1. What is Spring framework ?

- Spring is a framework of frameworks.
- Developer combine multiple framework to develop spring framework like,
  - EJB
  - Struct
  - Hibernate
- It is used to develop the applications like standalone application , web/ Enterprise application as well as loosely coupled applications.
- It is used to integrate different components of application while development.
- It depends on HAS-A relationship and runtime polymorphism.
- It is used to avoid boilerplate coding from JDBC, Servlet and Hibernate.

NOTE :-

❖ **Loose coupling**

- Dynamically changing method implementation for respective method signature is reffered as loose coupling.
- As well as one object will not depends on another object.

❖ **Tight coupling**

- Providing fixed method implementation for respective method signature is reffered as tight coupling.
- As well as one object will depends on another object.

2. Explain features of spring framework ?

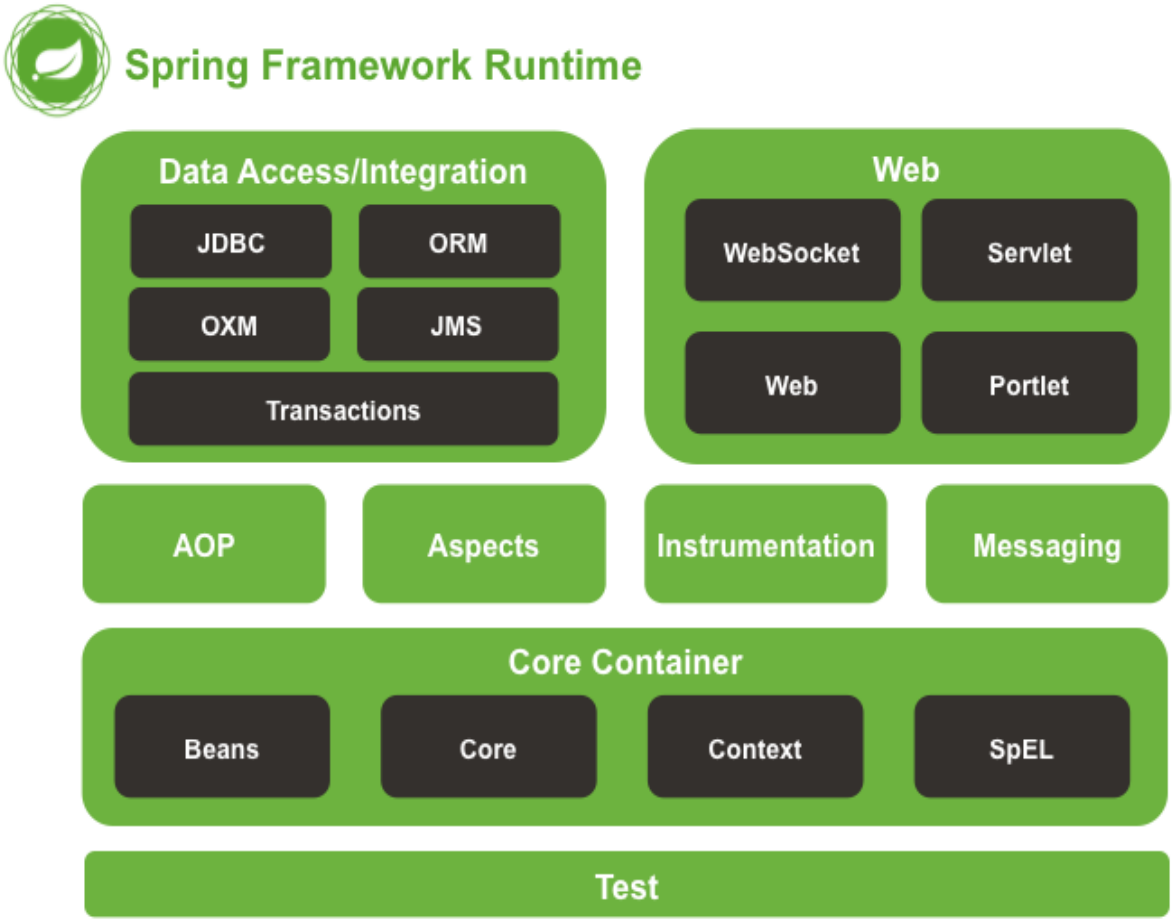
- It is lightweight and open source framework.
- It supports **IOC** (inversion of control) & **DI** (Dependency injection).
- It provides MVC design pattern to build web applications(Model view controller).
- It provide external API like persistence API to work with jdbc and hibernate.

