

Creational(Yaratımsal) Tasarım Desenleri

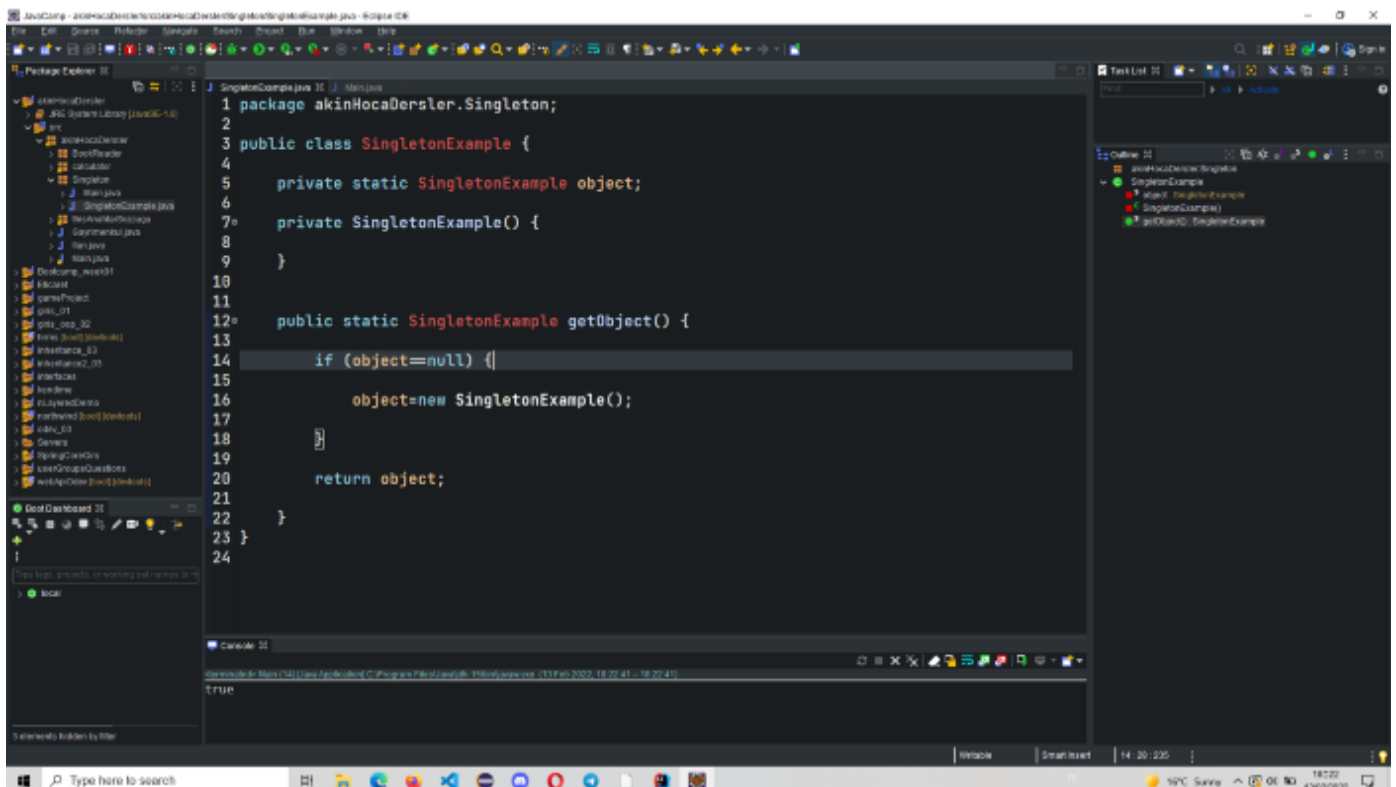
Yaratımsal Desenleri kodun esnekliğini ve tekrar kullanılabilirliğini arttıran nesne oluşturma mekanizmaları sunarlar.

