Bölüm 20

20 Arayüzler

Arayüz bir sınıf değildir. Arayüz yazmak sınf yazmaya çok benzer ancak ikisi birbirinden farklı konseptlerdir. Sınıf bir nesnenin özelliklerini ve davranışlarını belirtirken; bir arayüz sınıfların barındırdığı davranışları içerir.

Arayüz soyut metotların oluşturduğu bir koleksyondur. Bir sınıf arayüzü çalıştırırken arayüzün sahip olduğu soyut metotları da miras alır.

Bütün arayüz metotları sınıflarda tanımlanmalıdır.

Arayüzler sınıflarla şu şekilde benzerlik gösterir:

- Arayüz sınırsız sayıda metot içerebilir.
- Arayüzler .java uzantılı dosyalara yazılır ve dosya ile arayüz ismi aynı olmalıdır..
- Arayüzün bitkodları .class dosyasının içinde yer alır.
- Arayüzler paketler halinde bulunur.

Ancak bir arayüz bir sınıftan şu yönleriyle ayrılmaktadır.

- **Arayüzün instance'I alınamaz**
- **Arayüz constructor metot barındırmaz.**
- **Arayüzün içindeki bütün metotlar soyuttur.**
- Arayüzler instance alanları bulunduramazlar. Tek bulundurabilecekleri alanlar da hem static hem de final olarak tanımlanmalıdır.
- **Arayüzler sınıflara genişletilemez, sınıflara uygulanırlar.**
- Bir arayüz başka arayüzlerle genişletilebilir

201. Arayüzlerin Tanımlanması

Arayüz oluşturmak için interface kelimesi kullanılır. Aşağıdaki örnekte bir tanımlama bulunmaktadır.

Örnek:

Kapsüllemeyi anlatan örneği inceleyelim.

```
public interface NameOfInterface
{
   //Any number of final, static fields

   //Any number of abstract method declarations\
}
```

--Arayüzler aşağıdaki özelliklere sahiptir.

- Arayüzler kendiliğinden soyuttur. Tanımlama yaparken abstract kelimesini kullanmaya gerek yoktur.
- Arayüzün içindeki metotlar da soyut olduğundan abstract kelimesine gerek yoktur.
- Arayüzün içindeki bütün metotlar tam olarak publictir.

Örnek:

```
/* File name : Animal.java */

interface Animal {

   public void eat();
   public void travel();
}
```

20.2 Arayüzlerin Uygulanması

Bir sınıfın arayüzü çalışıtmasını; arayüzün bütün davranışlarını yapmayı kabul ettiği bir kontratı imzalamak olarak düşünebilirsiniz.Eğer sınıf arayüzün bütün davranışlarını gerçekleştirmiyorsa, kendini soyut olarak tanımlamalıdır.

Bir sınıf arayüzü uygulamak için implements kelimesini kullanır. Implements kelimesi sınıf tanımlanmasında kullanılır ve devamında extends kısmı gelir.

```java
/* File name : MammalInt.java */

public class MammalInt implements Animal{

   public void eat(){
      System.out.println("Mammal eats");
   }

   public void travel(){
      System.out.println("Mammal travels");
   }

   public int noOfLegs(){
      return 0;
   }

   public static void main(String args[]){
      MammalInt m = new MammalInt();
      m.eat();
      m.travel();
   }
}
```

Aşağıdaki sonuç oluşacaktır.

```
Mammal eats
Mammal travels
```

Bir arayüz yazarken uyulması gereken kurallar şunlardır.

