DAY 7 (QUESTION)

Part 1

1. Why does defining a custom constructor suppress the default constructor in C#?
    - the class assumes that the custom constructor defines how objects should be initialized and stop automatically generated default constructor

2. How does method overloading improve code readability and reusability?
    - Readability >> allow the same method name use to be used for similar operations on different types || parameter number .
    - Reusability>> minimizing code duplication and encapsulating behavior under one method name.

3. What is the purpose of constructor chaining in inheritance?
    - Prevents redundancy by allowing the parent class to handle its initialization , and child class inherits && add specific properties

4. How does new differ from override in method overriding?
    - Override used virtual method defined in the parent class with a new implementation in the derived class.(support Polymorphism).
    - New Used to hide a method in the base class without overriding it.(searched this point)

5. Why is ToString() often overridden in custom classes?
    - ToString more Readability>>easy understanding by returning a string that represents data in a readable format.

6. Why can't you create an instance of an interface directly?
    - Interface is rule not an implementation.
    - signature for property && signature for method
7. What are the benefits of default implementations in interfaces introduced in C# 8.0?
    - complete function (signature + body)

8. Why is it useful to use an interface reference to access implementing class methods?
    - Interface ref support use Polymorphism .

9. How does C# overcome the limitation of single inheritance with interfaces?
    - class can implement multiple interfaces.

10. What is the difference between a virtual method and an abstract method in C#?
    - abstract method : A method declared in a base class **without any implementation**.
    - virtual method : A method with a **default implementation** in the parent class , can override in derived class.

Part 2

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

	class	struct
type	Reference type	Value type
Inheritance	allow	Not allow
Memory Location	heap	stack
Default Constructor	Has Default Constructor	Not has

- Structs is value type ClR allocate 4 bytes at stack
- Classes is reference type declare ref of type object, refer to null

,ClR allocate 4 bytes at stack , CLR allocate 0 bytes at heap

If inheritance is relation between classes clarify other relations between classes.

Relationship	Description	Example
Inheritance	"Is-a" relationship; one class inherits from another.	`Dog` is a `Animal`.
Association	"Uses-a" relationship; one class uses another class.	`Student` uses `Course`.
Aggregation	"Has-a" relationship; a whole has parts, but parts can exist independently.	`Library` has `Books`.
Composition	"Has-a" relationship; a whole has parts, and parts cannot exist independently.	`Car` has an `Engine`.
Dependency	"Uses-a" relationship; one class depends on another for functionality.	`Printer` depends on `Report`.
Realization	A class implements an interface.	`Car` implements `IVehicle`.
Polymorphism	Objects of different types are treated as objects of a common base type.	`Shape` reference can refer to both `Rectangle` and `Circle`.