**Key Differences Between Class and Struct**

| **Feature** | **Class** | **Struct** |
| --- | --- | --- |
| **Type** | **Reference Type** (Stored on Heap) | **Value Type** (Stored on Stack) |
| **Memory Allocation** | Allocated on the **heap** (managed by Garbage Collector) | Allocated on the **stack** (faster access) |
| **Default Behavior** | Passed by **reference** (changes reflect in caller) | Passed by **value** (copy is created) |
| **Inheritance** | **Supports Inheritance** | 🚫 **Does NOT support inheritance** (but supports interfaces) |
| **Default Constructor** | Can have a **parameterless constructor** | 🚫 **Cannot have an explicit parameterless constructor** |
| **Performance** | Slower due to heap allocation and GC overhead | Faster due to stack allocation |
| **Best for** | Large, complex objects (e.g., database models) | Small, lightweight objects (e.g., coordinates, points) |

//Implementation with class

public class Car

{

public string Model;

public int Speed;

public Car(string model, int speed)

{

Model = model;

Speed = speed;

}

}

class Program

{

static void Main()

{

Car car1 = new Car("Tesla", 100);

Car car2 = car1; // Reference Copy

car2.Speed = 200;

Console.WriteLine(car1.Speed); // Output: 200 (Reference Change)

}

}

//implementation with structure

public struct CarStruct

{

public string Model;

public int Speed;

public CarStruct(string model, int speed)

{

Model = model;

Speed = speed;

}

}

class Program

{

static void Main()

{

CarStruct car1 = new CarStruct("Tesla", 100);

CarStruct car2 = car1; // Value Copy

car2.Speed = 200;

Console.WriteLine(car1.Speed); // Output: 100 (Separate Copy)

}

}