1 – IOC and DI means:

- IOC: App'in yaşam döngüsü boyunca birbirine az bağımlı olan nesneler oluşturmayı hedefleyen prensiptir. IOC kullanan sınıfa bir interface inject edildiğinde, ilgili interface metotları kullanılabilir olur. Böylece IOC kullanan sınıf sadece kullanacağı metotları bilir, sınıf içerisinde daha fazla metot olsa bile interface'de belirtilen metotlara erişebilecektir.
- DI: Bir sınıfın bağımlılıklarından kurtulmasını amaçlayan ve o nesneyi olabildiğince bağımsızlaştıran bir programlama prensibidir. Bir sınıfın bağımlı olduğu nesneden bağımsız hareket edebilmesini sağlayabilir ve kod üzerinde olası geliştirmelere karşın değişiklik yapma ihtiyacını ortadan kaldırır.

2 – Spring Bean Scopes?

- Singleton: bir bean default olarak singleton'dur. Singleton beani o bean'in tek bir örneğini oluşturur. Yani tekil nesneler oluşturulması için kullanılmaktadır.
- Prototype: Singletondan farklı olarak gerekli referanslar için birer tane yeni bean örneği oluşturur.
- Request: Web uygulamalarında HTTP request geçerli olduğu sürece geçerli olacak bir nesne oluşturur.
- Session: Web uygulamarında HTTP session geçerli olduğu sürece geçerli olacak bir nesne oluşturur.
- globalSession: Tüm HTTP sessionları için bir tane nesne oluşturur.

3 – What does @SpringBootApplication do?

@SpringBootApplication ifadesi @EnableAutoConfiguration, @ComponentScan ve @SpringBootConfiguration ifadelerinin birleşimidir.

- @SpringBootConfiguration: @Configuration ifadesini kullanarak sınıfın bir bean sınıfı olduğunu belirtir.
- @ComponentScan: parametre verilmeyip varsayılan olarak kullanıldığı sınıfa ait paketinin altındaki bean sınıflarını tarar ve kaydeder.

4 – Why Spring Boot over Spring? , What is the primary difference between Spring and Spring Boot?

Spring	SpringBoot
Kurumsal java uygulamaları geliştirmek için kullanılır.	Rest api oluşturmak için kullanılır.
Kurumsal uygulama geliştirmeyi basitleştirmeyi amaçlar.	Kod uzunluğunu kısaltma ve kolay web uygulama geliştirmeyi amaçlar.
Ana özelliği DI'dır.	Ana özelliği otomatik yapılandırmadır.
Sunucuya manuel kurmak gerekir	Tomcat ve Jetty gibi gömülü sunucu sağlar.
Bellek veritabanı sağlamaz	H2 gibi bellek veritabanını destekler

5 – What is Singleton and where to use it?

Bir sınıfın tek bir örneğini almak için kullanılır. Amaç oluşturulan nesneye global erişim noktası sağlamaktır. Sistem çalıştığı süre boyunca ikinci bir örnek oluşturmaz. Böylelikle istenilen nesnenin tek bir defa oluşturulması garanti altına alınır. Singleton nesneler ilk çağrıldıklarında bir kere oluşturulurlar ve sonraki istekler bu nesne üzerinden karşılanır.

6 – Explain @RestController annotation in Sprint boot?

@RestController, @Controller ve @ResponseBody'nin birleşiminden oluşur.

- @Controller: Sınıfın, dışarıdan gelen requestleri yakalaması gerek bir sınıf olduğunu belirtir.
- @ResponseBody: Eldeki veriyi JSON olarak serialize edip geri gönderir.

8 – Why to use VCS ?

VCS olmadan karmaşık kod geliştirme imkansızdır. Sürüm kontrolü, kaynak kodundaki değişiklikleri izleme ve kontrol etmek için kullanılır. Kod tabanın bütünlüğünü sağlamak için gerekli bir araçtır.

- Codebasein kolay değiştirilebilmesi
- Hataları geri alma
- Ekip çalışması

9 – What are SOLID Principles ? Give sample usages in Java?

SOLID yazılım prensipleri diye adlandırılan ve dünya üzerinde, OOP nesne tabanlı yazılım geliştirirken kullanılan standartlaştırılmış 5 önemli tasarım ilkesi vardır.

Bu ilkelerin amacı, yazılım tasarımlarını daha anlaşılır, bakımı daha kolay ve genişletilmesi daha kolay hale getirmektir.

