**Spring Boot:->** Java Spring Boot is an open-source tool that makes it easier to use Java-based frameworks to create microservices and web apps.

**Microservices:->** Microservices are standalone applications that function together in a unified manner.

When a microservice fails only single function should become unavailable, while rest of the system remains unaffected.

while a microservices architecture is a collection of smaller, independently deployable services.

**Monolithic:->** Monolithic architecture is a singular, large computing network with one code base that couples all of the business concerns together

A monolithic application is built as a single unified unit

**Bean:->** Any object in the Spring framework that we initialize through Spring container is called Spring Bean.

**1.** What is the difference between **@component @bean**?

**Ans:-** While @Component is used for automatic scanning and registering of beans,  
 @Bean is used for explicit manual registration of beans.  
 Another difference is that @Component is used for general classes while   
 @Bean is used for creating a specific instance of a class and providing it to the Spring container.

**2.** **@SpringBootApplication** is a combination of 3 three annotations **(@Configuration, @EnableAutoConfiguration, @ComponentScan**)

**@Configuration:**  
-> The class which contains @Configuration annotation, then the class is enabled for the bean creation  
(In Spring we use XML file for the bean creation)

**@ComponentScan**  
->We don't need to give the bean definition to the configuration file of the stereotype annotations like for (@Controller, @Entity and @Service etc)  
-> We need to give the package location to the @componentscan   
-> So that it will go to that location and scan these stereotype annotations and create the objects of those classes and placed it into the ApplicationContext  
-> & then it sends wherever our program needs these objects.

**@EnableAutoConfiguration**(bcoz of this spring Boot is famous)  
-> This basically download a jar file related to this.  
-> spring.factory this file contains all the classes which are responsible for the AutoConfigurations.  
-> How and when this bean needs to load. it is written inside the above class file.

**4. Annotations:**

**a) @Autowired:-**   
@Autowired annotation is used for automatic dependency injection.  
Spring framework is built on dependency injection and we inject the class dependencies through spring bean configuration file.  
@Bean is used for creating a specific instance of a class and providing it to the Spring container.

**b) @Service:-**   
@Service annotation is used with classes that provide some business functionalities.  
Spring context will autodetect these classes when annotation-based configuration and classpath scanning is used.

**c) @RestController:-**   
@RestController annotation contains @Controller and @ResponseBody annotations.  
This annotation is applied to a class to mark it as a request handler.  
Spring RestController annotation is used to create RESTful web services. using Spring MVC.

**d) @Component:-**   
@Component is an annotation that allows Spring to detect our custom beans automatically.  
Means without writing any explicit code, Spring will: Scan our application for classes annotated with @Component.  
Instantiate them and inject any specified dependencies into them.

**e) @ComponentScan:-**   
@ComponentScan without arguments tells Spring to scan the current package and all of its sub-packages.

**3. Dependencies:**

**a) Spring Web:-** spring-boot-starter-web dependency transitively pulls in all dependencies related to web development.  
 It also reduces the build dependency count.

**b) Data JPA:-** Spring Boot JPA is a Java specification for managing relational data in Java applications.  
 It allows us to access and persist data between Java object/ class and relational database.  
 JPA follows Object-Relation Mapping (ORM).

**(Q):** How does **Spring Boot** works?

**Ans:** **Spring Boot** automatically configures your application based on the dependencies you have added to the project by using annotation.  
 The entry point of the spring boot application is the class that contains @SpringBootApplication annotation and the main method.  
 Spring Boot automatically scans all the components included in the project by using @ComponentScan annotation.

**(Q):** What does the **@SpringBootApplication** annotation do internally?

**Ans:** The **@SpringBootApplication** annotation is equivalent to using **@Configuration, @EnableAutoConfiguration**,   
 and **@ComponentScan** with their default attributes.

**Spring Boot** enables the developer to use a single annotation instead of using multiple. But, as we know, Spring provided loosely coupled features that we can use for each annotation as per our project needs

**Spring Boot Actuator:-**   
 Spring Boot provides actuator to monitor and manage our application.  
 Actuator is a tool which has HTTP endpoints. when application is pushed to production,  
 we can choose to manage and monitor our application using HTTP endpoints.

**5.** How to connect Spring Boot to the database using **JPA**?

**Ans:-** Spring Boot provides spring-boot-starter-data-jpa starter to connect Spring application with relational database.  
 You can use it into project POM (Project Object Model) file.

