

Nesne Yönelimli Programlama

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Not: Bu dersin sunumları, "Java Programlama Dili ve Yazılım Tasarımı, Altuğ B. Altıntaş, Papatya Yayıncılık, 2016" kitabı kullanılarak hazırlanmıştır.

Konular

- Polimorfizm
- Geç Bağlama
- Final ve Geç Bağlama
- Polimorfizm ve Tekrar Kullanılabilirlik
- Genişletilebilirlik
- Soyut Sınıflar ve Yordamlar
- Polimorfizm ve Yapılandırıcılar
- Kalıtım ve Yukarıya Çevirim (Upcasting)
- Aşağıya Çevirim (Downcasting)

Polimorfizm

- **Polimorfizm** (çok biçimlilik), **nesneye yönelik programlamanın önemli kavramlarından birisidir.**
- **Polimorfizm ile kalıtım yakından ilişkilidir.**
- **Kalıtım** konusunda iki taraf bulunmaktadır, **ana sınıf** ve bu sınıftan türeyen **alt sınıf/sınıflar.**
- **Alt sınıf**, türetildiği **ana sınıfa ait tüm özelliklere sahip olur.**
- Ana sınıf ne yapıyorsa türetilen alt sınıfta bu işlemlerin aynısını yapabilir.
- **Türetilen alt sınıfların** kendilerine ait bir çok **yeni özelliği de olabilir.**
- **Türetilen alt sınıfa ait nesneyi, ana sınıf tipindeki referansa bağlamak yukarıya çevirim (upcasting) işlemidir.**

3

Polimorfizm

- Örnekte, **upcasting, polimorfizm** ve geç bağlama (**late binding**) yapılmıştır.
- Ana sınıf **Asker**, türeyen sınıflar **Er** ve **Yuzbasi** sınıflarıdır.
- **Er bir Askerdir, Yuzbasi bir Askerdir.**

```
class Asker {
    public void selamVer() {
        System.out.println("Asker Selam verdi");
    }
}

class Er extends Asker {
    public void selamVer() {
        System.out.println("Er Selam verdi");
    }
}

class Yuzbasi extends Asker {
    public void selamVer() {
        System.out.println("Yuzbasi Selam verdi");
    }
}

class Yuzbasi extends Er {
    public void selamVer() {
        System.out.println("Yuzbasi Selam verdi");
    }
}

public class Polimorfizm {

    public static void hazirOl(Asker a) {
        a.selamVer(); // ! Dikkat !
    }

    public static void main(String args[]) {
        Asker a = new Asker();
        Er e = new Er();
        Yuzbasi y = new Yuzbasi();
        hazirOl(a); // yukarı çevirim ! yok !
        hazirOl(e); // yukarı çevirim (upcasting)
        hazirOl(y); // yukarı çevirim (upcasting)
    }
}
```

Asker Selam verdi
Er Selam verdi
Yuzbasi Selam verdi

4

Polimorfizm

- **Asker** sınıfının yaptığı her işi **Er** ve **Yuzbasi** da yapabilir.
- Bu iki sınıf kendisine özgü özelliklere de sahip olabilir.
- **Asker** sınıfı ile **Er** ve **Yuzbasi** sınıflarının arasında kalıtımsal bir ilişki vardır.
- **Asker** türünde nesne alan `hazirOl()` yordamına **Er** ve **Yuzbasi** tipindeki referanslar gönderilebilir (upcasting).
- Polimorfizm `hazirOl()` yordamının içerisinde yapılmaktadır.
- `hazirOl()` içerisinde **Asker** tipinde olan **a** **referansı kendisine gelen 2 farklı nesneye (Er ve Yuzbasi) bağlanmıştır.**
- Aşağıdaki ifadelerin tümü doğrudur:
`Asker a = new Asker();`
`Asker a = new Er();`
`Asker a = new Yuzbasi();`

5

Konular

- Polimorfizm
- **Geç Bağlama**
- Final ve Geç Bağlama
- Polimorfizm ve Tekrar Kullanılabilirlik
- Genişletilebilirlik
- Soyut Sınıflar ve Yordamlar
- Polimorfizm ve Yapılandırıcılar
- Kalıtım ve Yukarıya Çevirim (Upcasting)
- Aşağıya Çevirim (Downcasting)