- **Bir sınıf birden fazla arayüz uygulayabilir.**
- **Bir sınıf sadece bir sınıfa genişletilebilir ancak bir çok arayüze genişletilebilir.**
- Bir arayüz; sınıfların yaptığı gibi diğer arayüzlere genişletilebilir

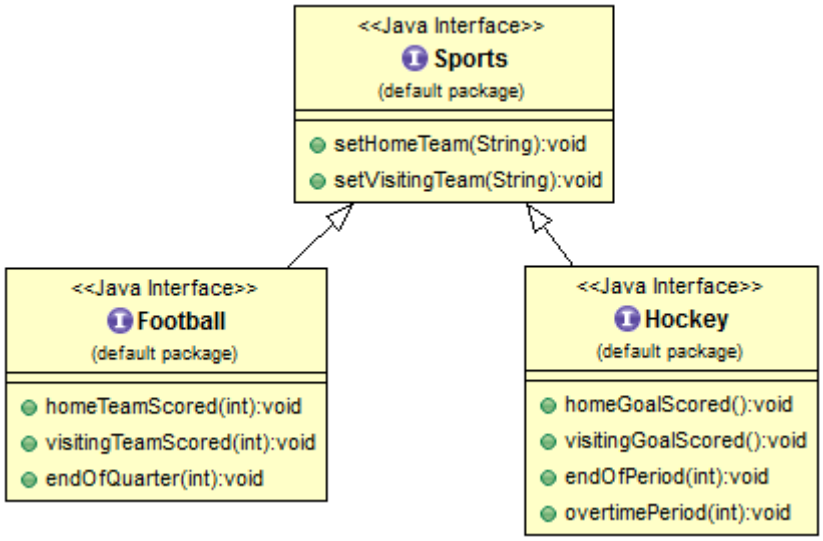
Arayüzlerin Genişletilmesi

Bir arayüz; sınıfların yaptığı gibi diğer arayüzlere genişletilebilir. Extends keyword misaslandırmak için kullanılır. Kelimenin kullanıldığı arayüz ana arayüze doğru genişler. Aşağıdaki Hockey ve Football arayüzleri Sports arayüzünden genişletilmiştir.

```java
//Filename: Sports.java

public interface Sports
{
   public void setHomeTeam(String name);
   public void setVisitingTeam(String name);
}
```

```java
//Filename: Football.java

public interface Football extends Sports
{
   public void homeTeamScored(int points);
   public void visitingTeamScored(int points);
   public void endOfQuarter(int quarter);
}
```

```java
//Filename: Hockey.java

public interface Hockey extends Sports
{
   public void homeGoalScored();
   public void visitingGoalScored();
   public void endOfPeriod(int period);
   public void overtimePeriod(int ot);
}
```



Hockey arayüzünde 4 adet metot vardır ancak iki tanesini Sports'dan miras alır,böylelikle Hockey'den genişletilen sınıf bütün 6 metodu da almış olur. Aynı şekilde Football arayüzünü alan bir sınıf 3 metodu Football'dan ikisini de Sports arayüzünden alır.

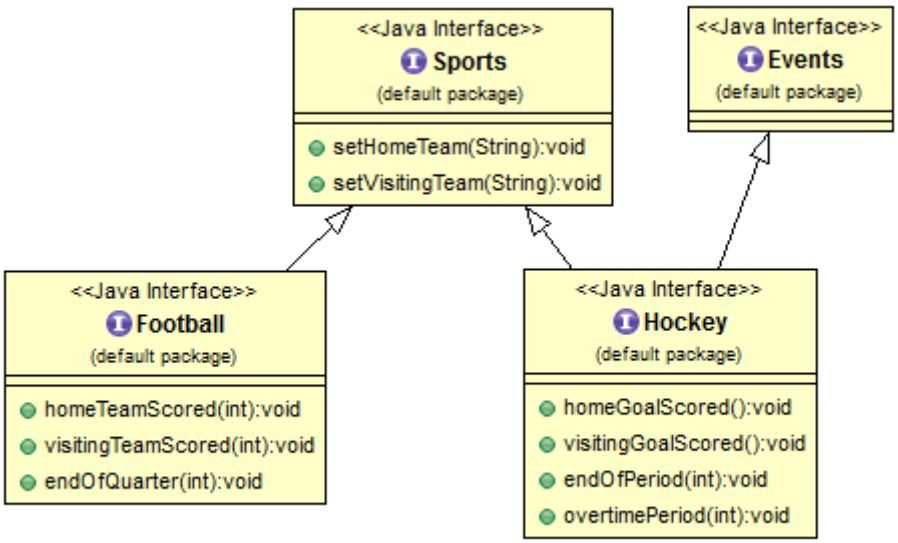
20.4 Birden Çok Arayüzün Genişletilmesi

Java'da bir sınıf sadece bir adet sınıftan genişletilebilir. Çoklu mirasa izin verilmez. Arayüzler sınıflar değildir o yüzden birden fazla arayüze genişletilebilir.

extend kelimesi kullanıldıktan sonra genişletilecek ana arayüzler virgüllerle ayrılarak yazılır.

