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

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Kompozisyon

- Uygulamalarda önceden yazılmış **sınıfları tekrar kullanmak için iki yöntem vardır:**
  - Kompozisyon
  - Kalıtım (inheritance)
- **Kompozisyon** ile önceden yazılmış sınıflar yeni yazılan sınıfın içerisinde doğrudan kullanılabilir.
- **Kalıtım** ile yeni bir sınıf önceden yazılmış başka bir sınıftan türetilebilir.
- Yeni türetilen bir sınıf **türetildiği sınıfın özelliklerine sahip olur.**
- Yeni türetilen bir sınıf **türetildiği sınıfın özelliklerini değiştirerek kullanabilir.**
- Yeni türetilen sınıfın kendisine ait **yeni özellikleri de tanımlanabilir.**

3

Kompozisyon

- Aşağıdaki örnekte **`Elma` sınıfı**, **`Meyve`** sınıfını doğrudan kendi içerisinde tanımlayarak, **`Meyve` sınıfının içerisindeki erişilebilir olan özellikleri kullanabilir.**
- Burada, **`Elma` sınıfının içinde `Meyve` sınıfından bir nesne oluşturulmuştur.**

```
class Meyve {

    // ...

}
```

```
class Elma {

    private Meyve m = new Meyve();

    // ...

}
```

4

Kompozisyon

- **`AileArabasi`** içinde **`Motor`** sınıfı ile nesne oluşturuldu.
- **`AileArabası`** sınıfının **`hareketEt()`** ve **`dur()`** metotlarında, önce **`Motor`** sınıfına ait yordamlar çağrılmıştır.
- Motor sınıfında **`private`** **olan** **`motor_gucu`** **alanına erişilemez.**

```
package ErisimOrnekler;

public class Motor {
    private static int motor_gucu = 3600;

    public void calis() {
        System.out.println("Motor Calisiyor");
    }

    public void dur() {
        System.out.println("Motor Durdu");
    }
}
```

```
Motor Calisiyor
Aile Arabasi Calisti
Motor Durdu
Aile Arabasi Durdu
```

```
package ErisimOrnekler;

public class AileArabasi {
    private Motor m = new Motor();

    public void hareketEt() {
        m.calis();
        System.out.println("Aile Arabasi Calisti");
    }

    public void dur() {
        m.dur();
        System.out.println("Aile Arabasi Durdu");
    }

    public static void main(String args[]) {
        AileArabasi aa = new AileArabasi();
        aa.hareketEt();
        aa.dur();
    }
}
```

Kompozisyon

- **`Voltran`** isimli robot **6 farklı sınıf ile oluşturulmuştur.**

```
class Govde {
    void benzinTankKontrolEt() {
    }
}

class SolBacak {
    void maviLazerSilahiAtesle() {
    }
}

class SagBacak {
    void kirmiziLazerSilahiAtesle() {
    }
}

class SagKol {
    void hedeHodoKalkaniCalistir() {
    }
}

class SolKol {
    void gucKaynagiKontrolEt() {
    }
}
```

```
class Kafa {
    void tumBirimlereUyariGonder() {
    }

    void dusmanTanimlamaSistemiDevreyeSok() {
    }
}

public class Voltran {
    Govde gv = new Govde();
    SolBacak slb = new SolBacak();
    SagBacak sgb = new SagBacak();
    SagKol sgk = new SagKol();
    SolKol slk = new SolKol();
    Kafa kf = new Kafa();

    public static void main(String args[]) {
        Voltran vr = new Voltran();
        vr.kf.dusmanTanimlamaSistemiDevreyeSok();
        vr.kf.tumBirimlereUyariGonder();
        vr.sgb.kirmiziLazerSilahiAtesle();
    }
}
```

6

Konular

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Kalıtım

- Kalıtım (inheritance), **nesneye yönelik programlamanın en önemli kavramlarındandır.**
- Kalıtım **bir sınıftan diğer bir sınıfın türetilmesidir.**
- Yeni türetilen sınıf, **türetildiği sınıfın global alanlarına** ve **yordamlarına** (statik olsa dahi) otomatik olarak **sahip olur** (private olanlara doğrudan erişim yapamaz.).
- Yeni **türetilen sınıf**, türetildiği sınıfın **private global alanlarına ve yordamlarına** (statik olsa dahi) **erişim hakkına sahip olamaz.**
- Yeni **türetilen sınıf** türetildiği sınıfın sadece **public ve protected** global alanlar ve yordamlara **erişim hakkına sahip olur.**

