# Assignment DAY 07

### Why does defining a custom constructor suppress the default constructor in C#?

When you define a custom constructor in a class, the default constructor (a constructor with no parameters) is no longer automatically provided by the compiler. This is because C# assumes that you want to fully control the object initialization. If you need a parameterless constructor, you must explicitly define it.

### How does method overloading improve code readability and reusability?

Method overloading allows multiple methods with the same name but different parameters. This improves code readability by using a consistent name for similar operations and reusability by reducing the need for different method names for similar functionalities.

### What is the purpose of constructor chaining in inheritance?

Constructor chaining allows a derived class to call a constructor from its base class, ensuring proper initialization of the base class properties before initializing the derived class properties. It helps to avoid code duplication and ensures that the base class is properly initialized.

### How does new differ from override in method overriding?

'new' hides a method in the base class, but does not allow polymorphic behavior at runtime. It provides a new implementation while keeping the base class method intact. 'override' modifies the method in the base class, and allows polymorphism, where the overridden method is called at runtime based on the actual object type.

### Why is ToString() often overridden in custom classes?

Overriding the ToString() method in custom classes allows the class to provide a meaningful string representation of an object, making debugging and logging easier by outputting useful information about the object instead of the default 'ClassName'.

### Why can't you create an instance of an interface directly?

An interface defines a contract but doesn't provide an implementation. Therefore, you cannot create an instance of an interface. You can only create instances of classes that implement the interface.

### What are the benefits of default implementations in interfaces introduced in C# 8.0?

Default implementations allow you to define method bodies within interfaces, making it easier to extend existing interfaces without breaking the implementing classes. It improves maintainability and backward compatibility.

### Why is it useful to use an interface reference to access implementing class methods?

Using an interface reference allows you to write code that is decoupled from specific implementations. It enhances flexibility and allows polymorphic behavior, where the method implementation depends on the actual object type, not the reference type.

### How does C# overcome the limitation of single inheritance with interfaces?

C# allows a class to implement multiple interfaces, overcoming the single inheritance limitation by providing multiple inheritance of behaviors. This enables a class to inherit from multiple sources of functionality.

### What is the difference between a virtual method and an abstract method in C#?

'virtual' methods provide a default implementation that can be overridden by derived classes, whereas 'abstract' methods do not provide an implementation and must be implemented by derived classes.

### What is the difference between class and struct in C#?

Classes are reference types, while structs are value types. This means that classes are stored on the heap and passed by reference, whereas structs are stored on the stack and passed by value. Structs have no inheritance but can implement interfaces.

### If inheritance is a relation between classes, clarify other relations between classes.

Other common relations between classes include composition, where one class contains instances of other classes; aggregation, which is a special form of composition where the contained objects can exist independently; and association, which is a general relationship between two or more classes.