3. What is software engineering principle ?

- It is a set of rules which are used to develop an application for reusability of code as well to maintain the application code.
- Ex. Inversion of control.

- This Software Engineering principles are used in each and every application irrespective with programming languages.

4. Explain Spring Architecture.



1. Core container

- It is a major module of Spring framework , it is used to develop all the application of Spring framework.
  - **a. Core (Spring core)**
    - ➢ It provide features like IOC and DI.
  - **b. Beans (Spring Bean)**
    - ➢ It is used to create object in a Spring framework ,it provides beanfactory with factory design pattern implementation.
  - **c. Context**

- It extends core (Spring core) to provide advance features for an application , it also provides support for third party library.

d. SpEL

- It includes the modules which are used to handle data access and transaction processing in an application.

2. Data Access / Integration

- It is used to work with database application.

a) JDBC (Spring data JDBC)

- It provides abstraction level for JDBC technology which means programmer does not have to write explicit code for communication with database.

b) ORM (Object Reletional Mapping)

- It provides integration level for ORM Framework like hibernate with JPA.

c) OXM (Object XML Mapping)

- It is used to map java objects with XML files.

d) JMS (Java Messeging Service)

- It provides abstraction layer for creating , receving and sending the messege from one layer of application to another.

e) Transaction

- It helps to manage transaction by using classes and interfaces.

3. Web

- It is a container which consist modules to develop web applications.

a) Servlet

- Internally in spring web MVC , spring framework is using Servlet API to develop an web application.

5. What is Spring Core ?

- It is an important module of Spring framework ,to develop an application. It consists two important concepts that are **IOC** & **DI.**
- Without having knowledge of spring core programmer can't develop any application**.**
- But only Spring core knowledge is not enough to develop an application.
- Hence , programmer have to use other modules of Spring framework , like Spring data JDBC , Spring MVC along with spring core.

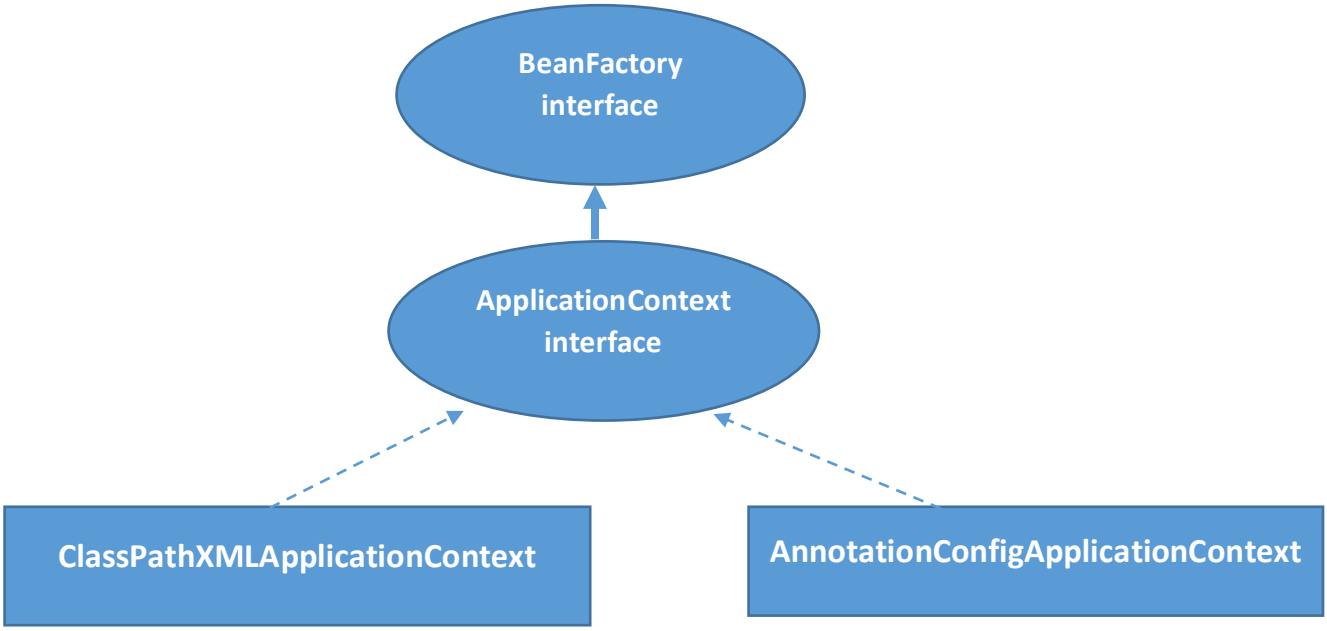
6. What is IOC ?

