Adaptör(Adapter) Tasarım Deseni

Nedir?

Adapter tasarım deseni, **structural** tasarım desenlerinden biridir. Bu tasarım deseni, birbiriyle **ilişkili olmayan** interface'lerin birlikte çalışmasını sağlar. Bu işlemi ise, bir sınıfın interface'ini diğer bir interface'e **dönüştürerek** yapar.

Adapter tasarım deseni ismini gerçek hayattaki adaptörlerden almıştır. Örneğin, telefon şarz cihazı bir adaptördür. 240V'luk gerilimi 5V'a dönüştürür. Mobil şarz cihazı, mobil şarz soketi ile duvar soketi arasında bir adaptör görevi görür.

Ne zaman Kullanılır?

Farklı interface'lere sahip sınıfların birbiriyle çalışabilmesini sağlamak amacıyla kullanılır. Örnek vermek gerekirse, bir XML dosyasının Document Object **Model** interface'ini, bir ağaç yapısına dönüştürülmesi için kullanılabilir.

Nasıl Kullanılır?

Var olan sistemin interface'i, target interface olarak adlandırılır. Bu interface'i implement edecek bir **Adapter** sınıfı yaratılır. **Adapter** sınıfında, **Adaptee** interface türünden bir **sınıf değişkeni** bulunur. Son olarak client sınıfı **Adapter** sınıfı nesnesi ve **Adaptee** nesnesini yaratır.

Daha iyi anlamak için şöyle bir **örnek** verelim: Matematik işlemleri yapan bir modülümüz olsun. Bu modüle String işlemleri yapan bir modül eklemek istiyoruz. String modülü içerisinde spesifik işlemleri yapan sınıflar bulunacak. Bu işlemi yaparken var olan modülün değiştirilmemesi gerekir. Örneğimizdeki var olan modülü **Target** interface temsil etmektedir. String modülünü **Adaptee** interface'i, bu ikisi arasındaki ilişkiyi ise **Adapter** sınıfı sağlamaktadır. String modülü içerisindeki sınıflar ise **Adaptee** interface implement eden sınıflardır.

Faydaları Nedir?

1. Birbiriyle **ilişkili olmayan** interface'lerin **birlikte** çalışmasını sağlar.
2. Kodların **yeniden** yazılması **engeller.**
3. Var olan modül(ler) değiştirilmeden sisteme yeni modüller eklenebilir.

Gerekenler

Türü interface olan **Target** ve **Adaptee** sınıfları. **Adaptee** interface'i adapte olması gereken sınıfı veya sınıfları temsil eder.
En az bir tane alt **Adaptee** sınıfı
Target interface'i implement edecek ve **Adaptee** interce türünden sınıf değişkeni barındıran **Adapter** sınıfı
Target interce'i kullanan **Client** sınıfı

Örnek Kullanım Alanları

1. java.util.Arrays#asList()
2. InputStreamReader
3. OutputStreamWriter

Örnek Uygulama

Target interface

```
1   /**
2     * Target interface
3     */
4   public interface Duck {
5        public void quack();
6        public void fly();
7   }
```

Adaptee interface

```java
/**
  * Adaptee interface
  */
public interface Turkey {
     public void gobble();
     public void fly();
}
```

Somut Adaptee Sınıfı

```java
/**
  * Somut Adaptee sinifi
  */
public class WildTurkey implements Turkey {
     public void gobble() {
         System.out.println("Gobble gobble");
     }

     public void fly() {
         System.out.println("I'm flying a short distance");
     }
}
```

Adapter Sınıfı

```java
/**
  * Adapter sinifi
  */
public class TurkeyAdapter implements Duck {
     Turkey turkey;

     public TurkeyAdapter(Turkey turkey) {
         this.turkey= turkey;
     }

     public void quack() {
         turkey.gobble();
     }

     public void fly() {
         turkey.fly();
     }
}
```