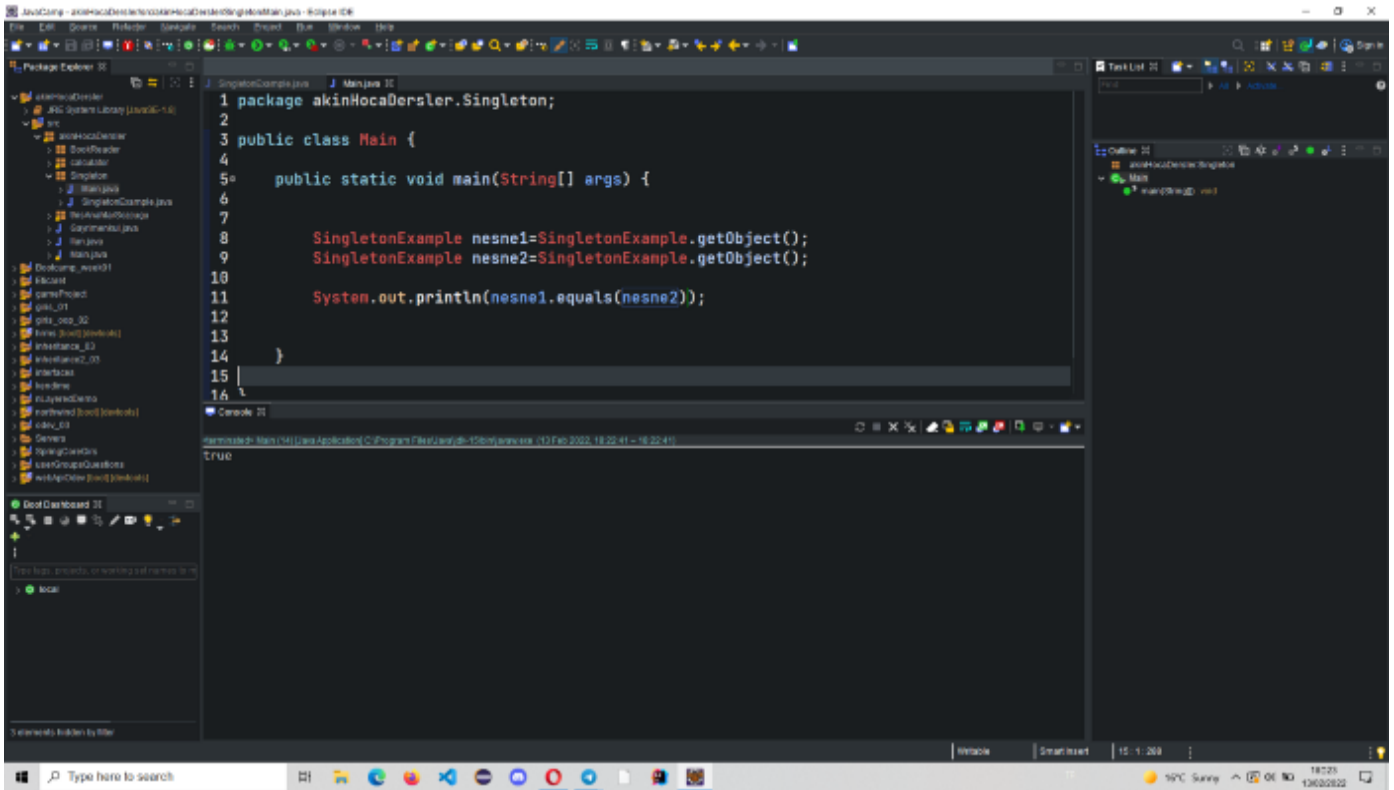
1-Singleton Tasarım deseni

1-Singleton

Bu tasarım deseni bir classtan sadece bir tane instance yaratılmasını sağlar. Yani tekrar tekrar kullanılmak için oluşturulur. Daha öncesinde bir instance oluşmamışsa yeni bir tane üretilir. Spring bunu IoC container içerisinde gerçekleştiriyor yani bu işi Spring ele alıyor ve kontrol ediyor.

- Yapıcı metodu private yapılmalıdır.
- Static bir metot tanımlanmalıdır ve nesne boşsa yeni bir tane üretilir