Geç Bağlama

- **Polimorfizm** ve geç bağlama (**late binding**) ilişkilidir.
- **Er** nesnesine bağlı **Er** **tipindeki referans** (**e**) **hazirOl()** **yordamına** **parametre** olarak **gönderilebilir**.
- **Er nesnesine ait** **selamVer()** **yordamı bulunamazsa**, **Asker nesnesine ait** **selamVer()** **yordamı çağrılır**.
- **Er sınıfında**, **Asker ana sınıfına ait** olan **selamVer()** **yordamı** **override** yapıldığından, **Er nesnesinin** **selamVer()** **yordamı çağrılır**.
- **Yuzbasi nesnesine bağlı** **Yuzbasi tipindeki referans** **hazirOl()** **yordamına** **parametre** olarak **gönderiliyor**.
- **Hangi nesnenin** **selamVer()** **yordamının** **çağrılacağına çalışma-anında (run-time)** **karar veriliyor** (**late binding**-geç bağlama).
- Bir yordamın ait olduğu nesne derleme anında belli ise, **erken bağlama (early binding**-erken bağlama) denir.

7

Geç Bağlama

```
public class Polimorfizm {  
  
    public static Hayvan rasgeleSec() {  
        int sec = ((int) (Math.random() * 3));  
        Hayvan h = null;  
        if (sec == 0)  
            h = new Hayvan();  
        else if (sec == 1)  
            h = new Kartal();  
        else if (sec == 2)  
            h = new Timsah();  
        return h;  
    }  
  
    public static void main(String args[]) {  
        Hayvan[] h = new Hayvan[3];  
        // diziye doldur  
        for (int i = 0; i < 3; i++) {  
            h[i] = rasgeleSec(); // upcasting  
        }  
        // dizi elemanlarını ekrana bas  
        for (int j = 0; j < 3; j++) {  
            h[j].avYakala(); // !Dikkat!  
        }  
    }  
}  
  
class Hayvan {  
    Hayvan () {  
        System.out.println("Hayvan yapılandırıcı");  
    }  
    public void avYakala() {  
        System.out.println("Hayvan avYakala");  
    }  
}  
  
class Kartal extends Hayvan {  
    Kartal () {  
        System.out.println("Kartal yapılandırıcı");  
    }  
    public void avYakala() {  
        System.out.println("Kartal avYakala");  
    }  
}  
  
class Timsah extends Hayvan {  
    Timsah () {  
        System.out.println("Timsah yapılandırıcı");  
    }  
    public void avYakala() {  
        System.out.println("Timsah avYakala");  
    }  
}
```

Hayvan yapılandırıcı
Timsah yapılandırıcı
Hayvan yapılandırıcı
Kartal yapılandırıcı
Hayvan yapılandırıcı
Timsah avYakala
Kartal avYakala
Hayvan avYakala

Hayvan yapılandırıcı
Hayvan yapılandırıcı
Timsah yapılandırıcı
Hayvan yapılandırıcı
Hayvan avYakala
Timsah avYakala
Hayvan avYakala

8

Konular

- Polimorfizm
- Geç Bağlama
- **Final ve Geç Bağlama**
- Polimorfizm ve Tekrar Kullanılabilirlik
- Genişletilebilirlik
- Soyut Sınıflar ve Yordamlar
- Polimorfizm ve Yapılandırıcılar
- Kalıtım ve Yukarıya Çevirim (Upcasting)
- Aşağıya Çevirim (Downcasting)

Final ve Geç Bağlama

- **Bir sınıf final yapılırsa, bu sınıfa ait tüm yordamlar final yapılmış olur.**
- Tek başına **bir yordam final yapıldığında**, türetilmiş alt sınıflar tarafından **override yapılamaz.**
- Eğer **bir yordam override yapılamazsa**, o zaman **geç bağlama (late binding) özelliği ortadan kalkar.**
- Bir nesneye ait **final olmayan bir yordam çağrıldığında**, Java **geç bağlama (late binding) olup olmadığını kontrol eder.**

Final ve Geç Bağlama

- Kaplan sınıfına ait `goster()` yordamında `kp` nesnesi için geç bağlama yapılmıştır.

```
class Kedi {
    public void yakalaAv() {
        System.out.println("Kedi sinifi Av yakaladi");
    }
}

class Kaplan extends Kedi {
    public static void goster(Kedi k) {
        k.yakalaAv();
    }
    public void yakalaAv() {
        System.out.println("Kaplan sinifi Av yakaladi");
    }
    public static void main(String args[]) {
        Kedi k = new Kedi();
        Kaplan kp = new Kaplan();
        goster(k);
        goster(kp); // yukari dogru cevirim (upcasting)
    }
}
```

