NESNEYE YÖNELİK PROGRAMLAMA

Kalıtım

Emir Öztürk

KALITIM

- Üst sınıf
- Ortak özellikler
- super
- extends
- protected

Kalıtım(Inheritance), bir sınıfın başka bir sınıfın tüm özelliklerini içermesi ve kullanabilmesi için kullanılan bir yapıdır.
Ortak olan özellikler bir üst sınıfta toplandıktan sonra kalıtım bu sınıf üzerinden gerçekleştirilir.
Oluşturulan bu üst sınıfa temel (base) ya da super class adı verilir.
Java için bir sınıfın diğerinden miras alması için extends anahtar kelimesi kullanılır.
Ayrıca miras almış bir sınıfın içerisinden üst sınıfın değişkenlerine erişebilmek için üst sınıf değişkenlerinin protected olarak tanımlanması gerekmektedir.

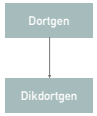
TEMEL SINIF

```
class Dortgen{
    private int birinciKenar;
    private int ikinciKenar;
    private int ucuncuKenar;
    private int dorduncuKenar;
    public Dortgen(int BirinciKenar,int IkinciKenar,int UcuncuKenar,int DorduncuKenar){
        birinciKenar = BirinciKenar;
        ikinciKenar = IkinciKenar;
        ucuncuKenar = UcuncuKenar;
        dorduncuKenar = DorduncuKenar;
    }
    public Dortgen(){
        birinciKenar=-1;
        ikinciKenar=-1;
        ucuncuKenar=-1;
        dorduncuKenar=-1;
    }
    public int CevreHesapla(){
        return birinciKenar+ikinciKenar+ucuncuKenar+dorduncuKenar;
    }
}
```

Örneklerde temel sınıf olarak kullanılacak Dortgen sınıfı

TEKİL KALITIM

- Bir üst sınıf
- Bir alt sınıf



Tekil kalıtım bir sınıfın sadece bir üst sınıftan miras almasıdır. Örneğin bir önceki slayttaki Dortgen sınıfı bir üst sınıf olarak kabul edilip bu sınıfı miras alan Dikdortgen sınıfı tanımlanabilir.

PROTECTED

```
class Dikdortgen extends Dortgen{
    public int CevreHesapla(){
        return birinciKenar+ikinciKenar+ucuncuKenar+dorduncuKenar;
    }
}
```

```
class Dortgen{
    private int birinciKenar;
    private int ikinciKenar;
    private int ucuncuKenar;
    private int dorduncuKenar;
```

→

```
class Dortgen{
    protected int birinciKenar;
    protected int ikinciKenar;
    protected int ucuncuKenar;
    protected int dorduncuKenar;
```

```
class Dikdortgen extends Dortgen{
    public int CevreHesapla(){
        return birinciKenar+ikinciKenar+ucuncuKenar+dorduncuKenar;
    }
}
```

Dortgen sınıfından miras alabilmek için sınıf adının yanına extends kelimesi ile üst sınıfın ismi yazılır.

ÜST SINIF METOTLARI

```
class Dikdortgen extends Dortgen{
    public Dikdortgen(int KisaKenar,int UzunKenar){
        birinciKenar = KisaKenar;
        ikinciKenar = KisaKenar;
        ucuncuKenar = UzunKenar;
        dorduncuKenar = UzunKenar;
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Dikdortgen d = new Dikdortgen( KisaKenar: 5, UzunKenar: 10);
        System.out.println(d.CevreHesapla());
    }
}
```

Ekran çıktısı: 30

Dikdortgen sınıfından bir nesne oluşturulduğında CevreHesapla() metodu olmadığı halde miras aldığı Dortgen sınıfında bu metodun olması sayesinde hatasız bir şekilde ekran çıktısı alınabilmektedir.

ÇOK SEVİYELİ (MULTI LEVEL) KALITIM

- Bir üst sınıf
- Hem üst hem alt sınıf

Çok seviyeli kalıtımda bir sınıf diğer bir sınıftan miras alırken, miras alan alt sınıf başka bir sınıf için miras alınacak bir üst sınıf olabilir. Örneğin bir B sınıfı bir A sınıfından miras alırken A sınıfının alt sınıfı olarak kabul edilirken, bir C sınıfı da B sınıfından miras alabilir ve B sınıfı C sınıfı için üst sınıf olur.