- **IOC** stands for inversion of control.
- It is a Software enginearing principle.
- It is used to externalize object creation , which means external component will create object for a programmer.
- The main purpose of IOC is , programmer should focus on functionality and business logic of an application.
- The external component ,who will create object for programmer is an **IOC container (Spring container).**
- IOC is responsible for reversing the control of object creation as well as management.
- **Ex.** In traditional approach of java , all the operations releted to objects are performed by the programmer ,but in spring framework because of IOC entire creation of object as well as management taken care by an intelligent program that is IOC Container.

7. What is IOC container / Spring Container.

- It is an intelligent program which will externalize object creation.
- It is also reffered as Spring container.
- It is represented by using two interfaces .
    1. Beanfactory interface.
    2. ApplicationContext Interface.

- BeanFactory interface provide basic features as well as programmer can develop standalone application.
- It supports only XML file configuration metadata.
- ApplicationContext interface extends Beanfactory inteface,it provide advance feature as well as programmers can deveop web application.
- It supports XML file as well as Annotations configuration metadata.
- It has two implementation classes.
    1. ClasspathXMLApplcationContext
    2. AnnotationConfigApplicationContext



- When programmer provide configuration metadata using XMl file then make use of ClassPathXMLApplicationContext implimentaion class.
- When programmer provides configuration metadata by using annotation then make use of AnnotationConfiApplicationContext.
- Programmer have to provide two information for an IOC container.
    1. Discription of object / Bean → class
    2. Configuration metadata → Annotation / XML file

Application context

Annotations / XML file

8. Explain internal working of IOC.

- When programmer launch IOC container, IOC container object get create In a JVM.
- IOC container reads XML file configuration or annotation configuration and it will create an object for a respective class.it will assign name for an object and it will upcast object to object class.
- **Syntax to launch IOC container for XML file :**

```
ApplicationContext context = new
ClassPathXmlApplicationContext("/org/jsp/Assignment/empInfo.xml")
```

- **Syntax for IOC container for annotation :**

```
ApplicationContext context = new
AnnotationConfigApplicationContext(Configuration.Class);
```

9. How to get object from IOC Container ?

- To get object from IOC container programmer are using **"getBean()"** method.
- **getBean** is a **non static** method present in **BeanFactory interface**.
- It is an overloaded method.
- EX.
  getBean("Name of object") → need to perform downcasting because returns object class.

  getBean(ClassName.class) →avoid downcasting but work for only one <bean> tag for respective class.

  getBean("Name of an object" , className.Class) → avoid downcasting & works for multiple objects.

10. Explain Scope of object / Bean.

- It is used to control object creation.

- It also represents visibility of Spring objects.
- IOC container creates an object of respective class based on configuration provided by programmer in XML file and annotations.
- There are four types of scope :

a) singleton

- Bydeafult scope of every object is signleton.
- For singleton scope IOC container create only one object for a respective bean tag.
- This objects get create at the time of launching IOC container.
- At every call of getBean () , IOC container returns same object.

b) Prototype

- If programmer wants to create multiple objects by using one bean tag , then they preffers scope as prototype.
- When programmers call getBean() then IOC container will create an object of respective class.
- At every call of getBean() , IOC container create new object of respective class.
- EX. :
  If programmer called getBean() for 5 times then IOC container will create 5 objects.

c) Request

- If scope of an object is request then IOC container will create object , when user sends a HTTP request.
- At every http request IOC container creates new objects.
- Basically it is used for web Applications.

d) Session

- If scope of an object is session then IOC container creates an object at every session in a web application.
- Ex.

1)to modify scope of an object in XML file.

```
<bean id="s1" class="org.jsp.scope.Sample"
scope="prototype"></bean>
```

2)to modify scope by using annotation make use of @Scope annotation above method name / class name.

