What is a Copy Constructor?

A copy constructor is a special constructor that initializes a new object as a copy of an existing object. Unlike C++, C# does not provide a default copy constructor for classes or structs. Instead, you must explicitly define one to control how data is copied, especially when dealing with **reference-type fields** (to avoid shallow copies).

- **Deep Copy vs. Shallow Copy**: A copy constructor ensures a **deep copy** (new instances of reference-type fields) rather than a shallow copy (shared references).
- **Structs vs. Classes**:
  - Structs (value types) are inherently copied when assigned, but a copy constructor can enforce custom logic.
  - Classes (reference types) require explicit copy logic to avoid unintended reference sharing.

What is an Indexer?

An indexer allows an object to be accessed like an array using an index (e.g., obj[0] or obj["key"]). It is syntactic sugar for a property named this[] and is useful when a class/struct represents a **collection** of data.

When to Use Indexers

- **Custom Collections**: Access elements in a list, dictionary, or matrix.
- **Data Abstraction**: Expose data in a structured way (e.g., database rows, spreadsheet cells).
- **Multi-Dimensional Data**: Use multiple parameters (e.g., matrix[row, col]).

Business Use Cases

1. **Database Wrapper**: Access records by index or key.
2. **Spreadsheet/Grid**: Retrieve cells by row and column.
3. **Configuration Manager**: Access settings by string keys.
4. **Shopping Cart**: Access items by product ID.