- Single responsibility principle (SRP): Her sınıfın veya metotun tek bir işlevi olmasıdır.
- Open Closed Principle (OCP): Her sınıfın geliştirmeye açık fakat değişime kapalı olmasıdır
- Liskov Substitution Principle (LSP): Türetilen sınıflar, türeyen sınıfların tüm özelliklerini kullanmak zorundadır.
- Interface Segregation Principle (ISP): Her interface'in belirli bir amacı olmalıdır. Tüm metotları kapsayan tek bir interface kullanmak yerine, her biri ayrı metot gruplarına hizmet veren birkaç interface tercih edilmelidir.
- Dependency Inversion Principle (DIP): Sınıflar arası bağımlılıklar olabildiğince az olmalıdır özellikle üst seviye sınıflar alt seviye sınıflara bağımlı olmamalıdır.

10 -What is RAD model?

RAD Modeli veya Hızlı Uygulama Geliştirme modeli, herhangi bir özel planlama olmaksızın prototiplemeye dayalı bir yazılım geliştirme sürecidir. RAD modelinde, planlamaya daha az dikkat edilir ve geliştirme görevlerine daha fazla öncelik verilir. Kısa sürede yazılım geliştirmeyi hedefler.

SDLC RAD modellemesinin aşağıdaki aşamaları vardır

- İş modelleme
- Veri modelleme
- Süreç modelleme
- Uygulama üretimi
- Test ve ciro

RAD Modeli veya Hızlı Uygulama Geliştirme modeli, herhangi bir özel planlama olmaksızın prototiplemeye dayalı bir yazılım geliştirme sürecidir. RAD modelinde, planlamaya daha az dikkat edilir ve geliştirme görevlerine daha fazla öncelik verilir. Kısa sürede yazılım geliştirmeyi hedefler.

11 -What is Spring Boot starter? How is it useful?

Starterlar kısaca uygulamaya ekleyebileceğimiz bir dizi bağımlılık tanımlayıcısıdır. Kullanmak istediğimiz teknolojiler için arama yapıp teker teker bağımlılık ekleme zahmetinden kurtarır. Starterlar sayesinde ihtiyacımız olan Spring ve ilgili teknolojileri kolayca uygulamamıza ekleyebiliriz.

12 – What are the Spring Boot Annotations?

- @Bean: Bir metodun Spring tarafından yönetilen bir Bean ürettiğini belirtir
- @Service: Belirtilen sınıfın bir servis sınıfı olduğunu belirtir.
- @Repository: Veritabanı işlemlerini gerçekleştirme yeteneği olan yapıldığı repository sınıfını belirtir.
- @Configuration: Bean tanımlamaları gibi tanımlamalar için bir Bean sınıfı olduğunu belirtir
- @Controller: Requestleri yakalayabilme yeteneği olan bir web controller sınıfını belirtir.
- @RequestMapping: Controller sınıfının handle ettiği HTTP Requestlerin path eşleştirmesini yapar
- @Autowired: Constructor, Değişken yada setter metodlar için dependency injection işlemi gerçekleştirir
- @SpringBootApplication: Spring Boot autoconfiguration ve component taramasını aktif eder.

13 – What is Spring Boot dependency management?

Spring Boot, dependencyleri ve yapılandırmayı otomatik olarak yönetir. Spring Boot'un her sürümü, desteklediği dependencylerin bir listesini sağlar. Dependencyler listesi, Maven ile kullanılabilecek Malzeme Listelerinin bir parçası olarak mevcuttur. Bu yüzden konfigürasyonumuzdaki dependencylerin versiyonunu belirtmemize gerek yok. Spring Boot kendini yönetir. Spring Boot sürümünü güncellediğimizde Spring Boot, tüm bağımlılıkları tutarlı bir şekilde otomatik olarak yükseltir.

14 - What is Spring Boot Actuator?

Spring Boot Actuator, Spring Boot Framework'ün bir alt projesidir. Çalışan herhangi bir uygulama hakkında operasyonel bilgileri ortaya çıkarmak için HTTP uç noktalarını kullanır. Bu kitaplığı kullanmanın temel yararı, üretime hazır uygulamalardan sistem durumu ve izleme ölçümleri almamızdır. Ayrıca, metriclerin toplanması, trafiğin anlaşılması veya veri tabanının durumunun bilinmesi Actuator ile son derece kolay hale gelir.