Design patterns are typical solutions to commonly occurring problems in software design. They are like pre-made blueprints that you can customize to solve a recurring design problem in your code.

HISTORY:

The concept of patterns was first described by Christopher Alexander in **A Pattern Language: Towns, Buildings, Construction**. The book describes a "language" for designing the urban environment.

The idea was picked up by four authors: Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm. In 1994, they published **Design Patterns: Elements of Reusable Object-Oriented Software**, in which they applied the concept of design patterns to programming.

WHY learn design patterns?

- because it teaches you how to solve all sorts of problems using principles of object-oriented design.
- Design patterns define a common language that you and your teammates can use to communicate more efficiently. You can say, "Oh, just use a Singleton for that," and everyone will understand the idea behind your suggestion. No need to explain what a singleton is if you know the pattern and its name.

CLASSIFICATION:

The most basic and low-level patterns are often called *idioms*. They usually apply only to a single programming language.

- **Creational patterns** provide object creation mechanisms that increase flexibility and reuse of existing code.
    - **Factory Method -** is a creational design pattern that provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created.
    - **Abstract Factory Method -** is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.
    - **Builder -** is a design pattern that lets you construct complex objects step-by-step. It allows you to build different types and representations of an object that share the same construction code.
    - **Prototype -** is a creational design pattern that lets you copy existing objects without making your code dependent on their classes. Prototype allows us to hide the complexity of making new instances from the client.
    - **Singleton -** is one of the most fundamental and widely used creational design patterns in software engineering. It is designed to ensure that a class has only one instance and provides a global point of access to that instance.
        - Lazy Singleton, Thread-Safe Singleton, and Enum Singleton.
- **Structural patterns** explain how to assemble objects and classes into larger structures, while keeping these structures flexible and efficient.
    - **Bridge -** structural design pattern that is meant to decouple an abstraction from its implementation so that the two can very independently.
        - Structural - deals with the composition of objects or classes
        - Decouple - separating the abstraction from its implementation

- **Behavioral patterns** take care of effective communication and the assignment of responsibilities between objects.
    - **Observer -** maintains one-to-many dependency between Subject(Observable) and its dependents(Observer) in such a way that whenever state of Subject changes, its dependents get notified.
- **Null Object -** The Null object pattern is a design pattern that simplifies the use of dependencies that can be undefined.