**What is Design Pattern?**

Design patterns are solutions to general problems that software developers faced during software development.

**What is Gang of Four (GOF)?**

In 1994, four authors Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides published a book titled **Design Patterns - Elements of Reusable Object-Oriented Software** which initiated the concept of Design Pattern in Software development.

These authors are collectively known as **Gang of Four (GOF)**. According to these authors design patterns are primarily based on the following principles of object orientated design.

* Program to an interface not an implementation
* Favor object composition over inheritance

## Types of Design Patterns

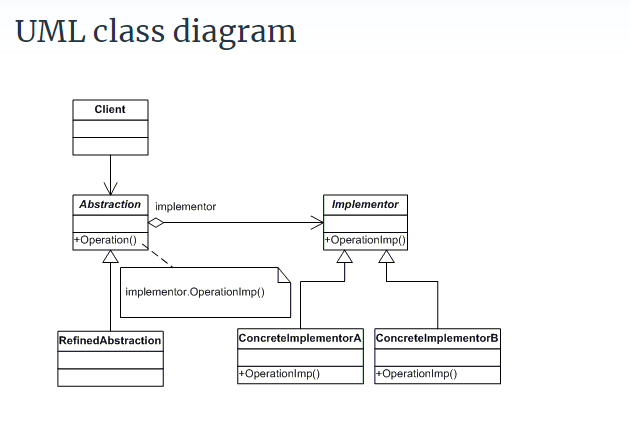
As per the design pattern reference book **Design Patterns - Elements of Reusable Object-Oriented Software** , there are 23 design patterns which can be classified in three categories: Creational, Structural and Behavioral patterns.

**Bridge design pattern** is a **Structural Design Pattern.**

**What is Bridge Design Pattern?**

Bridge is used when we need to decouple an abstraction from its implementation so that the two can vary independently. This type of design pattern comes under structural pattern as this pattern decouples implementation class and abstract class by providing a bridge structure between them.

This pattern involves an interface which acts as a bridge which makes the functionality of concrete classes independent from interface implementer classes. Both types of classes can be altered structurally without affecting each other.



Participants

The classes and objects participating in this pattern are:

* **Abstraction**   (BusinessObject)
  + defines the abstraction's interface.
  + maintains a reference to an object of type Implementor.
* **RefinedAbstraction**   (CustomersBusinessObject)
  + extends the interface defined by Abstraction.
* **Implementor**   (DataObject)
  + defines the interface for implementation classes. This interface doesn't have to correspond exactly to Abstraction's interface; in fact the two interfaces can be quite different. Typically the Implementation interface provides only primitive operations, and Abstraction defines higher-level operations based on these primitives.
* **ConcreteImplementor**   (CustomersDataObject)
  + implements the Implementor interface and defines its concrete implementation.