Örneğin Hockey hem Sports hem de Event arayüzlerinden genişletilseydi aşağıdaki gibi tanımlanacaktı.

```
public interface Hockey extends Sports, Event
```

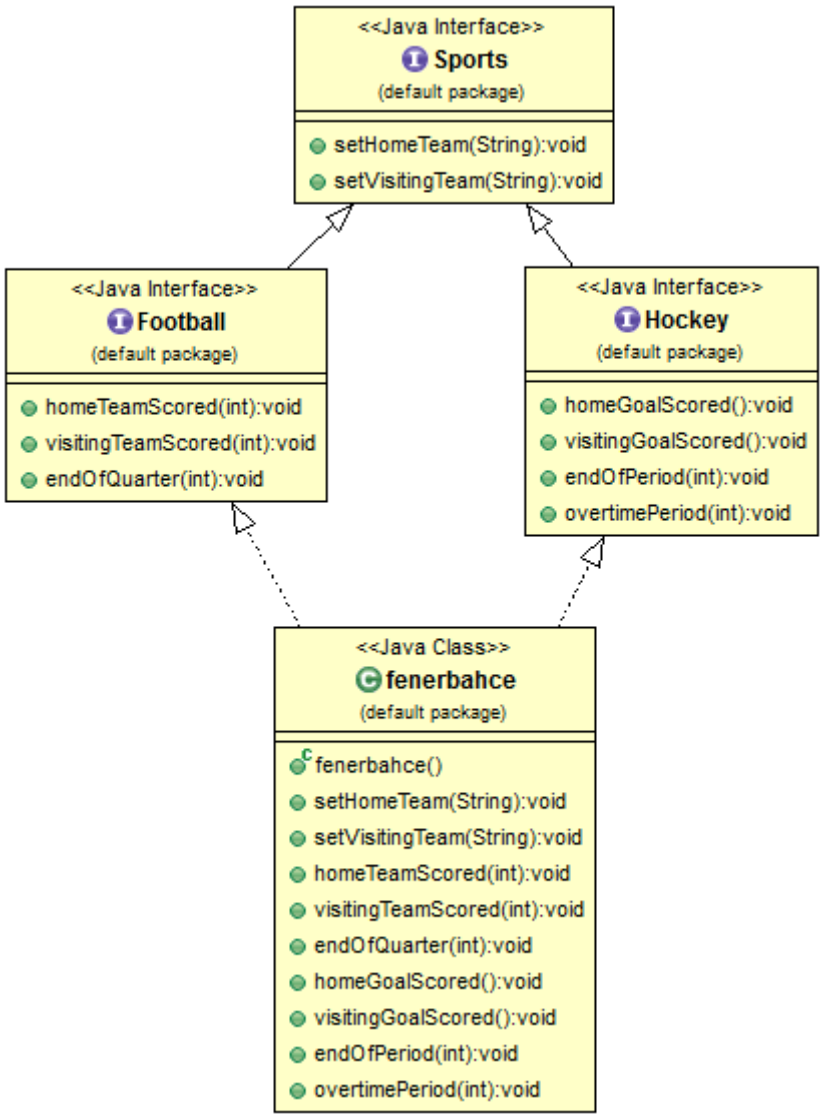


20.5 Birden Çok Arayüzün Uygulanması

Java'da bir sınıf sadece bir adet sınıftan genişletilebilir. Çoklu mirasa izin verilmez. Arayüzler sınıflar değildir o yüzden birden fazla arayüzden genişletilebilir.

implements kelimesi kullanıldıktan sonra, kullanılacak arayüzler virgülle ayrılarak yazılır.

