Part02:

1- What we mean by Generalization concept using Generics ?

Ans: **Generalization** using **Generics** refers to the ability to write code that is decoupled from specific data types. Instead of writing separate logic for int, string, or custom objects, you define a **type parameter** that acts as a placeholder.

Before Generics, developers often used the object type to achieve generalization. However, this caused two major issues:

1. **Performance:** Value types like int had to be boxed into an object and unboxed back, which is computationally expensive.
2. **Type Safety:** You could accidentally add a string to a list of objects intended for int, leading to runtime crashes.

Generics solve both. They allow you to define a class or method once, but the compiler specializes it for the exact type you use at compile time.

2- What we mean by hierarchy design in real business ?

Ans: In a real business context, **Hierarchy Design** refers to organizing data or entities into a parent-child relationship. This mimics how actual companies operate: a CEO has Managers, who have Employees; or a Category has Sub-categories, which have Products.

1. Inheritance: The "Is-A" Hierarchy

This is the most direct way to design a hierarchy. You define a **Base Class** with shared logic and **Derived Classes** that specialize that logic.

2. **Interface-Based Hierarchy (Abstraction)** Sometimes business entities aren't related by "blood" (inheritance) but by "ability."