*Dependency Injection is****a fundamental aspect of the Spring framework****, through which the Spring container “injects” objects into other objects or “dependencies”.*

*Dependency injection (DI) is a process whereby objects define their dependencies (that is, the other objects with which they work) only through constructor arguments, arguments to a factory method, or properties that are set on the object instance after it is constructed or returned from a factory method. The container then injects those dependencies when it creates the bean.*

*Simply put, this allows for loose coupling of components and moves the responsibility of managing components onto the container.*

*DI exists in two major variants: Constructor-based dependency injection and Setter-based dependency injection –*

***Setter Dependency Injection (SDI)****: This is the DI methods. In this, the DI will be injected with the help of setter and/or getter methods. Now to set the DI as SDI in the bean, it is done through the bean-configuration file For this, the property to be set with the SDI is declared under the <property> tag in the bean-config file.*

***Constructor Dependency Injection (CDI)****: In this, the DI will be injected with the help of contructors. Now to set the DI as CDI in bean, it is done through the bean-configuration file For this, the property to be set with the CDI is declared under the <constructor-arg> tag in the bean-config file.*

| **Setter DI** | **Constructor DI** |
| --- | --- |
| Poor readability as it adds a lot of boiler plate codes in the application. | Good readability as it is separately present in the code. |
| The bean must include getter and setter methods for the properties. | The bean class must declare a matching constructor with arguments. Otherwise, BeanCreationException will be thrown. |
| Requires addition of @Autowired annotation, above the setter in the code and hence, it increases the coupling between the class and the DI container. | Best in the case of loose coupling with the DI container as it is not even required to add @Autowired annotation in the code.([Implicit constructor injections for single constructor scenarios after spring 4.0](https://spring.io/blog/2016/03/04/core-container-refinements-in-spring-framework-4-3)) |
| Circular dependencies or partial dependencies result with Setter DI because object creation happens before the injections. | No scope for circular or partial dependency because dependencies are resolved before object creation itself. |
| Preferred option when properties are less and mutable objects can be created. | Preferred option when properties on the bean are more and immutable objects (eg: financial processes) are important for application. |