Örnek olarak, eğer fenerbahce sınıfı hem Hockey hem de Football sınıfını uygulasaydı; kod aşağıdaki gibi olmalıydı.




```
public class fenerbahce implements Football, Hockey{

        public void setHomeTeam(String name) {

        }

        public void setVisitingTeam(String name) {

        }

        public void homeTeamScored(int points) {

        }

        public void visitingTeamScored(int points) {

        }

        public void endOfQuarter(int quarter) {

        }

        public void homeGoalScored() {

        }
```

```
        public void visitingGoalScored() {

        }

        public void endOfPeriod(int period) {

        }

        public void overtimePeriod(int ot) {

        }
}
```

PART 21

21 Üstüne Yazmak (Overriding)

Bir önceki bölümde süper sınıflar ve alt sınıflardan bahsetmiştik. Eğer bir sınıf bir üst sınıftan bir metodu miras alıyorsa, o metodun üstüne yazmak mümkündür. Ancak metot final olarak tanımlanmamış olmalıdır.

Üstüne yazmanın avantajı şudur. Alt sınıfa özel bir metodun davranışını belirlemek. Yanı alt sınıf; ana sınıftan aldığı bir metodu kendi gereksinimlerine göre değiştirebilir.

Nesne yönelimli programlamanın koşullarında üstüne yazmak; var olan bir metodun işlevselliğini değiştirmektir.

Örnek:

Örneği inceleyelim:

```
class Animal{

   public void move(){
      System.out.println("Animals can move");
   }
}

class Dog extends Animal{

   public void move(){
      System.out.println("Dogs can walk and run");
```

```
   }
}

public class TestDog{
   public static void main(String args[]){
      Animal a = new Animal();
      Dog    b = new Dog();

      a.move();// runs the method in Animal class

      b.move();//Runs the method in Dog class
   }
}
```

Bu aşağıdaki sonucu verecektir.

```
Animals can move
Dogs can walk and run
```

Aşağıdaki örneği göz önüne alalım

```
class Animal{

   public void move(){
      System.out.println("Animals can move");
   }
}

class Dog extends Animal{

   public void move(){
      System.out.println("Dogs can walk and run");
   }
   public void bark(){
      System.out.println("Dogs can bark");
   }
}

public class TestDog{

   public static void main(String args[]){
      Animal a = new Animal();
      Dog b = new Dog();

      a.move();// runs the method in Animal class
      b.move();//Runs the method in Dog class
      b.bark();
   }
}
```

Bu aşağıdaki sonucu verecektir.

```
Animals can move
Dogs can walk and run
Dogs can bark
```

21.1 Üstüne Yazma Kuralları (Overriding Rules)

- Argüman listesi değiştirilen metotla tamamen aynı olmalıdır.
- Geri dönüş tipi aynı olmalıdır.
- Erişim düzeyi değiştirilen metottan daha fazla olamaz. Örneğin süper sınıfın metodu public tanımlandıysa; alt sınıfta değiştirilecek metot public veya private olamaz.
- **final olarak tanımlanan bir metot değiştirilemez.**
- **static olarak tanımlanan bir metot değiştirilemez**
- **Bir metot miras alınamıyorsa değiştirilemez**
- **Constructor metotlar değiştirilemez.**

21.2 Super Kelimesinin Kullanılması

Değiştirilen bir metodun süper sınıf versiyonu çağırılıyorsa super kelimesi kullanılmalıdır.

```
class Animal{

   public void move(){
      System.out.println("Animals can move");
   }
}

class Dog extends Animal{

   public void move(){

      super.move(); // invokes the super class method

      System.out.println("Dogs can walk and run");
   }
}

public class TestDog{

   public static void main(String args[]){

      Dog b = new Dog();
      b.move(); //Runs the method in Dog class

   }
}
```

Bu aşağıdaki sonucu verecektir:

```
Animals can move
Dogs can walk and run
```