











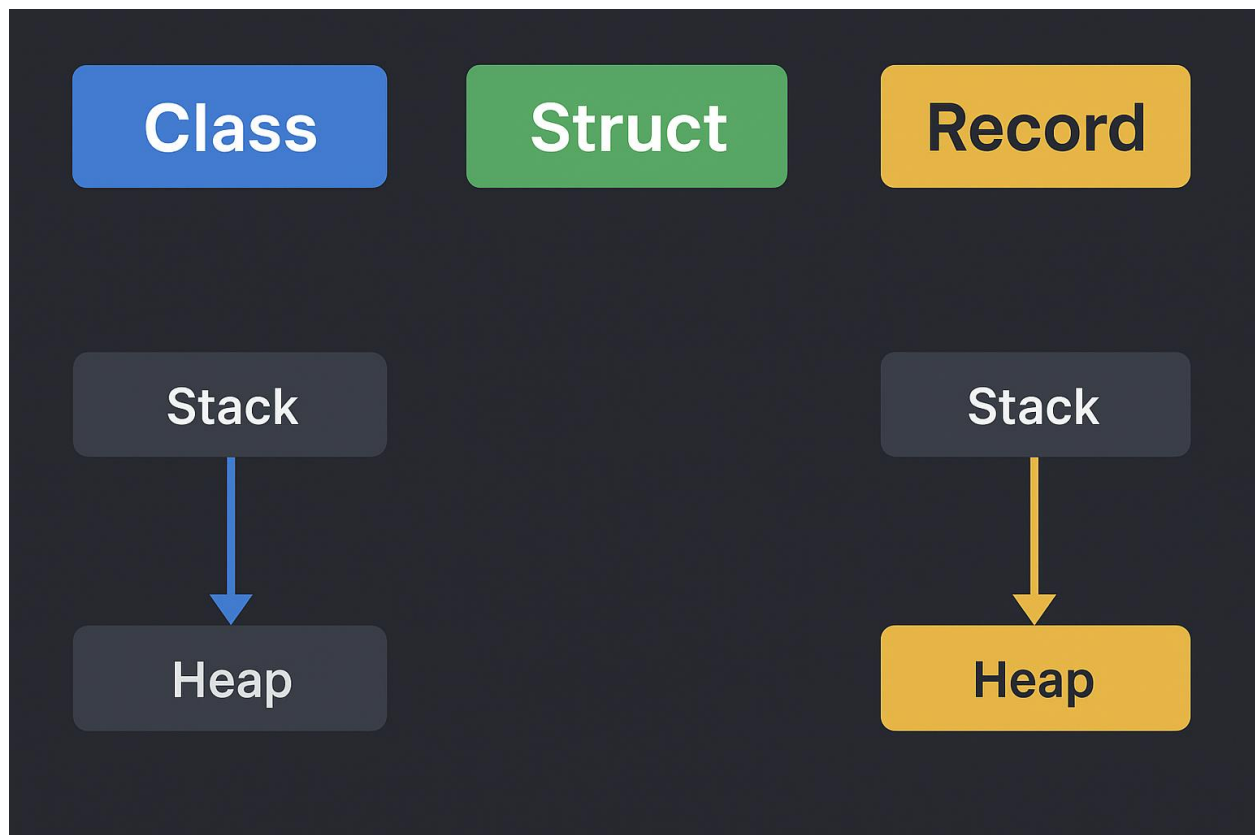


Feature / Aspect	Class	Struct	Record
Type Category	Reference Type	Value Type	Reference Type (default)
Stored In Memory	Heap	Stack (usually)	Heap
Copied By	Reference (both points to same object)	Value (creates a copy)	Reference (but equality by value)
Default Equality (== , .Equals)	Compares memory reference	Compares field values	Compares content (value equality)
Mutability	Usually mutable (can change fields)	Usually, mutable	Usually immutable (init only)
Inheritance Support	✅ Yes (can inherit other classes)	❌ No inheritance allowed	✅ Yes (can inherit and be inherited)
Implements Interfaces	✅ Yes	✅ Yes	✅ Yes
Parameterless Constructor	✅ Allowed	✅ From C# 10+	✅ Allowed
Custom Constructor	✅ Yes	✅ Yes	✅ Yes
Destructors (finalizers)	✅ Allowed	❌ Not allowed	✅ Allowed
Default Access Modifier (members)	private	private	private
Supports readonly / init properties	✅ Yes	✅ Yes	✅ Yes (commonly used)
Implements Polymorphism	✅ Yes (virtual , override)	❌ No	✅ Yes
Default ToString() behavior	Prints class name	Prints struct name	Prints property names & values (auto)



Feature / Aspect	Class	Struct	Record
Default Equals() behavior	Reference equality	Value equality	Value equality (auto-generated)
Can be null?	 Yes	 No (unless Nullable<T>)	 Yes
Used For	Complex data / objects with behavior	Lightweight data structures (e.g. Point, Vector)	Immutable data models, DTOs, records
Performance	Slightly slower (heap allocation)	Faster for small types (stack allocation)	Similar to class but optimized equality
Introduced In	C# 1.0	C# 1.0	C# 9.0
Can use with with expression	 No	 No	 Yes (creates copy with new values)
Supports Deconstruction (var (x, y) = obj)	 No	 No	 Yes
Example Declaration	<code>class Person { public string Name; }</code>	<code>struct Point { public int X; }</code>	<code>record Student(string Name, int Age);</code>

Scenario	Best Choice
You need reference behavior (shared objects)	 Class
You need lightweight copies (performance, small data)	 Struct
You need immutable, data-focused types (auto equality)	 Record



1. The Two Areas of Memory

When a program runs, C# uses two main memory areas:

Area	What it stores	Example
 Stack	Small, short-lived data (local variables, struct copies)	<code>int x = 5;</code>
 Heap	Large, long-lived data (objects, classes, records)	<code>new Person()</code>

2. Class (Reference Type):

```
class Person { public string Name; }
```

When you create a class object:

```
Person p1 = new Person();
```

```
Person p2 = p1;
```

What happens:

- p1 and p2 are both on the **stack**,
but they **point to the same Person** object on the **heap**.
- Changing p2.Name also changes p1.Name.

Memory:

Stack:

p1 → 7

p2 → 7

Heap:

[Person: Name="Ahmed"]

Shared reference, stored on heap.

3. Struct (Value Type):

```
struct Point { public int X; public int Y; }
```

When you create a struct:

```
Point a = new Point { X = 5, Y = 10 };
```

```
Point b = a; // copies the value
```

```
b.X = 100;
```

What happens:

a and b are separate copies on the stack.

Changing b does not affect a.

Memory:

Stack:

a → [X=5, Y=10]

b → [X=100, Y=10]

Independent copy, stored on stack.

4. Record (Reference Type but Compares by Value):

```
record Student(string Name, int Age);
```

When you create two records:

```
Student s1 = new("Ali", 22);
```

```
Student s2 = new("Ali", 22);
```

What happens:

Both objects are on the heap,
but C# considers them equal because their data is the same.

Memory:

Stack:

```
s1 → [Name="Ali", Age=22]
```




```
s2 → [Name="Ali", Age=22]
```

Heap:

two separate objects, but equal by content

Reference type, but compares by value.

Summary Table (Simplified)

Type	Stored In	Copied By	Compares By	Typical Use
 Class	Heap	Reference	Memory Address	Complex objects
 Struct	Stack	Value	Field Values	Small, fast data (Point, Vector)
 Record	Heap	Reference	Field Values	Immutable data, models