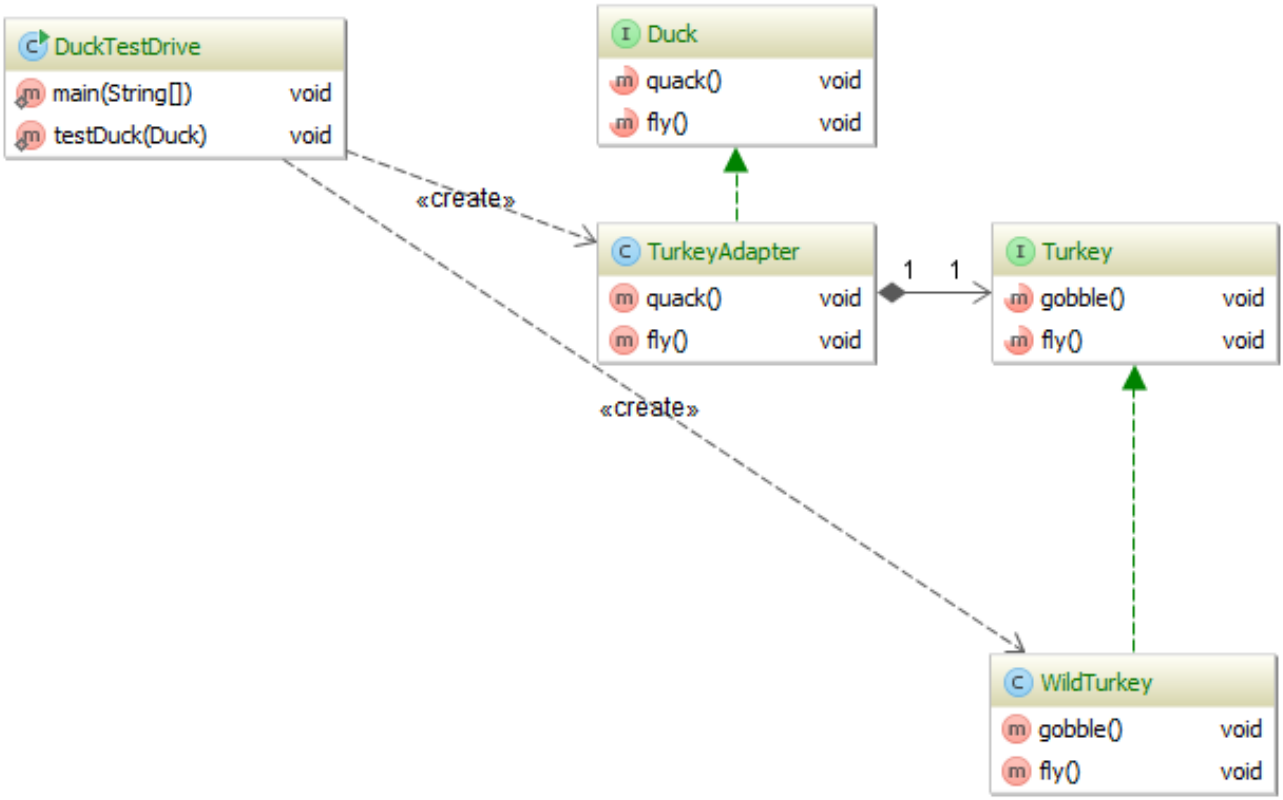
Client Sınıfı

```java
/**
  * Test sinifi
  */
public class DuckTestDrive {
     public static void main(String[] args) {
         WildTurkey turkey = new WildTurkey();
         Duck turkeyAdapter = new TurkeyAdapter(turkey);

         System.out.println("The Turkey says...");
         turkey.gobble();
         turkey.fly();

         System.out.println("\nThe TurkeyAdapter says...");
         testDuck(turkeyAdapter);
     }
```

```
16        static void testDuck(Duck duck) {
17            duck.quack();
18            duck.fly();
19        }
20    }
```

Uygulamada **Duck(Ördek)** sınıfı ile **Turkey(Hindi)** sınıfı arasında bir adaptör(**TurkeyAdapter** sınıfı) bulunmaktadır. **TurkeyAdapter** sınıfı, **Turkey** interface'ini **Duck** interface'ine adapte etmek için bir dönüştürücü görevi görmüştür. **Target** yani **hedef** interface(Örneğimizdeki **Duck** interface'i) uygulamamızda var olan interface anlamına gelir ve bu interface değiştirilmez. Entegre edilen adaptee yani adapte(Örneğimizdeki **Turkey** interface'i) interface'idir. **Turkey** interface ile **Duck** interface arasındaki entegrasyon işini ise **Adapter** sınıfı(Örneğimizdeki **TurkeyAdapter** sınıfı) yapar.

Uygulamamızın UML Diagramı



Adapter Tasarım Deseni'nin Şematik Gösterimi