**Dependency Injection**

Dependency Injection (DI) is a design pattern used to implement IoC(Inversion of control). It allows the creation of dependent objects outside of a class and provides those objects to a class through different ways. Using DI, we move the creation and binding of the dependent objects outside of the class that depends on them. Dependency Injection reduces the hard-coded dependencies among classes by injecting those dependencies at run time instead of design time technically.

Dependency Inversion Principle gives us the guidelines for writing loosely-coupled classes. Here is the definition:

* High-level modules should not depend on low-level modules. Both should depend on abstractions.
* Abstractions should not depend upon details. Details should depend upon abstractions.

The Dependency Injection pattern involves 3 types of classes

1. **Client Class:** The client class (dependent class) is a class which depends on the service class.
2. **Service Class:** The service class (dependency) is a class that provides service to the client class.
3. **Injector Class:** The injector class injects the service class object into the client class.

**Types of Dependency Injection**

1. **Constructor Injection:**

In the constructor injection, the injector supplies the service (dependency) through the client class constructor.

1. **Property Injection:**

In the property injection (aka the Setter Injection), the injector supplies the dependency through a public property of the client class.

1. **Method Injection:**

In this type of injection, the client class implements an interface which declares the method(s) to supply the dependency and the injector uses this interface to supply the dependency to the client class.

**Advantages of Dependency Injection**

1. Reduces class coupling
2. Increases code reusability
3. Improves code maintainability
4. Make unit testing possible

**What is Inversion Of Control (IOC)**

Inversion of Control is a software design architecture where the framework controls the program flow. It means we are changing the control from the normal way. IOC is used to increase modularity of the program and make it extensible.

**Difference between DIP and IOC:**

As stated earlier, DIP says high level module should not depend on low level module for abstraction and IOC provides abstraction. So if you want to make independent higher level module from the lower level module then you have to invert the control so that low level module will not control the interface and creation of the object.

**Service Locator:**

DI is not the only way to break this dependency, SL is such another technique to implement Inversion of Control.

The idea behind a service locator is to have an object that knows how to get all of the services that an application might need.

**Difference between DI and SL:**

The Service Locator allows you to "resolve" a dependency within a class and the Dependency Injection allows you to "inject" a dependency from outside the class.

When you use a service locator, every class will have a dependency on your service locator but in dependency injection,the dependency injector will typically be called only once at startup, to inject dependencies into the main class.

The Service Locator pattern is easier to use in an existing codebase as it makes the overall design loosely coupled without forcing changes to the public interface. Code that is based on the Service Locator pattern is less readable than the equivalent code that is based on Dependency Injection.