Kedi sinifi Av yakaladi
Kaplan sinifi Av yakaladi

11

Final ve Geç Bağlama

- Kaplan sınıfına ait `yakalaAv()` yordamı **final** yapıldığı için **override edilemez ve late binding yapılamaz.**

```
class Kedi2 {
    public final void yakalaAv() {
        System.out.println("Kedi sinifi Av yakaladi");
    }
}

class Kaplan2 extends Kedi2 {
    public static void goster(Kedi2 k) {
        // k.yakalaAv(); // ! dikkat !
    }
    /*
    * final edemez public void yakalaAv() {
    * System.out.println("Kaplan sinifi Av yakaladi"); }
    */
    public static void main(String args[]) {
        Kedi2 k = new Kedi2();
        Kaplan2 kp = new Kaplan2();
        goster(k);
        goster(kp);
    }
}
```

12

Konular

- Polimorfizm
- Geç Bağlama
- Final ve Geç Bağlama
- Polimorfizm ve Tekrar Kullanılabilirlik
- Genişletilebilirlik
- Soyut Sınıflar ve Yordamlar
- Polimorfizm ve Yapılandırıcılar
- Kalıtım ve Yukarıya Çevirim (Upcasting)
- Aşağıya Çevirim (Downcasting)

Polimorfizm ve Tekrar Kullanılabilirlik

- Kalıtım ve polimorfizm olmazsa, **upcasting** ve **downcasting** yapılamaz.

```
public class IsYeriNon {
    public static void mesaiBasla(Object[] o) {
        for (int i = 0; i < o.length; i++){
            if (o[i] instanceof Calisan) {
                Calisan c = (Calisan) o[i]; // aşağıya çevirim
                c.calis();
            } else if (o[i] instanceof Mudur) {
                Mudur m = (Mudur) o[i]; // aşağıya çevirim
                m.calis(); // ((Mudur) o[i]).calis();
            } else if (o[i] instanceof Programci) {
                Programci p = (Programci) o[i]; // aşağıya çevirim
                p.calis();
            } else if (o[i] instanceof Pazarlamaci) {
                Pazarlamaci paz = (Pazarlamaci) o[i]; // aşağıya çevirim
                paz.calis();
            }
            // ...
        }
    }

    public static void main(String[] args) {
        Object[] o = new Object[4];
        o[0] = new Calisan(); // yukarı çevirim (upcasting)
        o[1] = new Programci(); // yukarı çevirim (upcasting)
        o[2] = new Pazarlamaci(); // yukarı çevirim (upcasting)
        o[3] = new Mudur(); // yukarı çevirim (upcasting)
        mesaiBasla(o);
    }
}
```

Programci Calisiyor
Pazarlamaci Calisiyor
Mudur Calisiyor

```
class Calisan {
    public String pozisyon = "Calisan";
    public void calis() {
    }
}

class Mudur {
    public String pozisyon = "Mudur";
    public Mudur() { // yapılandırıcı
        pozisyon = "Mudur";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Mudur Calisiyor");
    }
}

class Programci {
    public String pozisyon = "Programci";
    public Programci() { // yapılandırıcı
        pozisyon = "Programci";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Programci Calisiyor");
    }
}

class Pazarlamaci {
    public String pozisyon = "Pazarlamaci";
    public Pazarlamaci() { // yapılandırıcı
        pozisyon = "Pazarlamaci";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Pazarlamaci Calisiyor");
    }
}
```

Kalıtım yok!!!

Polimorfizm ve Tekrar Kullanılabilirlik

- **Kalıtım ve polimorfizm**, upcasting ve downcasting ile **tekrar kullanılabilirliği artırır**.
- **mesaiBasla()** yordamı polimorfizm ve geç bağlama olduğundan tek satırla yazılabilir.

```
public class IsYeri {  
    public static void mesaiBasla(Calisan[] c) {  
        for (int i = 0; i < c.length; i++) {  
            c[i].calis(); // !Dikkat!  
        }  
    }  
  
    public static void main(String args[]) {  
        Calisan[] c = new Calisan[4];  
        c[0] = new Calisan(); // yukarı çevirim gerekmiyor  
        c[1] = new Programci(); // yukarı çevirim (upcasting)  
        c[2] = new Pazarlamaci(); // yukarı çevirim (upcasting)  
        c[3] = new Mudur(); // yukarı çevirim (upcasting)  
        mesaiBasla(c);  
    }  
}
```

```
Programci Calisiyor  
Pazarlamaci Calisiyor  
Mudur Calisiyor
```

```
class Calisan {  
    public String pozisyon = "Calisan";  
    public void calis() {  
    }  
}  
  
class Mudur extends Calisan {  
    public Mudur() { // yapılandırıcı  
        pozisyon = "Mudur";  
    }  
    public void calis() { // iptal etme (override)  
        System.out.println("Mudur Calisiyor");  
    }  
}  
  
class Programci extends Calisan {  
    public Programci() { // yapılandırıcı  
        pozisyon = "Programci";  
    }  
    public void calis() { // iptal etme (override)  
        System.out.println("Programci Calisiyor");  
    }  
}  
  
class Pazarlamaci extends Calisan {  
    public Pazarlamaci() { // yapılandırıcı  
        pozisyon = "Pazarlamaci";  
    }  
    public void calis() { // iptal etme (override)  
        System.out.println("Pazarlamaci Calisiyor");  
    }  
}
```

Kalıtım var!!!

