**IOC :** Inversion Of Control is a design principle, which explains that an object creation,dependency injection, object lifecycle management, object destruction… etc. has to managed by an external entity.

In Spring, Spring Container is the external entity, which implements principles of IOC.

**Dependency Injection**

Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application.

Dependency Injection makes our programming code loosely coupled. To understand the DI better, Let's understand the Dependency Lookup (DL) first:

**Dependency Lookup**

The Dependency Lookup is an approach where we get the resource after demand. There can be various ways to get the resource for example:

A obj = **new** AImpl();

In such a way, we get the resource(instance of A class) directly by new keyword. Another way is factory method:

A obj = A.getA();

**Problems of Dependency Lookup**

There are mainly two problems of dependency lookup.

* **tight coupling** The dependency lookup approach makes the code tightly coupled. If resources are changed, we need to perform a lot of modification in the code.
* **Not easy for testing** This approach creates a lot of problems while testing the application especially in black box testing.

**Dependency Injection**

The Dependency Injection is a design pattern that removes the dependency of the programs. In such cases we provide the information from the external source such as an XML file. It makes our code loosely coupled and easier for testing.

The process of injecting dependencies into the dependent objects is called Dependency

Injection.

Dependency Injection taking care by the spring container(IOC container) in spring.

In spring DI is generally achieved in 2 ways,

* setter injection .
* constructor injection

class Employee{ -> Dependent

int eid,

String ename; -> Dependencies

Address address;

}

class Address{

int aid,

String loc;

}

**Setter injection:**

If dependency is injected to dependent object via setter method, it is known as setter

injection.

Step 1: In spring bean class declare the dependency variable.

Step 2: Define setter method for that variable.

Step 3: Use <property> tag in bean configuration file and supply the value.

**Constructor injection:**

If dependency is injected to dependent object via constructor, it is known as constructor

injection.

Step 1: In spring bean class declare dependency variables.

Step 2: Define constructor which will take that variable as parameter.

Step 3: Use <constructor-arg> tag in bean configuration file and supply the value.

**Note** : To inject primitive values use value attribute/tag

To inject object references use ref attribute/tag