**Design Pattern**

A design pattern is a general, reusable solution to a commonly occurring problem in software design. Unlike complete designs that the programmer would translate directly into code, these patterns represent guidelines that assist the developer in approaching the problem at hand in a variety of contexts. They are a best practice that enables software developers to write robust and maintainable code.

There are three main types of patterns:

Creational Patterns: These deal with the mechanisms of object creation, trying to create objects in a manner suitable for the given situation. Some of the most common examples are:

- Singleton: Allows for controlled use of a class by ensuring that there is only one instance of it, and then provides a global point of access to that instance.

- Factory Method: Declares an interface for creating an object, but it is up to its subclasses to alter the type of object that will be created.

Structural Patterns: This group of patterns concerns with object composition, describing, in other words, how objects and classes can be combined to form larger structures. Some common examples would be:

- Adapter: Allows two incompatible interfaces to work together.

- Composite: Allows composing objects into tree structures to represent part-whole hierarchies.

Behavioral Patterns: These patterns deal with the relationships between objects, focusing on their responsibilities and algorithms. Some of the common examples would be:

- Observer: A one-to-many dependency between objects, so that when one object changes state, all of its dependents are notified and updated automatically.

- Strategy: Defines a family of algorithms, encapsulates each one, and makes them interchangeable.

These in turn will allow better flexibility, reusability, and maintainability of the designs of the software.