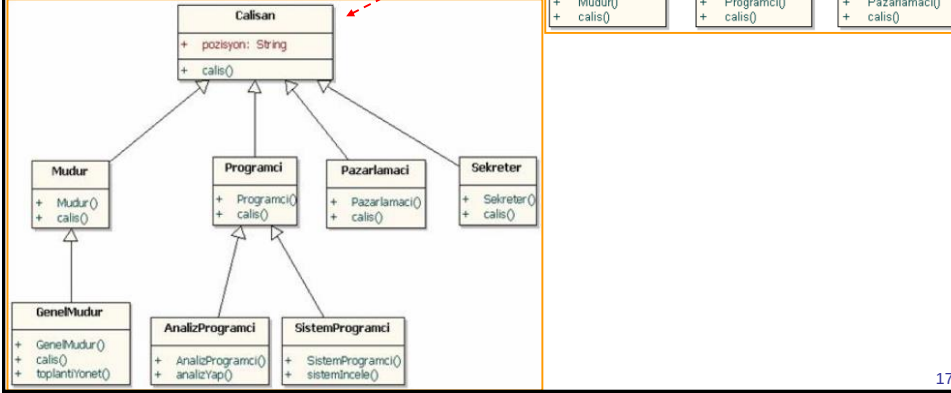
15

Konular

- Polimorfizm
- Geç Bağlama
- Final ve Geç Bağlama
- Polimorfizm ve Tekrar Kullanılabilirlik
- **Genişletilebilirlik**
- Soyut Sınıflar ve Yordamlar
- Polimorfizm ve Yapılandırıcılar
- Kalıtım ve Yukarıya Çevirim (Upcasting)
- Aşağıya Çevirim (Downcasting)

Geniřletilebilirlik

- Geniřletilebilirlik, mevcut hiyerarřiyi kalıtım ile geniřletmez.



17

Geniřletilebilirlik

```
class Calisan {
    public String pozisyon;
    public void calis() {
    }
}

class Mudur extends Calisan {
    public Mudur() { // yapilandirici
        pozisyon = "Mudur";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Mudur Calisiyor");
    }
}

class GenelMudur extends Mudur {
    public GenelMudur() { // yapilandirici
        pozisyon = "GenelMudur";
    }
    public void calis() { // iptal etme (override)
        System.out.println("GenelMudur Calisiyor");
    }
    public void toplantıYonet() {
        System.out.println("GenelMudur toplantı yönetiyor");
    }
}

class Programci extends Calisan {
    public Programci() { // yapilandirici
        pozisyon = "Programci";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Programci Calisiyor");
    }
}
```

```
class AnalizProgramci extends Programci {
    public AnalizProgramci() { // yapilandirici
        pozisyon = "AnalizProgramci";
    }
    public void analizYap() {
        System.out.println("Analiz Yapiliyor");
    }
}

class SistemProgramci extends Programci {
    public SistemProgramci() { // yapilandirici
        pozisyon = "SistemProgramci";
    }
    public void sistemIncele() {
        System.out.println("Sistem Inceleniyor");
    }
}

class Pazarlamaci extends Calisan {
    public Pazarlamaci() { // yapilandirici
        pozisyon = "Pazarlamaci";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Pazarlamaci Calisiyor");
    }
}

class Sekreter extends Calisan {
    public Sekreter() { // yapilandirici
        pozisyon = "Sekreter";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Sekreter Calisiyor");
    }
}
```

