same behaviors defined by the methods.

# POINTERS

\*\*Interviewer:\*\* Can you explain the difference between creating objects of a simple type, pointer type, and one created on the heap?

\*\*You:\*\* Sure! Let's consider a class `A`.

## 1. \*\*Simple Type: `A a;`\*\*

- \*\*Explanation:\*\* This creates an object `a` of type `A` on the stack. The object is automatically destroyed when it goes out of scope, which makes memory management simple and efficient.

- \*\*Example:\*\*

```cpp

A a; // 'a' is an instance of class A, created on the stack

```

## 2. \*\*Pointer Type: `A\* a;`\*\*

- \*\*Explanation:\*\* This declares a pointer `a` that can point to an object of type `A`, but it doesn't actually create an object. You need to allocate memory and assign it to this pointer to use it.

- \*\*Example:\*\*

```cpp

A\* a; // 'a' is a pointer to an object of type A, but no object is created yet

a = new A(); // 'a' now points to an object of type A created on the heap

delete a; // manually delete the object to avoid memory leak

```

## 3. \*\*Heap Allocation: `A\* obj = new A();`\*\*

- \*\*Explanation:\*\* This creates an object `obj` of type `A` on the heap. Objects created on the heap are not automatically destroyed when they go out of scope. You need to manually manage their memory using `delete`.

- \*\*Example:\*\*

```cpp

A\* obj = new A(); // 'obj' is a pointer to an object of type A created on the heap

// use 'obj'

delete obj; // manually delete the object to avoid memory leak

```

\*\*Summary:\*\* In simple terms, objects are instances of classes and are necessary to access the class's data members and methods.

- When you create an object of a simple type (`A a;`), it is managed automatically.

- When you use a pointer type (`A\* a;`), you have more flexibility but must manage memory allocation and deallocation yourself.

- Creating an object on the heap (`A\* obj = new A();`) is useful for dynamic memory management but requires explicit handling to prevent memory leaks.

---

This explanation clearly distinguishes between the different ways to create objects, including their memory management implications.