Part 01

1) Question: Why is it recommended to explicitly assign values to enum members in some cases?

Explicitly assigning values to enum members makes the code clearer and more predictable, especially when enums are used in databases or external systems where specific values are expected.

2) Question: What happens if you assign a value to an enum member that exceeds the underlying type's range?

If you assign a value outside the range of the underlying type (e.g., a value too large or small for the base type like byte or int), you may encounter errors or unexpected behavior, such as invalid casts.

3) Question: What is the purpose of the virtual keyword when used with properties?

The 'virtual' keyword allows properties or methods to be overridden by derived classes, enabling polymorphism and flexibility in class behavior.

4) Question: Why can’t you override a sealed property or method?

'sealed' signifies that a method or property cannot be overridden further. It is used to prevent a derived class from modifying its behavior, providing consistency.

5) Question: What is the key difference between static and object members?

Static members belong to the class itself, while object members belong to instances of the class. Static members are shared among all instances, whereas object members are specific to each instance.

6) Question: Can you overload all operators in C#? Explain why or why not.

No, not all operators can be overloaded. For example, you cannot overload 'is' or 'as' operators, because they have specific language semantics that cannot be changed.

7) Question: When should you consider changing the underlying type of an enum?

You should change the underlying type of an enum when you need to store a larger range of values or optimize memory usage. For example, changing from 'byte' to 'int' if more values are required.

8) Question: Why can't a static class have instance constructors?

A static class cannot have instance constructors because it is not meant to be instantiated. It only contains static members and is accessed directly through the class name.

9) Question: What are the advantages of using Enum.TryParse over direct parsing with int.Parse?

Enum.TryParse is safer because it handles invalid input gracefully, returning a Boolean value to indicate success or failure, avoiding exceptions.

10) Question: What is the difference between overriding Equals and == for object comparison in C# struct and class?

Overriding 'Equals' defines the custom logic for equality comparison. '==' checks reference equality by default for classes but can be overridden for value types (structs).

11) Question: Why is overriding ToString beneficial when working with custom classes?

Overriding ToString provides a meaningful string representation of an object, useful for debugging, logging, and displaying the object in UI components.

12) Question: Can generics be constrained to specific types in C#? Provide an example.

Yes, generics can be constrained. For example, 'public class MyClass<T> where T : IComparable' restricts T to types that implement IComparable.

13) Question: What are the key differences between generic methods and generic classes?

Generic methods are methods that can operate on types specified at runtime, while generic classes define types that can vary across instances. Methods are specific to the class they belong to.

14) Question: Why might using a generic swap method be preferable to implementing custom methods for each type?

A generic swap method can work with any type, reducing redundancy and making the code more maintainable. It avoids the need to implement swap logic for every data type.

15) Question: How can overriding Equals for the Department class improve the accuracy of searches?

Overriding Equals allows accurate comparison of Department objects, ensuring that searches and comparisons are based on the actual data rather than object references.

16) Question: Why is == not implemented by default for structs?

By default, structs in C# use value-type equality, so the compiler does not provide an implementation of '==' unless explicitly defined. You must override it to specify custom equality logic.

Part 02

2) What we mean by Generalization concept using Generics?

allows writing flexible, reusable code that works with any data type, enhancing type safety without specifying the exact type.

3) What do we mean by hierarchy design in real business?

refers to organizing roles, responsibilities, and processes in a layered structure to define authority, accountability, and efficient decision-making.