Kalıtım

- Aşağıdaki **örnekte**, **her Kaplan bir Kedi'dir.**
- **Her Kaplan Kedi'nin özelliklerini taşıyacaktır.**
- **Her Kaplan** Kedi'nin özelliklerinin üzerine **kendisine ait özellikleri ekleyebilir.**
- Her sınıfın içerisine main yordamı yazılarak tek başlarına çalışabilir hale getirilebilir.

```
class Kedi {

    // ..

}

class Kaplan extends Kedi {

    // ..

}
```

9

Kalıtım

- **Kaplan** sınıfı **`yakalaAv()`** ve **`ayaksayisi`** özelliklerini **Kedi sınıfından miras almıştır.**
- Kedi sınıfının **`ayaksayisi`** alanı **`protected`** erişime sahiptir.
- Protected üyelere **türetilmiş sınıflar erişebilir.**

```
package KalitimOrnekler;

class Kedi {
    protected int ayakSayisi = 4;

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

    public static void main(String args[]) {
        Kedi kd = new Kedi();
        kd.yakalaAv();
    }
}

class Kaplan extends Kedi {
    public static void main(String args[]) {
        Kaplan kp = new Kaplan();
        kp.yakalaAv();
        System.out.println("Ayak Sayisi = " + kp.ayakSayisi);
    }
}
```

Konular

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Gizli Kalıtım

- Oluşturulan **her yeni sınıf** otomatik ve gizli olarak **Object sınıfından türetilir.**
- **Object sınıfı** Java programlama dili içerisinde kullanılan **tüm sınıfların atasıdır (base class).**
- Örnekte, **`toString()`** ve **`equals()`** **yordamları** olmamasına rağmen bu yordamları **Object class'ından alır.**
- **Yeni bir sınıf tanımlandığında**, **Java** gizli ve otomatik olarak **extends Object** yapar.

```
public class Kalitim {

    public static void main(String[] args) {

        Kalitim ybs1 = new Kalitim();
        Kalitim ybs2 = new Kalitim();

        System.out.println("YeniBirSinif.toString()" + ybs1);
        System.out.println("YeniBirSinif.toString()" + ybs2);
        System.out.println("ybs1.equals(ybs2)" + ybs1.equals(ybs2));

    }
}
```

```
public class YeniBirSinif extends Object {
```

```
YeniBirSinif.toString()KalitimOrnekler.Kalitim@4554617c
YeniBirSinif.toString()KalitimOrnekler.Kalitim@74a14482
ybs1.equals(ybs2)false
```

12

Konular

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Kalıtım ve İlk Değer Alma Sırası

- Kalıtım, bir sınıftan başka bir sınıfı **kopyalamak değildir.**
- **Kalıtım**, türeyen bir sınıfın **türetildiği sınıfa ait özellikleri almasıdır**, ayrıca **kendisine ait özellikleri tanımlayabilmesidir**.
- Bir **sınıfa ait nesne oluşurken**, ilk **önce bu sınıfa ait yapılandırıcı (constructor) çağrılır.**
- Bir **nesne oluşmadan önce**, **türetildiği sınıfın nesnesi oluşturulmaya çalışılır.**
- Bu oluşturulma işlemi türetilen sınıflarla zincirleme devam eder **en sonunda Object sınıfına ulaşılır.**
- **Önce Object sınıfından nesne oluşturulur** ve sonra sırayla **türetilene doğru devam eder.**

Kalıtım ve İlk Değer Alma Sırası

- **`UcanYarasa`** nesnesinden önce, **`Yarasa`** sınıfının, ondan önce **`Hayvan`** sınıfının yapılandırıcısı çalıştırılacaktır.