ÇOK SEVİYELİ KALITIM

```
class Dortgen{
    protected int birinciKenar;
    protected int ikinciKenar;
    protected int ucuncuKenar;
    protected int dorduncuKenar;
    public Dortgen(int BirinciKenar,int IkinciKenar,int UcuncuKenar,int DorduncuKenar){
        birinciKenar = BirinciKenar;
        ikinciKenar = IkinciKenar;
        ucuncuKenar = UcuncuKenar;
        dorduncuKenar = DorduncuKenar;
    }
    public Dortgen(){
        birinciKenar=-1;
        ikinciKenar=-1;
        ucuncuKenarm=-1;
        dorduncuKenar=-1;
    }
    public int CevreHesapla(){
        return birinciKenar+ikinciKenar+ucuncuKenar+dorduncuKenar;
    }
}
```

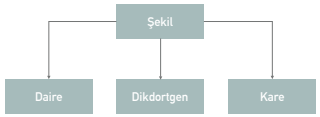
```
class Dikdortgen extends Dortgen{
    public Dikdortgen(int KisaKenar,int UzunKenar){
        birinciKenar = KisaKenar;
        ikinciKenar = KisaKenar;
        ucuncuKenar = UzunKenar;
        dorduncuKenar = UzunKenar;
    }
    public Dikdortgen(){
        birinciKenar = -1;
        ikinciKenar = -1;
        ucuncuKenar = -1;
        dorduncuKenar = -1;
    }
}
```

```
class Kare extends Dikdortgen{
    public Kare(int kenar){
        birinciKenar = kenar;
        ikinciKenar = kenar;
        ucuncuKenar = kenar;
        dorduncuKenar = kenar;
    }
}
```

Verilen örnekte Dikdortgen sınıfı Dortgen sınıfından miras alırken, Kare sınıfı ise Dikdortgen sınıfından miras almaktadır.

HİYERARŞİK KALITIM

- Bir üst sınıf
- Birden fazla alt sınıf



Hiyerarşik kalıtımda bir sınıf birden fazla sınıfın üst sınıfı olabilmektedir. Hiyerarşik olarak kullanılan sınıflar da yeniden miras alma için kullanılabilir. Örneğin B sınıfı farklı bir sınıf için üst sınıf görevi görebilmektedir.

HİYERARŞİK KALITIM

```
class Sekil{
    protected Color renk;
    public Sekil(){ }
}
```

```
class Kare extends Sekil{
    private int kenar;
    public Kare(int Kenar,Color Renk){
        kenar = Kenar;
        renk = Renk;
    }
}
```

```
class Dikdortgen extends Sekil{
    private int KisaKenar;
    private int uzunKenar;
    public Dikdortgen(int KisaKenar,int UzunKenar,Color Renk){
        kisaKenar = KisaKenar;
        uzunKenar = UzunKenar;
        renk = Renk;
    }
}
```

```
class Daire extends Sekil{
    private int r;
    public Daire(int R,Color Renk){
        r = R;
        renk = Renk;
    }
}
```

Örnek olarak bir Sekil sınıfından miras alarak Kare, Dikdortgen ve Daire sınıfları tanımlanabilmektedir. Bu üç sınıf Sekil sınıfının ortak renk alanına sahip olmakla beraber, her sınıf kendisi için ihtiyaç duyduğu değişkenleri ayrıca içermektedir. Yapıcılar ise bu sınıfların miras aldığı sınıftaki renk değerine atama yapabilmektedir.

