Arayüz Sınıflar ve Soyut Sınıflar

Computer Engineering Department

Java Course

Prof. Dr. Ahmet Sayar

Kocaeli University - Fall 2021

Arayüz (Interface) Kavramı

- Java'da **arayüz (interface), bir sınıfta olması gereken metod ve özellikleri tanımlayan yapıdır.**
- Kendisi normal bir sınıf değildir, sadece neyin yapılacağını göstermekte, ancak nasıl yapılacağını göstermemektedir.
- Java'da çoklu kalıtım dahili sınıflar ve arayüzlerle yapılmaktadır.
- Arayüzler değişkenler ve gövdesiz (soyut) metodlardan oluşur.
- Tüm methodlar gövdesiz olmak zorunda.
- { } arası boş bırakılarak da yazılamaz. Gövde boş da olsa bu şekilde olmaz.

Arayüz Oluşturma

- Arayüz tanımlanırken **Class** ifadesi yerine **Interface** ifadesi kullanılır.
- İçerdiği metodlar gövdesizdir.

```
public interface Matter{
        public double getDensity()
        public double getVolume();
        public double getMass();
}
```

- Bu ifade, “bir maddenin yoğunluğu, hacmi ve kütlesi olur” demenin Java’daki yoludur.
- Maddenin hacmi nasıl hesaplanır, kütlesi nasıl verilir, hiçbir şekilde belirtmemektedir.
- Sadece ne olması gerektiğini söylemektedir.

Arayüz Uygulaması

- Bir sınıfın arayüzdeki bütün metodları içerdiğini, gerçekleştirdiğini belirtmesine uygulama (implementation) denir ve '**implements‘** anahtar kelimesiyle ifade edilir.

```
public class CubeMatter implements
Matter{
        public double density=1.0;
        public double edge=1.0;
        public double getDensity(){
        return density;
        }
        public double getVolume(){
        return edge*edge*edge;
        }
        public double getMass(){
        return density*edge*edge*edge;
        }
}
```

```
public class SphereMatter implements Matter{
        public double density=1.0;
        public double radius=1.0;
        public double getDensity(){
        return density;
        }
        public double getVolume(){
        return
        (3.14*radius*radius*radius)/3;
        }
        public double getMass(){
        return
        density*(3.14*radius*radius*radius)/3;
        }
}
```

- Burada "Küp/Küre diye bir nesnemiz var ve o bir maddedir, bir maddede olabilecek bütün nitelikler onda da bulunur.“ demiş olduk ve bunların nasıl hesaplandığını gösterdik.

Örnek

```
interface Nakliye {
        public void yukle(double yuk) ;
        public void bosalt(double yuk);
    }
```

```
Class Ucak implements Nakliye{
    double tonaj=80;

    public void yukle(){ // Override
        if(yük<tonaj)
            System.out.println("Uçağa
Yükleniyor...");
    }

     public void bosalt(){ // Override
        if(yük<tonaj)
            System.out.println("Uçaktan
boşaltılıyor...")
    }
}
```

```
Class Gemi implements Nakliye{
    double tonaj=10000;

    public void yukle(){ // Override
        if(yük<tonaj)
            System.out.println("Gemiye
Yükleniyor...");
    }

     public void bosalt(){ // Override
        if(yük<tonaj)
            System.out.println("Gemiden
boşaltılıyor...")
    }
}
```

Örnek – Interface Sınıfı Nasıl Tanımlanabilir

```
public interface DortIslem {
    int k;                                          //???
    private double topla(int t1, int t2);           // ???
    public double cikar(int t1, int t2){};          //???
    protected double carp(int t1, int t2);          //???
    double bol(int t1, int t2);                     //???
    public double bol(int t1, int t2);              //???
}
```

Hangileri yanlistir neden?

Arayüzle Çoklu Kalıtım Sağlamak

- Bir alt sınıfın sadece bir direk üst sınıfı olabilir.
- Ancak aynı anda bir sınıfı extend edip bir veya daha fazla arayüzü implement edebilir.
- "extends" anahtar sözcüğünün aksine, "implements" anahtar sözcüğüyle bir sınıfın aldığı hiç bir şey yoktur. Sadece bazı metodları implement etmeyi taahhüt etmektedir.
- Bir sınıf istediği kadar arayüzü implement edebilir.

