**What are design patterns?**

We can define Design patterns something like that, **“Design patterns are documented, tried and tested (well) solutions for a recurring problems in a given context”**. What does that mean? It means that, Design pattern is a solution for a certain type of problem that people has been using for many years. The solution is tested and accepted as a general solution.

Let’s say if we want to implement a sorting algorithm, the first thing comes to our mind is bubble sort. Here the problem is sorting and solution is bubble sort. Same holds true for design patterns. For example if we want that in a system a particular object can have only one instance. The solution of this problem is “Singletone” design pattern.

The 23 patterns described by Gang of Four (GoF) are generally considered the foundation of all other design patters. They are categorized in three groups:

1. **Creational Patterns**
   * **Abstract Factory**:- Creates an instance of several families of classes
   * **Builder**: - Separates object construction from its representation
   * **Factory Method**:- Creates an instance of several derived classes
   * **Prototype**:- A fully initialized instance to be copied or cloned
   * **Singleton**:- A class in which only a single instance can exist
2. **Structural Patterns:**
   * **Adapter**:-Match interfaces of different classes.
   * **Bridge**:-Separates an object’s abstraction from its implementation.
   * **Composite**:-A tree structure of simple and composite objects.
   * **Decorator**:- Attached additional responsibility to an object dynamically.
   * **Façade**:-A single class that represents an entire subsystem.
   * **Flyweight**:- A fine-grained instance used for efficient sharing.
   * **Proxy**:-An object representing another object.

1. **Behavioral Patterns:**
   * **Chain of Responsibility**:- A way of passing a request between a chain of objects.
   * **Command**:-Encapsulate a command request as an object.
   * **Interpreter**:-A way to include language elements in a program
   * **Iterator**:-Sequentially access the elements of a collection.
   * **Mediator**:-Defines simplified communication between classes.
   * **Memento**: Capture and restore an object's internal state
   * **Observer**: - A way of notifying change to a number of classes.
   * **State**:-Alter an object's behavior when its state changes.
   * **Strategy**:-Encapsulates an algorithm inside a class.
   * **Template Method**:-Defer the exact steps of an algorithm to a subclass.
   * **Visitor**:-Defines a new operation to a class without change.