These are patterns to achieve loose coupling in java programming

**DI(Dependency Injection)**:  
Dependency injection is a pattern used to create instances of objects that other objects rely upon without knowing at compile time which class will be used to provide that functionality or simply the way of injecting properties to an object is called dependency injection.

We have three types of Dependency injection

1. Constructor Injection
2. Setter/Getter Injection
3. Interface Injection

Spring will support only Constructor Injection and Setter/Getter Injection.

**IOC(Inversion Of Control):**  
Giving control to the container to create and inject instances of objects that your application depend upon, means instead of you are creating an object using the new operator, let the container do that for you. Inversion of control relies on dependency injection because a mechanism is needed in order to activate the components providing the specific functionality

The two concepts work together in this way to allow for much more flexible, reusable, and encapsulated code to be written. As such, they are important concepts in designing object-oriented solutions.

**Example for Dependency injection**

Previously we are writing code like this

Public MyClass{

DependentClass dependentObject

/\*

At somewhere in our code we need to instantiate

the object with new operator inorder to use it or perform some method.

\*/

dependentObject= new DependentClass();

dependentObject.someMethod();

}

With Dependency injection, the dependency injector will take off the instantiation for us

Public MyClass{

/\* Dependency injector will instantiate object\*/

DependentClass dependentObject

/\*

At somewhere in our code we perform some method.

The process of instantiation will be handled by the dependency injector

\*/

dependentObject.someMethod();

}

The above process of giving the control to some other (for example the container) for the instantiation and injection can be termed as **Inversion of Control**

Inversion of control means the program delegates control to someone else who will drive the flow IOC (Inversion of control) is a general parent term while DI (Dependency injection) is a subset of IOC. IOC is a concept where the flow of application is inverted. The control of the logic which is not part of that entity is taken by someone else. DI provides objects that an object needs. So rather than the dependencies construct themselves they are injected. The biggest benefit achieved by the above approach is “Decoupling”, we can invoke an object and pass any object keeping objects independent improving reusability and maintenance.