@Scope("prototype")

11. Explain XML file configuration ?

- Create a XML file in a package.
- Provide XML schema spring configurations in a file by using beans tag.
- Beans tag represents collections of objects.
- In a beans tag to create an object programmer preffers bean tag along with ID and class attribute.
- Id attribute represents name of an object where as class attribute represents fullyQualifiedClassName.
- Ex.

```
<beans>
    <bean id="s1" class="org.jsp.scope.Sample" scope="prototype"></bean>
</beans>
```

12. Explain Init-method and destroy-method attribute.

- Both the attributes are used with bean tag.
- Itit-method is used to call given method at the time of creation of an object.
- destroy-method is used to call given method at the time of remove objects from the container.
- For both the attribute programmer have to provide method name as a value.
- Ex .

```
<bean id="s1" class="org.jsp.scope.Sample"  destroy-method="destruction" init-method="initialization"> </bean>
```

Note : -

- singleton class means the class for which programmer can create only one object throughts the JVM.
- Singleton scope means programmer can create only one object by using one bean tag for respective IOC container.

13. How to initialize primitive data types and String.

- To inject data into primitive data types and String type of variables , programmer have two approaches.
  1) Constructor injection

2) Setter injection

- When object gets create by using constructor with arguments as well as data will gets inject by using constructor with argument is reffred as constructor injection.
- To perform constructor injection , its mandatory to generate constructor with arguments in a respective class.
- In a XML file, programmer have to use `<constructor-arg>` tag with value attribute within bean tag.
- Ex :

```
<bean id = "a1" class="org.jsp.dependency_injection.Address">
        <constructor-arg value="1"></constructor-arg>
        <constructor-arg value="Hadpsar"></constructor-arg>
</bean>
```

- If constructor has more than one argument then programmer have to follow order of arguments while injecting data.
- No. of constructor arguments = No. of Constructor-arg tag.
- To assign the value without following order of a constructor arguments programmers are using **index** or **name** attribute with constructor-arg tag.
- Always index starts with 0.
- For name attribute,provide arguments name of constructor.
- Ex. :

```
<constructor-arg value="107" name="id"></constructor-arg>

<constructor-arg value="109"   index="0"></constructor-arg>
```

- When object get create by calling no argumented constructor and data will inject by using setter methods is refferd as setter injection.
- Programmer have to create no argumented constructor as well as setters method in a respective class.
- In a XML file programmer have to use property tag along with name and value attribute.
- Ex. :

```
<bean id="aut1" class="org.jsp.dependency_injection.Author">
        <property name="autorId" value="11" ></property>
        <property name="authorName" value="ShreyaSoni"></property>
</bean>
```

14. What is dependency ?

- When one object depends on another object then it refferd as dependency.
- To create a dependency between the object , programmers are using HAS-A relationship.

- Ex : if programmer creates Author class reffernce variable in book class as a global variable then author is a dependency of book class. Book class is refferd as dependent.

15. What is dependency Injection ?

- Injecting dependency object in dependent object is refferd as dependency injection (**DI**).
- It is a design pattern.
- Dependency injection is one way to archive **Invarsion of Control.**
- To archive dependency injection we have 3 ways,
  1) Constructor injection
  2) Setter injection
  3) Field injection
- Injecting a dependency by calling constructor with arguments is refferd as constructor injection.
- To perform constructor injection , programmers are using <Constructor-arg> tag with “ref” attribute.
- For this reffernce attribute assign name of dependency object.
- **Ex :** `<constructor-arg ref="a1"></constructor-arg>`
- Injecting dependency by calling setters method is refferd as setter injection.
- To perform setter injection programmers are using property tag with reference attribute.
- Ex. : `<property name="autohr" ref="aut1"></property>`

16. Diffrence between constructor injection and setter injection.

Constructor injection	Setter injection
Data and dependency injected by using constructor with arguments.	Data and dependency injected by using setters method.
Object created by using constructor with arguments.	Object created by using constructor with no arguments.

Constructor-arg tag is used in configuration file.	Property tag is used in configuration file.
It is recomonded to inject mandatory data and dependencies.	It is recomonded to inject optional data and dependencies.
Programmer cant modify the data.	Programmer can modify the data.
Dependent object will not get create without injecting dependency object.	Dependent object will get create without injecting dependency object.

17. What is Autowiring ?

- Implcitily , injecting a dependency object into dependent object is reffered as autowiring.
- Autowiring is a way to archive dependency injection .
- To perform autowiring no need to write explicit code in xml file or configuration class.
- To archive autowiring in xml file programmers are using “autowire” attribute with bean tag. Whereas to archive autowiring in configuration class, programmers are using “@Autowired annotation.
- Autowiring can be done in two ways.
  1) ByType
  2) ByName
- Ioc container will inject the dependency, by performing setter injection or constructor injection.

