“Part01”

Q1:- Why is it better to code against an interface rather than a concrete class?

🡪 Enhances flexibility, maintainability and testability making it a best practice in software development and It allows for cleaner architecture and easier adaptation to change.

-----------------------------------------------------------------------------------

Q2:- When should you prefer an abstract class over an interface?

🡪 Prefer an abstract class when you need to share code, manage state, provide default behavior or establish a clear hierarchy among related classes.

Interfaces are better when you want to define a contract without dictating how that contract should be fulfilled.

-----------------------------------------------------------------------------------

Q3:- How does implementing IComparable improve flexibility in sorting?

🡪 You can define how instances of your class should be compared and this allows you to specify custom sorting logic based on the properties that are relevant to your application.

-----------------------------------------------------------------------------------

Q4:- What is the primary purpose of a copy constructor in C#?

🡪 To create a new instance of a class as a copy of an existing instance and this is particularly useful for classes that contain reference types as it allows for a controlled way to duplicate objects while managing their state .

-----------------------------------------------------------------------------------

Q5:- How does explicit interface implementation help in resolving naming conflicts?

🡪 Resolve naming conflicts by allowing a class to implement interface members with the same name in a way that clearly differentiates them. This approach enhances encapsulation.

-----------------------------------------------------------------------------------

Q6:- what is abstraction as a guideline, what’s its relation with encapsulation?

🡪 abstraction simplifies complex systems by focusing on essential characteristics while hiding unnecessary details, while encapsulation bundles data and methods together and restricts access to the internal state of an object.

-----------------------------------------------------------------------------------

Q7:- How do default interface implementations affect backward compatibility in C#?

🡪 by allowing to add new methods to interfaces without breaking existing implementations.

-----------------------------------------------------------------------------------

Q8:- How does constructor overloading improve class usability?

🡪 by providing flexibility in object creation, enhancing readability and offering convenience for users.

-----------------------------------------------------------------------------------

“Part02”

1: - LinkedIn article about abstract class?

🡪 Abstract classes are essential in object-oriented programming, providing a blueprint for derived classes while preventing direct instantiation also they allow you to define common behavior and properties and ensuring consistency across subclasses.

-----------------------------------------------------------------------------------

2:- What we mean by coding against interface rather than class? and if u get it so What we mean by code against abstraction not concreteness?

🡪 Coding against an interface means designing your code to depend on interfaces rather than concrete implementations (classes).

Coding against abstraction emphasizes using abstract types (like interfaces or abstract classes) rather than specific and concrete implementations.

-----------------------------------------------------------------------------------

3:- What is abstraction as a guideline and how we can implement this through what we have studied?

🡪 guideline in OOP that helps manage complexity by exposing only the necessary details and hiding the rest and It can be implemented through interfaces and abstract classes also allowing for flexible.

-----------------------------------------------------------------------------------

“Part03 Bonus”

1:- What is operator overloading?

🡪 is a feature to redefine the behavior of standard operators for user to defined types (classes or structs) also This enables objects of these types to be manipulated using familiar operators, enhancing code readability and expressiveness.

-----------------------------------------------------------------------------------