Türetilen Arayüzler

- Diğer sınıflar gibi arayüzler de türetilebilir.
- Bu da extends anahtar kelimesiyle aşağıdaki örnekteki gibi yapılır.

```
interface InterfaceName2 extends InterfaceName1
{
    // Body of InterfaceName2
}
```

Arayüzle Çoklu Kalıtım Sağlamak

- Cube sınıfı hem Body sınıfını extend edebilir hem de Matter arayüzünü implement edebilir.

  public class Cube extends Body implements Matter{

  // ...

  }
- Üstsınıf'ın bir arayüzü implement etmesi durumunda altsınıf da etmiş sayılır.

  public class Body implements Matter{

  // ...

  }

  dersek

  public class Cube extends Body{

  // ...

  }

  dememiz yeterlidir. Cube sınıfı Matter‟i implement etmeyi de miras yoluyla almış demektir.

Arayüzlerin Kullanım Özellikleri

- Arayüzler bütün metodları soyut olan bir soyut sınıf gibi düşünülebilirler. Ancak sınıflardan ayrılan başka özellikleri vardır.
- Ayrıca söylensin veya söylenmesin bütün metod ve özellikler public sayılır ve başlarına protected veya private gibi anahtar sözcükler alamazlar.
- Arayüzler de bütün özellikler (değişkenler) **public, final** ve **static**'tir.
- O yüzden arayüzlerde tanımlanan özellikler bir veya daha fazla sınıfta kullanılan sabitler için kullanılır.
- Arayüzlerdeki özelliklere başlangıç değeri verilmelidir.

Arayüzlerin Kullanım Özellikleri

```
public interface MathConstants{
double PI=3.14;
}
public class Circle implements MathConstants{
private double radius=1.0;
public getCircumference(){
return 2*PI*radius;
}
}
```

- Circle sınıfında tanımlanmadığı halde PI değişkeni, sadece Math Constants arayüzünü implement ettiği için erişilebilir durumdadır. Bu anlamda PI bir sabittir.
- PI sadece implement edilen class'lar tarafından görülür
- Java'da **global değişken** ve **sabit** ihtiyacı duyulan yerlerde interface kullanılır.

Soyut Sınıflar

Soyut Sınıf Kavramı

- Programlama sırasında genel davranışları gösteren sınıflar *soyut sınıf* (“abstract class”) olarak kodlanabilir.
  - Soyutlama (“abstraction”)
  - **Soyut sınıflardan nesne oluşturulmaz**.
  - Ancak alt sınıflar yaratılabilir.
  - Soyut sınıflar, ilgili alt-sınıfları tanımlamak ve onlara ilişkin detayları doldurmak amacıyla kullanılırlar (tekrar kullanıma esas olarak).

Soyut Sınıflar

- Bazı metodlarını tanımlamış, bazılarının uygulamasını kendisinden türeyen sınıflara bırakmış olan sınıflara soyut sınıf (**abstract class)** denir.
- Tüm alt sınıfları tarafından paylaşılacak metodların tanımlarını içeren sınıflardır.
- Bunun yanında içeriği belirlenmemiş sadece etiketi belli **bir yada daha fazla soyut metod içerirler**.

Soyut Sınıflar

- Soyut sınıftan nesne oluşturulamaz.
- Mutlaka “extends” anahtar sözcüğü ile yeni sınıflara miras bırakarak kullanılmalıdır.
- Soyut yapılandırıcı ve soyut statik metod tanımlanamaz.
- Içinde hiç soyut method olmayan soyut sınıf olur mu? OLUR
- Ama içinde bir tane bile olsa soyut method varsa sınıf soyut olarak tanımlanmalı.

Soyut Sınıf Oluşturma

- Sınıfın ve metodun soyut olduğunu belirtmek için “abstract” anahtar sözcüğü kullanılır.
- Soyut metodlar şu örneklerde belirtildiği gibi gövdeleri olmayacak şekilde tanımlanır:

  public abstract void voids1();

  public abstract void ints2(int a);
- Eğer bir sınıf enaz bir soyut metod içeriyorsa soyut sınıf olarak tanımlanması gereklidir.

```
abstract class N{
        public abstract void voids1();
}
```

Soyut Sınıf Oluşturma

- Eğer bir alt sınıf soyut bir üst sınıftan türetilmiş ve üst sınıfın tüm soyut metodlarını kendi bünyesinde tanımlamamışsa o da bir soyut sınıf olmak zoundadır.

```
abstract class N{
        public abstract void voids1();
}
class N1 extends N{
        public void voids1(){ System.out.println(“s1”); }
}
abstract class N2 extends N{
        public void voids2(){ System.out.println(“s2”); }
}
```

Soyut Sınıflar

- Soyut sınıfların temel amacı büyük yazılım projelerinin organize bir şekilde geliştirilmelerini sağlamaktır.
  - İlk önce bir soyut sınıf oluşturulur ve yazması planlanan metodlar bu sınıfa soyut şekilde konur.
  - Sonra soyut sınıfların alt sınıfları tanımlanır.
  - Bütün soyut metodların alt sınıflardaki ifadeleri tamamlandığında yazılım tamamlanmış olur.

