**Object Oriented Design Patterns**

* **What are design patterns?**
  + They are typical solutions to commonly occurring problems in software design. They are like already available blueprints ready to be customized to our problem.
  + These are typical solutions to common problems in object-oriented design.
  + Patterns are not a specific code but are a general concept of solving a particular problem. We can follow the pattern details and implement that solution which suits the needs of our program.
  + When a solution repeats over and over in various projects, it is labelled as a pattern under some name.
  + Patterns are tried and tested solutions to common problems in software design.
  + **How do patterns differ from algorithms?**
    - While algorithms are step by step actions to achieve a goal, a pattern is a high-level description of a solution.
    - A simple analogy for:
      * **Algorithm:** is a cooking recipe, which will take clear steps to achieve the result.
      * **Patterns:** this is more like a blueprint; we will know what the result is and what the features are and its up to us how we implement it.
  + **What does the pattern consist of?**
    - The pattern description usually contains:
      * **Intent** briefly describes the problem and the solution.
      * **Motivation** further explains the problem and the solution the pattern makes possible.
      * **Structure** of classes shows each part of the pattern and how they are related.
      * **Code example** in one of thepopular programming languages which makes it easier to grasp the idea behind the pattern.
* **Classification of patterns**
  + Design patterns differ in the level of their complexity, level of detail, and scale of applicability to the entire system being designed.
  + The most basic and low-level patterns are often called ***idioms.*** They apply to only a single programming language.
  + The most universal and high-level patterns are called ***architectural patterns.*** Developers can implement them in any language. These can also be used to design architecture of entire applications unlike other patterns.
  + In addition to these patterns, they can also be categorized based on their intent, or purpose.
  + Major groups of patterns are:
    - Creational patterns
    - Structural patterns
    - Behavioral patterns

1. **Creational patterns:**
   * These patterns provide various object creation mechanisms, which increase flexibility and reuse of existing code.
     + Factory method
     + Abstract factory
     + Builder
     + Prototype
     + Singleton
2. **Structural patterns**
   * These patterns explain how to assemble objects and classes into larger structures while keeping these structures flexible and efficient.
     + Adapter
     + Bridge
     + Composite
     + Decorator
     + Facade
     + Flyweight
     + Proxy
3. **Behavioral patterns**
   * These patterns are concerned with algorithms and the assignment of responsibilities between objects.
     + Chain of Responsibility.
     + Command
     + Iterator
     + Mediator
     + Memento
     + Observer
     + State
     + Strategy
     + Template method
     + Visitor