Class & Struct - Reference vs Value Type - Heap vs Stack Memory

Friday, May 10, 2024 1:34 PM

Class

Blue print and is a reference type that means Memory is holding the reference, not actual value.

Struct

Value type.

Object

An Entity of that blue print.

Difference between Class and Struct

Structs are similar to classes in that they represent data structures that can contain data members and function members. However, unlike classes, structs are value types and do not require heap allocation. A variable of a struct type directly contains the data of the struct, whereas a variable of a class type contains a reference to the data, the latter known as an object.

As described in §8.3.5, the simple types provided by C#, such as int, double, and bool, are, in fact, all struct types.

Struct	**Class**
Value Type like int double	Reference Type
Uses only Stack	Uses Heap
Allocation and Deallocation memory is cheaper because of usage of Stack	Expensive
Assigning this to new variable copies the value, that's why bigger object consumes much time and space.	Assigning this to new variable copy the reference.

References

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/structs

What is Boxing and Unboxing and its impact on Garbage Collection.

Boxing is a process to convert any value type to type object, during a boxing, the value is wrapped using the System. Object and is stored in managed heap.

Boxing Example

```
int i = 123;
// The following line boxes i.
object o = i;
```

Unboxing Example

```
o = 123;
i = (int)o;  // unboxing
```