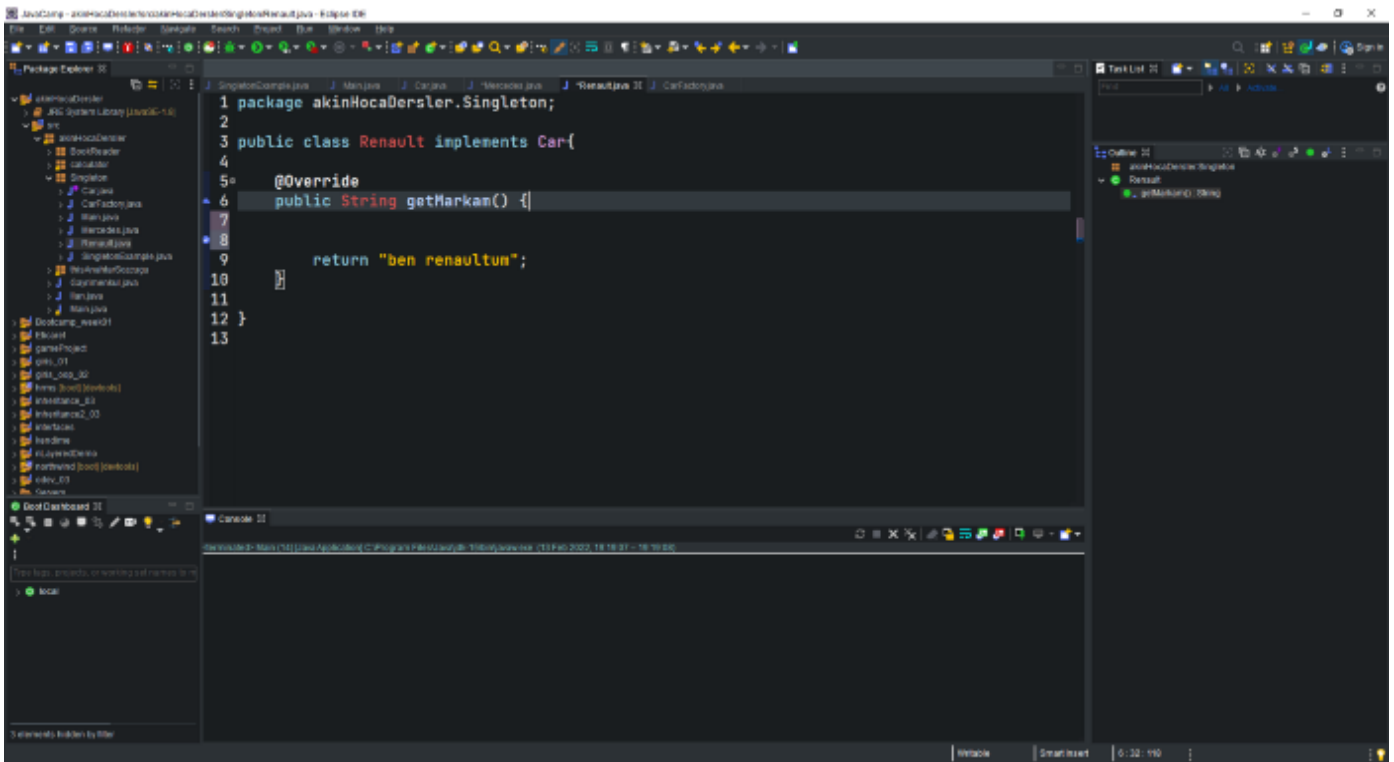


Burada görüldüğü üzere nesneler birbirine eşit olduğu görülmektedir. Çünkü yeni bir nesne üretimi olmadı.

