Provides generic reusable solution for most of the problem that occur during the development process.

Patterns basically shows how the class are defined and hoe different components communicate with each other.

Speed up the development process because address, most of the scenarios.

Language independent because it just provides an idea how the components need to be designed.

By using this you can make more flexible, reusable and maintainable code.

Its not mandatory to implement this pattern. By doing this we can address most of the future issue that might raise in the application.

Goal: Understand the mechanism of the design pattern and based on the application scenario you can use the right pattern for the development.

**Major classification:**

**Creational design pattern:**

Design patter mainly deals with the Class instantiation or object creation

Further categorized as Class creation pattern, object creation patter.

Class creation pattern: extensively uses class inheritance

Object creation pattern uses delegation model

Factory, abstract factor, singleton, builder, object pooling, prototype

Application of this pattern:

1. You have DB connection class that implement the DB connection related logic and in code multiple places you have to use this to perform db related operation. Using this in multiple places lead to creation of multiple connection to avoid this may be we need to design class as singleton
2. Suppose you want to implement multiple class doing similar things and you wan to have loosely coupled code go for factor pattern. Say application connect to some db server and for now its oracle and in future you want to change it to Sql with factory pattern we can achieve this

**Structural design pattern:**

Deals with the structuring of the patterns where in we organize different objects and class together to form lager structure and provide new fictionality.

1. Two interfaces are incompatible with each other but we want to establish communication between these to using adaptor called adaptor pattern.
2. Converts the interface of the class to another interface or class.

Adaptor patter, bridge pattern, Proxy, Decorator,Composite, Facade

**Behavioural design pattern:**

Deals with establishing the communication between the objects.