18

Geniřletilebilirlik

- Mevcut hiyerarřı geniřletildi ve 4 yeni sınıf eklenmiřtir. (GenelMudur, AnalizProgramci, SistemProgramci, Sekreter)

```
public class BuyukIsYeri {  
    public static void mesaiBasla(Calisan[] c) {  
        for (int i = 0; i < c.length; i++) {  
            c[i].calis(); // ! Dikkat !  
        }  
    }  
  
    public static void main(String args[]) {  
        Calisan[] c = new Calisan[7];  
        c[0] = new Calisan(); // yukarı çevirim gerekiyor  
        c[1] = new Programci(); // yukarı çevirim (upcasting)  
        c[2] = new Pazarlamaci(); // yukarı çevirim (upcasting)  
        c[3] = new Mudur(); // yukarı çevirim (upcasting)  
        c[4] = new GenelMudur(); // yukarı çevirim (upcasting)  
        c[5] = new AnalizProgramci(); // yukarı çevirim (upcasting)  
        c[6] = new SistemProgramci(); // yukarı çevirim (upcasting)  
        mesaiBasla(c);  
    }  
}
```

Programci Calisiyor
Pazarlamaci Calisiyor
Mudur Calisiyor
GenelMudur Calisiyor
Programci Calisiyor
Programci Calisiyor

19

Konular

- Polimorfizm
- Ge Baėlama
- Final ve Ge Baėlama
- Polimorfizm ve Tekrar Kullanılabilirlik
- Geniřletilebilirlik
- Soyut Sınıflar ve Yordamlar
- Polimorfizm ve Yapılandırıcılar
- Kalıtım ve Yukarıya Çevirim (Upcasting)
- Ařaėıya Çevirim (Downcasting)

Soyut Sınıflar ve Yordamlar

- **Aşağıdaki sınıf** hiçbir iş yapmamaktadır, ancak **birleştirici rolüne sahiptir**.

```
class Calisan {  
    public String pozisyon;  
    public void calis() {  
    }  
}
```

- **Soyut sınıflardan** `new()` ile **nesne oluşturulamaz**.
- **Soyut bir sınıftan türetilmiş alt sınıflara ait nesneler**, bu soyut sınıf tipindeki **referanslara bağlanabilirler (upcasting)**.
- Böylece **polimorfizm** ve **geç bağlama kullanılabilir**.
- **Bir sınıfın soyut olması için**, bu sınıfın **içerisinde en az bir adet soyut yordamın bulunması gerekir**.
- **Soyut yordamların gövde kısmı olmaz** (içi boş hiçbir iş yapmayan yordamdır).

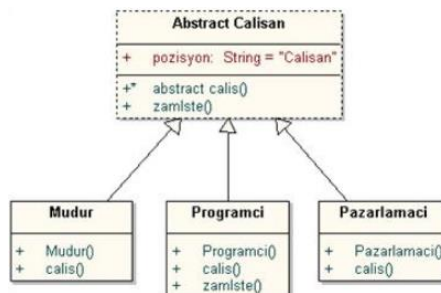
21

Soyut Sınıflar ve Yordamlar

- Soyut bir sınıftan türetilmiş **alt sınıflar**, **soyut sınıfın soyut yordamlarını override yapmak zorundadır**.
- Eğer türetilmiş sınıflar, soyut ana sınıflara ait soyut yordamları override yapmazsa, **derleme hatası oluşur**.

```
abstract void calis() ; // gövdesi olmayan soyut yordam
```

- **Soyut sınıfların içerisinde** soyut yordamların yanı sıra, **gövdeleri olan yani iş yapan yordamlar da bulunabilir**.



22

Soyut Sınıflar ve Yordamlar

- Soyut `Calisan` sınıfında 2 yordam var (`calis()` abstract).
- `zamIste()` soyut değil.
- Türetilen sınıflarda `calis()` **override** yapılmak zorundadır.

```
class Pazarlamaci extends Calisan {
    public Pazarlamaci() { // yapılandırıcı
        pozisyon = "Pazarlamaci";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Pazarlamaci Calisiyor");
    }
}

public class AbiYeri {
    public static void mesaiBasla(Calisan[] c) {
        for (int i = 0; i < c.length; i++) {
            c[i].calis(); // !Dikkat!
        }
    }
    public static void main(String args[]) {
        Calisan[] c = new Calisan[3];
        // c[0] = new Calisan(); // soyut sınıflar new ile direk oluşturulamazlar
        c[0] = new Programci(); // yukarı çevirim (upcasting)
        c[1] = new Pazarlamaci(); // yukarı çevirim (upcasting)
        c[2] = new Mudur(); // yukarı çevirim (upcasting)
        mesaiBasla(c);
    }
}
```

```
abstract class Calisan {
    public String pozisyon = "Calisan";
    public abstract void calis(); // soyut yordam
    public void zamIste() { // soyut olmayan yordam
        System.out.println("Calisan zamIste");
    }
}

class Mudur extends Calisan {
    public Mudur() { // yapılandırıcı
        pozisyon = "Mudur";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Mudur Calisiyor");
    }
}

class Programci extends Calisan {
    public Programci() { // yapılandırıcı
        pozisyon = "Programci";
    }
    public void calis() { // iptal etme (override)
        System.out.println("Programci Calisiyor");
    }
    public void zamIste() { // iptal etme (override)
        System.out.println("Programci Zam Istiyor");
    }
}
```