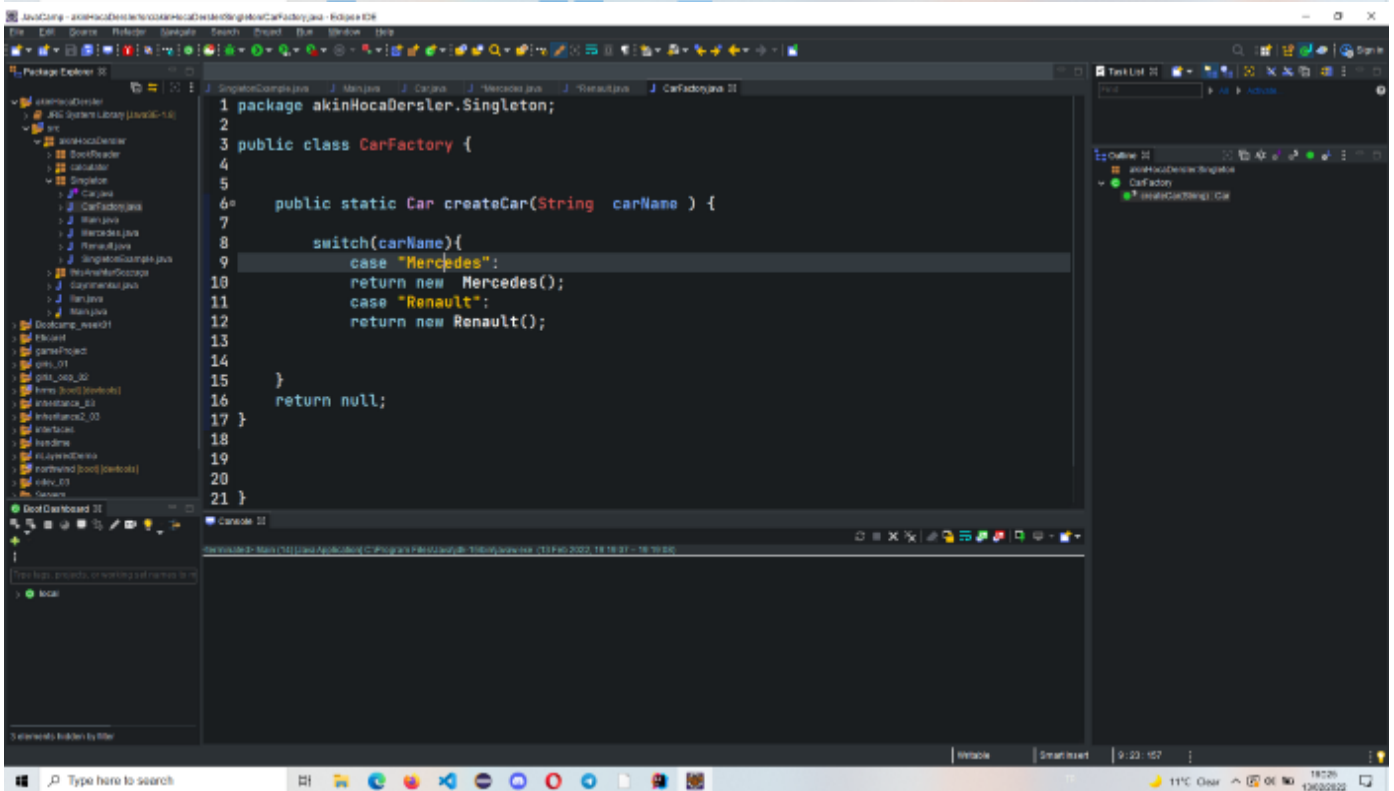
2-Factory(Fabrika)

Fabrika tasarım deseninde nesne üretirken bunu soyutlama yöntemiyle bir üretim sınıfındaki üretim metoduna verme işlemidir. Birbirine benzeyen sınıfları her seferinde new anahtar kelimesini kullanmayalım mantığıyla tasarlanmıştır.

Bir Car interface'miz bulunmaktadır.

