Nesne Tabanlı Programlama II

Öğr. Gör. Dr. Aysun ALTIKARDEŞ

AMAÇ

- Interface ve Abstract Class'lar

üzerinde önemle durularak, temel programlama bilgilerinin kazandırılması amaçlanmıştır.

INTERFACE KAVRAMI

Java'da interface, bir class'ta olması gereken method ve property'leri tanımlayan yapıdır.

Kendisi normal bir class değildir, sadece neyin yapılacağını göstermekte, ancak nasıl yapılacağını göstermemektedir.

INTERFACE KAVRAMI

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

Bu ifade, **'bir maddenin yoğunluğu, hacmi ve kütlesi olur'** demenin **Java**'daki yoludur.

INTERFACE IMPLEMENTASYONU

Bir class'ın interface'deki bütün **method**'ları içerdiğini, gerçekleştirdiğini belirtmesine **implementation** denir ve '**implements**' keyword'üyle kullanılır.

INTERFACE IMPLEMENTASYONU

```
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;
}
}
```

INTERFACE IMPLEMENTASYONU

Bu örnekte "Küp diye bir nesnemiz var ve o bir maddedir, yani bir maddede olabilecek bütün nitelikler onda da bulunur." demiş olduk ve bunların nasıl hesaplandığını gösterdik.

INTERFACE IMPLEMENTASYONU

```
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;
}
}
```

ABSRACT SINIFLAR

Bazı methodlarını implement etmiş, bazılarının imlementation'unun kendisini **extend** eden class'a bırakmış olan class'a **abstract class** denir.
Bu tip class'lar en çok, iki class'ın ortak methodlarından bazılarının implementation'u da **aynı olması** durumunda kullanılır.

ABSRACT SINIFLAR

```
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();
 }
```

ABSRACT SINIFLAR

Bir method (**getMass()**) henüz yazılmamış bir methodu (**getVolume()**'u) kullanarak bir işlem yapabilmektedir. Bu şekilde her cisim için ayrı ayrı hesap yapmaktan kurtulmuş olduk.

ABSRACT SINIFLAR

```
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;
  }
}
```

ABSRACT SINIFLAR

```
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;
  }
}
```

INTERFACE‘in KULLANIM ÖZELLİKLERİ

Interface'ler bütün methodları absract olan bir **absract class** gibi düşünülebilirler. Ancak class'lardan ayrılan başka özellikleri vardır.

Interface'lerde bütün propertyler **public**, **final** ve **static**'tir. O yüzden interface'lerde tanımlanan property'ler bir veya daha fazla class'da kullanılan sabitler için kullanılır.

INTERFACE'in KULLANIM ÖZELLİKLERİ

```
public interface MathConstants{
  double PI=3.14;
}

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