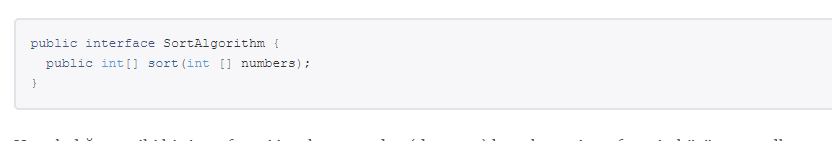
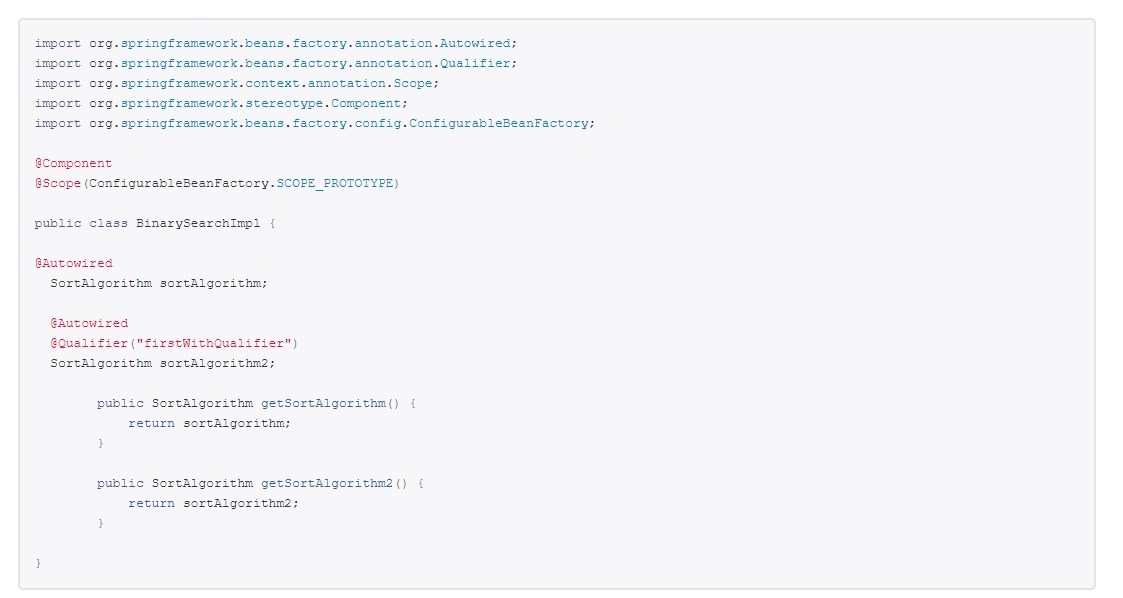
Bu yazıda Spring Boot Framework ile dependecy injection yani bağımlılık yönetiminin nasıl yapıldığına ve @Autowired, @Primary , @Qualifier gibi anotasyonlarının kullanımına değineceğiz.

@Component anotasyonu class’ın bir spring boot class’ı olduğunu belirtir.









İki class arasındaki belirgin olan farklılık ise görebileceğiniz gibi BubbleSortAlgorithm clasında @Primary, QuickShort classında @Qualifier anotasyonlarının kullanılmasıdır. Eğer bu anotasyonları kullanmamış olsaydık SortAlgorithm interfacesini implement eden 2 farklı class olduğu için SortAlgorithm interface’inden bir instance oluşturulurken hata vericekti. BubbleSortAlgorithm

class’ında @Primary anotasyonunu kullanarak interface’si implement eden birden fazla sınıf olsa bile BubbleSortAlgorithm class’ından bir instance oluşturulmasını sağlamış olduk. Fakat özel bir atama ile belirli bir sınıftan instance oluşturmak için @Qualifier anotasyonu kullanılır. Burada main class’sının üstüne koyulan @SpringBootApplication anotasyonu bu projenin bir spring boot projesi olduğunu ve bu class’ın olduğu dizinde ve alt dizinlerde spring boot class’larının aranacağını belirtir. public static void main methodu aldında SpringApplication.run(BasicsApplication.class, args) direk çağırılarak kullanılır.

@PostConstruct

public void init() throws Exception {

System.out.println("Inıt Metot PostConstruct : " + mesaj);

}

**Bu sınıfların içine verildiğinde ilk bu çalışır.**

[https://javainit.wordpress.com/2015/09/16/component-service-repository-ve-contro ller-annotationlarina-dair-notlar/](https://javainit.wordpress.com/2015/09/16/component-service-repository-ve-contro%20ller-annotationlarina-dair-notlar/)

<https://www.danvega.dev/blog/2017/03/27/spring-stereotype-annotations/>

<https://malikmasis.blogspot.com/2015/09/spring-framework.html>

https://www.yazilimgelistiricileri.com/spring-boot-ile-rest-servis-uygulama-ornegi/