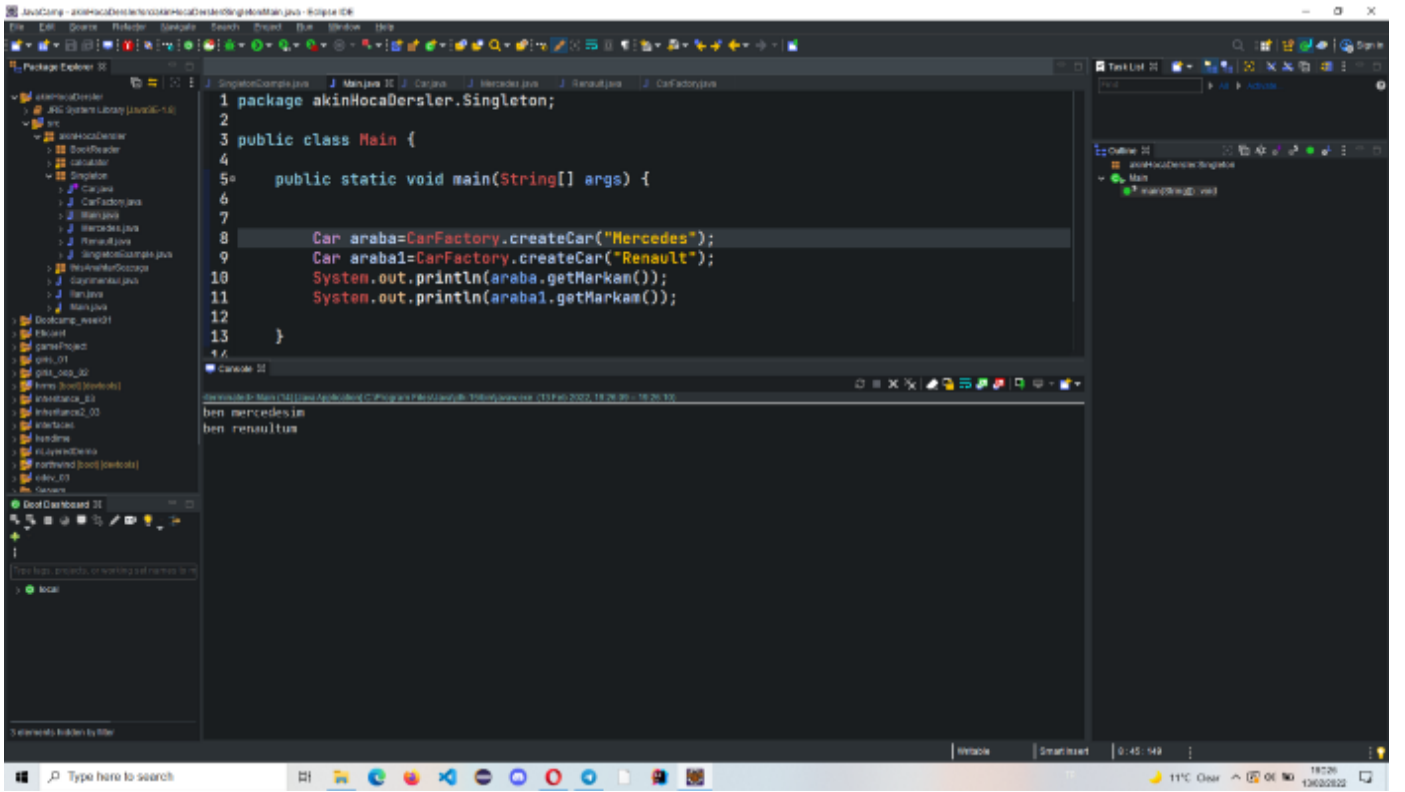


```
1 package akinHocaDersler.Singleton;  
2  
3 public class Renault implements Car{  
4  
5     @Override  
6     public String getMarkan() {  
7  
8  
9         return "ben renaultun";  
10  
11  
12 }  
13
```



```
1 package akinHocaDersler.Singleton;  
2  
3 public class CarFactory {  
4  
5  
6     public static Car createCar(String carName ) {  
7  
8         switch(carName){  
9             case "Mercedes":  
10                 return new Mercedes();  
11             case "Renault":  
12                 return new Renault();  
13  
14  
15         }  
16         return null;  
17 }  
18  
19  
20  
21 }
```

Verilen tipe göre nesne üretme işlemi yapıyor.



Görüldüğü üzere verilen araba ismine göre arabalar üretilmektedir.

Burada soyutlama işleminden yararlanıldı.

3-Builder

Bu tasarım deseni nesne yaratımlarımızda oldukça önemlidir. Sebebi ise bir nesne üretirken sınıfın yapıcı metodunda fieldların sayısı artabilir veya farklı fieldları kullanarak kombinasyonlar yapılabilir bunun önüne geçebilmek için anasınıfımızdaki yapıcı metot içerisine bir tip verilir ve kontrol bu sınıf üzerinden gerçekleşir. İç içe classlardan yararlandık burada.

```

3 public class Item {
4
5     private String name;
6     private int number;
7     private String color;
8
9     private Item(BuilderPattern builder) {
10
11         this.name=builder.name;
12         this.number=builder.number;
13         this.color=builder.color;
14     }
15
16     public String getName() {
17         return name;
18     }
19
20     public int getNumber() {
21         return number;
22     }
23
24     public String getColor() {
25         return color;
26     }
27
28
29     public static class BuilderPattern {
30
31         private String name;
32         private int number;
33         private String color;
34
35         public BuilderPattern() {

```

Görüldüğü üzere zorunlu alanlar ve null değerler göremeyeceğimiz bir tasarım gördük.