**6.** How to connect Spring Boot application to database using **JDBC**?

**Ans:-** Spring Boot provides starter and libraries for connecting to our application with **JDBC**.

**7.** **Spring** Vs **Spring Boot**?

**Spring:->** Spring is a web application framework based on Java. It provides tools and libraries to create a complete customized web application.  
 **Spring Boot:->** Whereas Spring Boot is a spring module which is used to create spring application project that can just run.

**8.** **Features** of **Spring Boot** that make it different?

a) Creates stand-alone spring application with minimal configuration needed.  
b) It has embedded tomcat, jetty which makes it just code and run the application.  
c) Provide production-ready features such as metrics, health checks, and externalized configuration. &  
d) No requirement for XML configuration.

**9.** What is **dependency Injection**?

**Ans:-** The process of injecting dependent bean objects into target bean objects is called **dependency injection**.

**Setter Injection:** The IOC container will inject the dependent bean object into the target bean object by calling the setter method.

Constructor Injection: The IOC container will inject the dependent bean object into the target bean object by calling the target bean constructor.

**Field Injection:** The IOC container will inject the dependent bean object into the target bean object by Reflection API.

**10.** What is an **IOC** **container**?

**Ans:-** **IOC** **container** is a framework for implementing automatic dependency injection.  
 It manages object creation and its life-time and also injects dependencies into the class.

**11. What is JDBC?**

JDBC is a Java API to connect and execute the query with the database.   
JDBC API uses JDBC drivers which is written in Java.

**12. What is JPA? (Java Persistence API)**

JPA is just a specification that facilitates object-relational mapping to manage relational data in Java applications. It provides a platform to work directly with objects instead of using SQL statements.

**13. What is Hibernate?**

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

**14. Native Query example?**

@Query(nativeQuery = true, value = "SELECT \* FROM employee where employeeId=:eId")   
Optional<Address> findAddressByEmployeeId(@Param("eId ") int employeeId);

(Where Address is the entity class)

**15. String Buffer? Its methods and features?**

StringBuffer is a class in Java that represents a mutable sequence of characters. It provides an alternative to the immutable String class, allowing you to modify the contents of a string without creating a new object every time.  
  
 The append() method is used to add characters, strings, or other objects to the end of the buffer.

 The insert() method is used to insert characters, strings, or other objects at a specified position in the buffer.

**16. How to create an Immutable class?**

Immutable class in java means that once an object is created, we cannot change its content.

* The class must be declared as final so that child classes can’t be created.
* Data members in the class must be declared private so that direct access is not allowed.
* Data members in the class must be declared as final so that we can’t change the value of it after object creation.
* A parameterized constructor should initialize all the fields performing a deep copy so that data members can’t be modified with an object reference.
* Deep Copy of objects should be performed in the getter methods to return a copy rather than returning the actual object reference

**17. Compile-time and Runtime Exception?**

A compile-time error generally refers to the errors that correspond to the syntax.   
A runtime error refers to the error that we encounter during the code execution during runtime.  
  
  
**18. Annotation means?**

Annotations are used to provide **supplemental** information about a program.   
It is not a part of the application that we develop.   
It does not have a direct effect on the operation of the code.

**19. Comperator?**

A comparison function, which imposes a total ordering on some collection of objects.  
Java Comparator interface is used to order the objects of a user-defined class.

**20. Constraints?**

SQL constraints are used to specify rules for data in a table.   
Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.  
example: column1 datatype *constraint*.  
  
**21. Concurrent Modification Exception?**

ConcurrentModificationException is a very common exception when working with Java collection classes. Java Collection classes are fail-fast, which means **if the Collection will be changed while some thread is traversing over it using iterator, the iterator.** **next() will throw ConcurrentModificationException**.  
  
**22. @Transactional?**

@Transactional annotation is used to manage transactions in a Spring boot application and used to define a scope of transaction.

**23. API?**

An application programming interface (API) is code that enables two software programs to communicate.  
An API defines how a developer should request services from an operating system (OS) or other application, and expose data within different contexts and across multiple channels.

**24. Encryption?**

Encryption is converting plaintext into ciphertext, which can only be accessed by authorized users with the correct cryptographic key, encryption is a computer process

**25. Decryption?**

The process of returning a meaningless communication (Ciphertext) to its original format is known as decryption (Plaintext).

**26. File Handling?**

In Java, with the help of File Class, we can work with files. This File Class is inside the java.io package. The File class can be used by creating an object of the class and then specifying the name of the file.

**27. REST API?**

(REST) is an architectural style that defines a set of constraints to be used for creating web services.