Programci Calisiyor
Pazarlamaci Calisiyor
Mudur Calisiyor

23

Soyut Sınıflar ve Yordamlar

- `CepTelefonuCizim` ve `MonitorCizim` sınıfları, ana sınıfa ait olan `noktaCiz()` yordamını override yapmıştır.
- Ana sınıftaki `cizgiCiz()` override yapan yordamı kullanır.

```
abstract class Cizim {
    // soyut yordam
    public abstract void noktaCiz(int x, int y);

    // soyut olmayan yordam
    public void çizgiCiz(int x1, int y1, int x2, int y2) {
        // noktaCiz(x,y) // yordamını kullanarak ekrana çizgi çiz
    }
}

class CepTelefonuCizim extends Cizim {
    // iptal ediyor (override)
    public void noktaCiz(int x, int y) {
        // cep telefonu ekranı için nokta çiz....
    }
}

class MonitorCizim extends Cizim {
    // iptal ediyor (override)
    public void noktaCiz(int x, int y) {
        // Monitor ekranı için nokta çiz....
    }
}

public class CizimProgrami {
    public void baslat(int x1, int y1, int x2, int y2) {
        // cep telefonunun ekranına çizgi çizmek için
        Cizim c1 = new CepTelefonuCizim();
        c1.cizgiCiz(x1, y1, x2, y2);
        // Monitor ekranına çizgi çizmek için
        Cizim c2 = new MonitorCizim();
        c2.cizgiCiz(x1, y1, x2, y2);
    }
}
```

Konular

- Polimorfizm
- Geç Bağlama
- Final ve Geç Bağlama
- Polimorfizm ve Tekrar Kullanılabilirlik
- Genişletilebilirlik
- Soyut Sınıflar ve Yordamlar
- Polimorfizm ve Yapılandırıcılar
- Kalıtım ve Yukarıya Çevirim (Upcasting)
- Aşağıya Çevirim (Downcasting)

Polimorfizm ve Yapılandırıcılar

- Futbolcu sınıfında `calis()` yordamı yapılandırıcıdan önce çalışmaktadır!!
- Futbolcu sınıfının `calis()` yordamı çalıştığında `antreman_sayisi` değerini almamıştır. Java 0 değerini (default value) atamıştır.

`calis()` cagrilmadan evvel
`Futbolcu calis()` 0
`calis()` cagrildiktan sonra
`Futbolcu.yapilandirici`
`Futbolcu calis()` 4

```
abstract class Sporcu {  
    public abstract void calis();  
    public Sporcu() { // yapilandirici yordam  
        System.out.println("calis() cagrilmadan evvel");  
        calis(); // ! Dikkat !  
        System.out.println("calis() cagrildiktan sonra");  
    }  
}  
  
class Futbolcu extends Sporcu {  
    int antreman_sayisi = 4;  
    public void calis() {  
        System.out.println("Futbolcu calis() " + antreman_sayisi);  
    }  
    public Futbolcu() { // yapilandirici yordam  
        System.out.println("Futbolcu.yapilandirici");  
        calis();  
    }  
}  
  
public class Spor {  
    public static void main(String args[]) {  
        Futbolcu f = new Futbolcu();  
        // Sporcu s = new Sporcu(); // ! Hata soyut sinif !  
    }  
}
```

Konular

- Polimorfizm
- Geç Bağlama
- Final ve Geç Bağlama
- Polimorfizm ve Tekrar Kullanılabilirlik
- Genişletilebilirlik
- Soyut Sınıflar ve Yordamlar
- Polimorfizm ve Yapılandırıcılar
- Kalıtım ve Yukarıya Çevirim (Upcasting)
- Aşağıya Çevirim (Downcasting)

Kalıtım ve Yukarıya Çevirim (Upcasting)

- **Yukarı çevirim (upcasting) güvenli yöntemdir**, daha çok özelliğe sahip bir tipten daha genel bir tipe doğru çevirim gerçekleştirilir.

