**Class vs Struct(Structures)**

1. **Type Feature**

 **Reference Type(class)**: Classes are reference types. Reference type holds a reference (or address) to where the actual data is stored. Example: Like sharing the same hotel room key, multiple variables can point to the same object and changes made by one are visible to all.

 **Value Type(struct)**: Structs are value types. Value types store the actual data directly in the variable. Example: Like individual shopping lists each variable holds its own copy of data and changes to one variable don’t affect others.

1. **Memory allocation Feature**

 **Class (Reference Type)**: Stored on the heap and the variable holds a reference to the object. Multiple variables can refer to the sameobject so changes made by one will affect others.

 **Struct** (**Value Type)**: Stored on the stack and the variable holds the actual data. Each variable holds its owncopy of the data so changes made to one donot affect others.

1. **Inheritance Feature**

 **Class (Reference Type)**: Classes support inheritance, meaning you can create new classes based on existing ones.

 **Struct** (**Value Type)**: Structs do not support inheritance.

1. **Default Constructor Feature**

 **Class (Reference Type)**: You can define your own default constructor. This means you can decide what values are assigned to an object's properties (like name or age) when it is first created. You can also provide additional constructors to allow for more specific initial values

 **Struct** (**Value Type)**: In struct C# provides a default constructor that you cannotchange. It automatically sets all properties to their default values (like 0 for numbers).

1. **Nullability Feature**

 **Class (Reference Type)**: Class objects can be null (meaning they can have no value assigned).

 **Struct** (**Value Type)**: Structs cannot be null unless you explicitly make them nullable.

1. **Copy Behaviour Feature**

 **Class (Reference Type)**: When you copy a class variable both variables point to the same object. Changes made through one reference will be reflected in the other.

 **Struct** (**Value Type)**: When you copy a variable of a struct a new independent copy is created. Changes made to one copy do not affect the other.