Örnek

- Bir küp için de, bir küre için de kütle:
  - Hacim ve yoğunluğun çarpımına eşittir.

- Ancak önceki örnekte yaptığımız iki nesnede de bu ortak özelliği kullanamıyor ve kütle hesaplamasını kendisi yapıyordu.

- Matter ara yüzünü implement eden ne kadar sınıf varsa bu işlem o kadar tekrarlanacak demektir. Bu 'ortak‘ işlemi bir kere yapıp, hep onun kullanılmasını sağlamak mümkündür.

Örnek-devam

```
abstract public class Body{
        public double density=1.0;
        public Body(double d){
        density=d;
        }
        public double getDensity(){
        return density;
        }
        public double getMass(){
        return density*getVolume();
        }
        abstract public double getVolume();
}
```

- Soyut Body sınıfı, hem küpün hem de kürenin ortak özelliklerini içermektedir.
- Yoğunluk özelliği ikisinde de ortaktır. Bu şekilde kalıtım yoluyla küp ve küreye geçebilir.

getVolume() metodu hacim hesaplamak için her nesne farklı bir method kullandığı için soyut bırakılmıştır.

getMass() hesaplaması sınıftan sınıfa değişmediği için tanımlanmıştır

Burada getMass() metodu henüz yazılmamış getVolume() metodunu kullanarak bir işlem yapabilmektedir.

Bu sadece NYP teknikleriyle mümkün olan bir programlama işlemidir.

Örnek-devam

```
public class CubeBody extends Body{
public double edge=1.0;
public CubeBody(double d,double e){
super(d);
edge=e;
}
public double getVolume(){
return edge*edge*edge;
}
}
```

```
public class SphereBody extends Body{
public double radius=1.0;
public SphereBody(double d,double r){
super(d);
radius=r;
}
public double getVolume(){
return (3.14 * radius * radius * radius )/3;
}
}
```

- İki sınıf da getMass() diye bir metod yazmak zorunda kalmadan , hiç bir ek kodlama yapmadan kütle hesabı yapabilmektedir.

Soyut Sınıf ve Arayüzler Özet

- Soyut sınıflar ve arayüzler, kendilerinden türetilen alt sınıflara ortak alanları ve üyeleri sunarlar.
- Tanımlanış amaçları her nekadar benzer olsa da yapıları ve kullanım şekilleri açısından farklılıkları vardır.
- Arayüzler sadece özelliklerin ve metodların tanımlamalarını içerirken, Soyut sınıflarda alanlar ve üyelerin tanımlamaların yanısıra işlevlerine de yer verilebilir.
- Soyut sınıflar arayüzlerin aksine static ve final olmayan alanlar-ozellikler içerebilir.
- Soyut sınıflar arayüzlerin aksine implement edilmiş metodlar içerebilir.
- Soyut sınıflar ortak özellikleri çok olan alt sınıfları yaratmakta kullanılabilir.
- Arayüzler ise genel kurallar ve kosulların kontrolu için kullanılan sınıf yapılarıdır.
- Yanlızca 1 abstract class dan extend edilebilir. Ancak 1 den fazla class implements edilebilir.

Soyut Sınıf -Örnek

```
package not04c;

public abstract class VergiKaynagi {
    private double oran;

    public VergiKaynagi(double oran) {
        this.oran = oran;
    }
    public double getOran() {
        return oran;
    }
    public abstract double getAylikVergi();
    public abstract double getYillikVergi();
    public abstract double getSabitVergi();

}
```

Soyut Sınıflar -Örnek

```
package not04c;
public class KiraKontrati extends VergiKaynagi {
    private double aylikKira;

    public KiraKontrati(double oran, double aylikKira) {
        super(oran);
        this.aylikKira = aylikKira;
    }
    public double getAylikVergi() {
        return aylikKira * getOran();
    }
    public double getYillikVergi() {
        return getAylikVergi() * 12;
    }
    public double getSabitVergi() {
        return 0;
    }
}
```

Abstract olan tum siniflari implement etmen gerekir, aksi taktirde extend eden sinif da abstract olmak zorundadir.