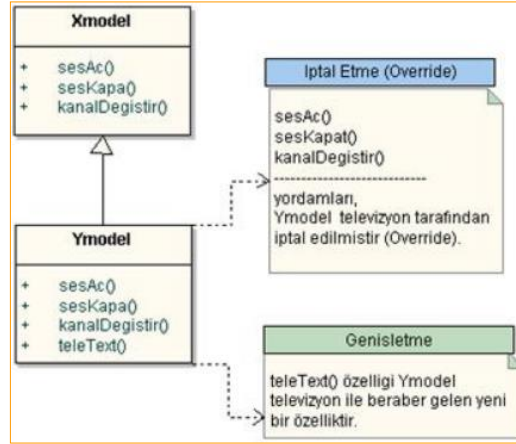
```
class Xmodel {
    public void sesAc() {
        System.out.println("X model televizyon sesAc()");
    }
    public void sesKapa() {
        System.out.println("X model televizyon sesKapa()");
    }
    public void kanalDegistir() {
        System.out.println("X model televizyon kanalDegistir()");
    }
}

class Ymodel extends Xmodel {
    public void sesAc() { // iptal etme (override)
        System.out.println("Y model televizyon sesAc()");
    }
    public void sesKapa() { // iptal etme (override)
        System.out.println("Y model televizyon sesKapa()");
    }
    public void kanalDegistir() { // iptal etme (override)
        System.out.println("Y model televizyon kanalDegistir() ");
    }
    public void teleText() {
        System.out.println("Y model televizyon teleText()");
    }
}
```

```
public class Televizyon {
    public static void main(String args[]) {
        // yukarıya çevirme
        Xmodel x_model_kumanda = new Ymodel();
        x_model_kumanda.sesAc();
        x_model_kumanda.sesKapa();
        x_model_kumanda.kanalDegistir();
        // !! hata !!, bu kumandanın böyle bir düğmesi yok :)
        // x_model_kumanda.teleText();
    }
}
```

```
Y model televizyon sesAc()
Y model televizyon sesKapa()
Y model televizyon kanalDegistir()
```

Kalıtım ve Yukarıya Çevirim (Upcasting)



29

Konular

- Polimorfizm
- Geç Bağlama
- Final ve Geç Bağlama
- Polimorfizm ve Tekrar Kullanılabilirlik
- Genişletilebilirlik
- Soyut Sınıflar ve Yordamlar
- Polimorfizm ve Yapılandırıcılar
- Kalıtım ve Yukarıya Çevirim (Upcasting)
- Aşağıya Çevirim (Downcasting)

Aşağıya Çevirim (Downcasting)

- Aşağıya çevirim (downcasting), yukarı çevirim (upcasting) işleminin tam tersidir.
- Aşağıya çevirim, daha genel bir tipten, daha özellikli bir tipe doğru geçiş demektir.
- Java programlama dilinde aşağıya çevirim yaparken, hangi tipe doğru çevirim yapılacağı açık olarak belirtmelidir.
- Yukarı çevirim (upcasting) işleminde böyle bir belirteç koyma zorunluluğu yoktur.

31

Aşağıya Çevirim (Downcasting)

- Object tipindeki dizi elemanlarını Xmodel nesnesine dönüştürmek için downcasting yapılmıştır.

```
// asagiya cevrim (Downcasting)
Xmodel x_model_kumanda = (Xmodel) ob[i];
```

```
class Xmodel {
    public void sesAc() {
        System.out.println("X model televizyon sesAc()");
    }
    public void sesKapa() {
        System.out.println("X model televizyon sesKapa()");
    }
    public void kanalDegistir() {
        System.out.println("X model televizyon kanalDegistir()");
    }
}

class Ymodel extends Xmodel {
    public void sesAc() { // iptal etme (override)
        System.out.println("Y model televizyon sesAc()");
    }
    public void sesKapa() { // iptal etme (override)
        System.out.println("Y model televizyon sesKapa()");
    }
    public void kanalDegistir() { // iptal etme (override)
        System.out.println("Y model televizyon kanalDegistir() ");
    }
    public void teleText() {
        System.out.println("Y model televizyon teleText()");
    }
}
```

```
public class Televizyon2 {
    public static void main(String args[]) {
        Object[] ob = new Object[2];
        ob[0] = new Xmodel(); // yukari cevrim (upcasting)
        ob[1] = new Ymodel(); // yukari cevrim (upcasting)
        for (int i = 0; i < ob.length; i++) {
            // asagiya cevrim (downcasting)
            Xmodel x_model_kumanda = (Xmodel) ob[i];
            x_model_kumanda.sesAc();
            x_model_kumanda.sesKapa();
            x_model_kumanda.kanalDegistir();
            // x_model_kumanda.teleText(); // bu kumanda da böyle bir düğme yok
            System.out.println("-----");
        }
    }
}
```

```
X model televizyon sesAc()
X model televizyon sesKapa()
X model televizyon kanalDegistir()
-----
Y model televizyon sesAc()
Y model televizyon sesKapa()
Y model televizyon kanalDegistir()
```

