Class vs Struct Differences

ASPECT	CLASS	STRUCT
Type	Reference type	Value type
Memory	Heap allocation	Stack allocation
Assignment	Copies reference	Copies entire value
Inheritance	Supports inheritance	No inheritance support
Default Constructor	Can define custom parameterless	Cannot define custom parameterless
Null Values	Can be null	Cannot be null (unless nullable)
Performance	Slower allocation/deallocation	Faster allocation/deallocation
Size Recommendation	Any size	Small data (≤16 bytes recommended)

Other Relations Between Classes:

1. **Composition ("Has-a" relationship)**
   - ✧ One class contains objects of another class as members
   - ✧ Strong ownership - contained object lifecycle depends on container

```
class Car {
    private Engine engine; // Car HAS-A Engine
}
```

2. **Aggregation ("Has-a" relationship)**
   - ✧ Weaker form of composition
   - ✧ Contained objects can exist independently

```
class Department {
    private List<Employee> employees; // Department HAS Employees
}
```

3. **Association ("Uses-a" relationship)**
   - ✧ Classes interact but are independent
   - ✧ Temporary relationships through method parameters

```
class Student {
    public void EnrollIn(Course course) { } // Student USES Course
}
```

4. **Dependency ("Depends-on" relationship)**
   - ✧ One class depends on another to function
   - ✧ Usually through method parameters or local variables

```
class OrderProcessor {
    public void Process(PaymentService service) { } // Depends on PaymentService
}
```

5. **Realization/Implementation**
   - ✧ Class implements an interface contract

```
class Rectangle : IShape { } // Rectangle IMPLEMENTS IShape
```