ÜST SINIFIN KULLANILMASI / YAPICI SIRASI

```
class Sekil{
    public Sekil(){ System.out.println("Sekil Yapicisi"); }
}
class Dikdortgen extends Sekil{
    public Dikdortgen(){ System.out.println("Dikdortgen Yapicisi"); }
}
public class Main {
    public static void main(String[] args) {
        Sekil s = new Sekil();
        var vs = new Sekil();
        Dikdortgen d = new Dikdortgen();
        var vd = new Dikdortgen();
        Sekil sd = new Dikdortgen();
    }
}
```

Sekil Yapicisi
Sekil Yapicisi
Sekil Yapicisi
Dikdortgen Yapicisi
Sekil Yapicisi
Dikdortgen Yapicisi
Sekil Yapicisi
Dikdortgen Yapicisi

Bir temel sınıfın varsayılan yapıcısı varsa, miras almış başka bir sınıf oluşturulduğunda da önce bu temel sınıfın yapıcısı çalıştırılır.
Alt sınıflar temel sınıfın referansları ile oluşturulabilirler. Örneğin son satırda Sekil sınıfından bir referans Dikdortgen sınıfından bir nesneyi işaret edebilir.
var kelimesi ile oluşturulan değişkenler tür çözümleme işlemini new kelimesinden sonra kullanılan sınıf ismi ile yaptıkları için son satırdaki gibi temel sınıf referansı durumu söz konusu değildir.

ÜST SINIFIN KULLANILMASI / PARAMETRE AKTARIMI

```
class Sekil{ }
class Dikdortgen extends Sekil{ }
class Kare extends Sekil{ }
class Daire extends Sekil { }

public class Main {
    static void sekilAl(Sekil s){}

    public static void main(String[] args) {
        var di = new Dikdortgen();
        var ka = new Kare();
        var da = new Daire();
        sekilAl(di);
        sekilAl(ka);
        sekilAl(da);
    }
}
```

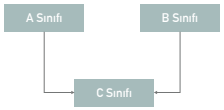
Üst sınıftan bir nesne parametre alan fonksiyonlara bu sınıftan türetilen sınıfların da verilebilmesi mümkündür. Şekil sınıfından bir nesne alan sekilAl metoduna Sekil sınıfından miras almış Dikdortgen, Kare ve Daire sınıfları da parametre olarak verilebilmektedir.

ÜST SINIFIN YAPICISININ ÇAĞIRILMASI

Bir alt sınıfta üst sınıfının parametreli yapıcısının çağırılması istendiği takdirde super kullanılmalıdır.
super kelimesi kullanılmadığında sınıf oluşturulduğunda öncelikle bir Kutu sınıfı yapıcısı çağırılmak istenecektir (varsayılan boş yapıcı). Böyle bir yapıcının olmaması halinde (soldaki durum) hata oluşmaktadır. Yapıcı içerisinde önce üst sınıfın parametreli yapıcısının çağırılması için super anahtar kelimesi kullanılmaktadır.

ÇOKLU KALITIM

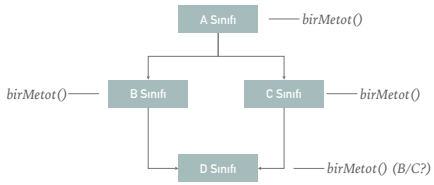
- Birden fazla üst sınıf
- Bir alt sınıf
- ~~Java~~



Bir sınıfın birden fazla üst sınıftan miras almasına çoklu kalıtım adı verilir.
Bir C sınıfı hem A hem de B sınıfının özelliklerini içerebilmektedir.
Java çoklu kalıtımı desteklememektedir.

DIAMOND PROBLEM

- Bir sınıftan türetilen 2 sınıf
- 2 sınıfın aynı metodu aşırı yüklemesi
- D sınıfının metodu içermemesi



Çoklu kalıtımın desteklenmesi durumunda bir D sınıfı B ve C sınıflarından miras aldığı takdirde eğer bu B ve C sınıfları ortak bir sınıftan miras alıyorsa diamond problem oluşur.
A sınıfının içerisinde bir metot olduğunda B ve C sınıflarının bu metodu kendilerine göre aşırı yüklediklerini kabul edelim. Bu durumda D sınıfı bu metodu kullanmaya kalktığında hangi metodun çağırılacağı belirsiz olmaktadır. Bu sorunu çözmek adına ve çoklu kalıtıma ihtiyaç duyulması durumunda arayüzler kullanılabilmektedir.