32

Aşağıya Çevirim (Downcasting)

- Aynı örnek **çalışma anında tip tanımlama (Run Time Type Identification - RTTI)** ile yapılabilir.

```
if (o instanceof Ymodel) { // RTTI
    Ymodel y_model_kumanda = (Ymodel) o;
```

```
class Xmodel {
    public void sesAc() {
        System.out.println("X model televizyon sesAc()");
    }
    public void sesKapa() {
        System.out.println("X model televizyon sesKapa()");
    }
    public void kanalDegistir() {
        System.out.println("X model televizyon kanalDegistir()");
    }
}

class Ymodel extends Xmodel {
    public void sesAc() { // iptal ediyor (override)
        System.out.println("Y model televizyon sesAc()");
    }
    public void sesKapa() { // iptal ediyor (override)
        System.out.println("Y model televizyon sesKapa()");
    }
    public void kanalDegistir() { // iptal ediyor (override)
        System.out.println("Y model televizyon kanalDegistir()");
    }
    public void teleText() {
        System.out.println("Y model televizyon teleText()");
    }
}
```

```
public class Televizyon {
    public static void main(String args[]) {
        Object[] ob = new Object[2];
        ob[0] = new Xmodel();
        ob[1] = new Ymodel();
        for (int i = 0; i < ob.length; i++) {
            Object o = ob[i];
            if (o instanceof Ymodel) { // RTTI
                Ymodel y_model_kumanda = (Ymodel) o; // artık güvende
                y_model_kumanda.sesAc();
                y_model_kumanda.sesKapa();
                y_model_kumanda.kanalDegistir();
                y_model_kumanda.teleText();
            } else if (o instanceof Xmodel) { // RTTI
                Xmodel x_model_kumanda = (Xmodel) o; // artık güvenli
                x_model_kumanda.sesAc();
                x_model_kumanda.sesKapa();
                x_model_kumanda.kanalDegistir();
            }
        }
    }
}
```

```
X model televizyon sesAc()
X model televizyon sesKapa()
X model televizyon kanalDegistir()
Y model televizyon sesAc()
Y model televizyon sesKapa()
Y model televizyon kanalDegistir()
Y model televizyon teleText()
```

33

Aşağıya Çevirim (Downcasting)

- Aynı örnek daha kısa yazılabilir.

```
// asagiya cevrim (Downcasting)
Xmodel x_model_kumanda = (Xmodel) ob[i];
```

```
class Xmodel {
    public void sesAc() {
        System.out.println("X model televizyon sesAc()");
    }
    public void sesKapa() {
        System.out.println("X model televizyon sesKapa()");
    }
    public void kanalDegistir() {
        System.out.println("X model televizyon kanalDegistir()");
    }
}

class Ymodel extends Xmodel {
    public void sesAc() { // iptal etme (override)
        System.out.println("Y model televizyon sesAc()");
    }
    public void sesKapa() { // iptal etme (override)
        System.out.println("Y model televizyon sesKapa()");
    }
    public void kanalDegistir() { // iptal etme (override)
        System.out.println("Y model televizyon kanalDegistir()");
    }
    public void teleText() {
        System.out.println("Y model televizyon teleText()");
    }
}
```

```
public class Polimorfizm {
    public static void main(String args[]) {
        Object[] ob = new Object[2];
        ob[0] = new Xmodel(); // yukari cevrim (upcasting)
        ob[1] = new Ymodel(); // yukari cevrim (upcasting)
        Xmodel x_model_kumanda = new Xmodel();
        for (int i = 0; i < ob.length; i++) {
            // asagiya cevrim (Downcasting)
            if (ob[i] instanceof Ymodel)
                x_model_kumanda = (Ymodel) ob[i];
            else
                x_model_kumanda = (Xmodel) ob[i];
            x_model_kumanda.sesAc();
            x_model_kumanda.sesKapa();
            x_model_kumanda.kanalDegistir();
            if (x_model_kumanda instanceof Ymodel)
                ((Ymodel) x_model_kumanda).teleText();
            System.out.println("-----");
        }
    }
}
```

```
X model televizyon sesAc()
X model televizyon sesKapa()
X model televizyon kanalDegistir()
-----
Y model televizyon sesAc()
Y model televizyon sesKapa()
Y model televizyon kanalDegistir()
Y model televizyon teleText()
-----
```

34