18. Explain autowiring ByType ?

- Ioc container will create dependency object then dependent object.
- Ioc container will search dependency object by its type.
- Ioc container will inject dependency by calling setters method.
- If there are multiple objects of same type then Ioc container will throw an exception “NoUniqueBeanDefinationException”.

19. Explain Autowiring ByName ?

- Ioc container will create dependency object then dependent object .

- Ioc container will search dependency by using argument name specified in setter method.
- Dependency object name and setter argument name must be same.
- Ioc container will inject dependency by calling setters method.
- If there are multiple objects of dependency then Ioc container will inject the object which is having same name as per setter argument without throwing an exception.

NOTE :

If Ioc container does not found a respective object while performing byType or byName autowiring then it will inject null value in a dependency.

20. Explain Autowire attribute ?

- Autowire attribute is used to perform autowiring in Xml file.
- It is used with bean tag.
- For autowire attribute,programmer can provide following values
  1) No
  3) Default
  4) ByName
  5) ByType
  6) Constructor
  7) Auto-detect
- Syntax :

```
<bean id="a" class="FQCN" autowire="byType">
```

21. What is Annotation ?

- Annotation is used to represent metadata of variables, methods ,class & interface.
- It is used to provide more information releted to above components for JVM and compiler.
- Programmer have to specify annotation above className , variable name , interface name , Method name.
- Annotations name represented by using pascel case.
- EX. @Override , @webservlet , @Component etc.

22. Explain Annotation Configurations in Spring Core.

- Compare to XML file configuration , Annotation configuration is easy.
- Programmer have to create configuration class , to perform configurations releted to object.
- Denote configuration class with “@configuration” annotation .
- In above configuration class programmer have to create a method to create an object as well as to manage dependency.
- Denote above method with “@Bean” annotation.
- Bydefault object name will be same as method name.
- Ex.

```
@Configuration
public class Config {

	@Bean("b1")
	public Employee getObject()
	{
		return new Employee(11, "Kalayni", 23000, "kg@gmail.com","kg123");
	}
}
```

- To modify scope of an object , programmers are using @scope annotation above the method.
- Ex.

```
@Bean (“d1”)

@Scope (“prototype”)

Public Demo getObject(){

	return new Demo();

}
```

Note :- for configuration class IOC container creates an object ,a s well as IOC container call the methods which are denoted with @Bean annotation , for creation of an object.

23. What is AutoScanning ?

- Implicit creation of an object is refferd as Autoscanning.
- Programmers, no need to write explicit code, to create an Object in XML file as well as in configuration class.

- To perform Autoscanning , make use of **@componentScan** annotation above configuration class Along with **basepackges** attribute
- Denote the classes with @component annotation ,for which programmer wants to create an object.
- Ex.

```
@Component("t1")
//bydefault object name is same of class name
@Scope("prototype")
public class Test {

	public Test() {
		System.out.println("Constructor of Test class...");
	}
}

@Configuration
//Implicit object creation / Auto-Scanning
@ComponentScan(basePackages = "org.jsp.Auto_Scan")
public class Config
{
	public Config() {
		System.out.println("Config class constructor");
	}
}
```

24. Explain internal working of AutoScanning ?

- IOC container create an object of Configuration class.
- It will scan the package which is specified in @componentscan annotation.
- While scanning the package , it will create an object for the classes which are denoted with @component annotation.
- Bydefault name the object will be same as class name.

25. Explain initialization of string & primitive data type by using annotation.

- Programmer can inject data in primitive data type and string type of variables in three ways.
  1) **By using constructor injection**
  2) **By using setter injection**
  3) **By using field injection**
- Programmers are using @Value annotation along with data above variable name or with constructor arguments or above setters method.
- **Ex.**

  @Value(“Kalyani”)

  Private String name;

26. What is @Qualifier annotation ?

- It is used to avoid ambiguity problem for IOC container if there are multiple obejcts of same type.
- Programmers are using **@Qualifier** annotation above variable name along with **@autowired** annotation.
- **Ex .**

```
@Value("Vivo")
Private String brand;
@Autowired
@Qaulifier ("jio")
Private Simcard simcard;
```

27.

28.

29.