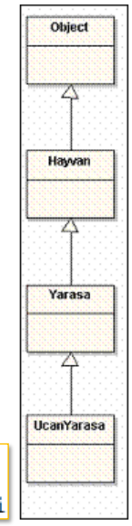
```
class Hayvan {
    public Hayvan() {
        System.out.println("Hayvan Yapilandiricisi");
    }
}

class Yarasa extends Hayvan {
    public Yarasa() {
        System.out.println("Yarasa Yapilandiricisi");
    }
}

class UcanYarasa extends Yarasa {
    public UcanYarasa() {
        System.out.println("UcanYarasa Yapilandiricisi");
    }

    public static void main(String args[]) {
        UcanYarasa uy = new UcanYarasa();
    }
}
```

```
Hayvan Yapilandiricisi
Yarasa Yapilandiricisi
UcanYarasa Yapilandiricisi
```



Konular

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Parametre Alan Yapılandırıcılar ve Kalıtım

- Ana sınıfa ait **yapılandırıcı çağırma işlemi**, **varsayılan yapılandırıcılar için otomatik olarak yürütülür.**
- Ancak, **parametre alan yapılandırıcılar için doğrudan çağırmak gerekir.**
- **Ana sınıfın parametre alan yapılandırıcısını** açık olarak **`super` anahtar kelimesi ile çağırmak gerekir.**
- **`this`** anahtar kelimesinin kullanılışında olduğu gibi, **`super`** anahtar kelimesi de **içinde bulunduğu yapılandırıcının ilk satırında yer almalıdır.**

17

Parametre Alan Yapılandırıcılar ve Kalıtım

```
class Insan {
    public Insan(int par) {
        System.out.println("Insan Yapilandiricisi " + par);
    }
}

class ZekiInsan extends Insan {
    public ZekiInsan(int par) {
        super(par + 1); // dikkat
        System.out.println("ZekiInsan Yapilandiricisi " + par);
    }
}

class Hacker extends ZekiInsan {
    public Hacker(int par) {
        super(par + 1); // dikkat
        System.out.println("Hacker Yapilandiricisi " + par);
    }

    public static void main(String args[]) {
        Hacker hck = new Hacker(5);
    }
}
```

```
Insan Yapilandiricisi 7
ZekiInsan Yapilandiricisi 6
Hacker Yapilandiricisi 5
```

18

Parametre Alan Yapılandırıcılar ve Kalıtım

- **`super`** ile ilk satırda olmayan hatalı çağırma.

```
class Insan2 {
    public Insan2(int par) {
        System.out.println("Insan2 Yapilandiricisi " + par);
    }
}

class ZekiInsan2 extends Insan2 {
    public ZekiInsan2(int par) {
        System.out.println("ZekiInsan2 Yapilandiricisi " + par);
        super(par + 1); // 2. satira yaziliyor ! hata !
    }
}

class Hacker2 extends ZekiInsan2 {
    public Hacker2(int par) {
        System.out.println("Hacker2 Yapilandiricisi " + par);
        System.out.println(".......selam..........");
        super(par + 1); // 3. satira yaziliyor ! hata !
    }

    public static void main(String args[]) {
        Hacker2 hck2 = new Hacker2(5);
    }
}
```

Constructor call must be the first statement in a constructor

19

Konular

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Kompozisyon ve Kalıtım

- Yeni oluşturulan sınıfın içerisinde, **önceden yazılmış sınıfların özelliklerinden faydalanmak** için **kompozisyon** ve **kalıtım** kullanılabilir.
- **Kompozisyon**, önceden yazılmış sınıfların özelliklerini kullanmak için **basit bir yöntemdir.**
- **Kalıtım**, önceden yazılmış bir **sınıfın**, belli bir problem için **yeni versiyonunu yazma işleminde kullanılabilir.**
- **Kalıtımda türetilen** sınıf ile **türeyen** sınıf **arasında** bir **ilişki olmalıd**ır (Kedi ve Kaplan, Bisiklet ve Motosiklet).

21

Kompozisyon ve Kalıtım

```
class ArabaMotoru {
    public void calis() {
    }
    public void dur() {
    }
}

class Pencere {
    public void asagiyaCek() {
    }
    public void yukariyaCek() {
    }
}

class Kapi {
    Pencere pencere = new Pencere();
    public void ac() {
    }
    public void kapa() {
    }
}

class Tekerlek {
    public void havaPompala(int olcek) {
    }
}
```

```
public class Araba {
    ArabaMotoru arbm = new ArabaMotoru();
    // 2 kapili spor bir araba olsun
    Kapi sag_kapi = new Kapi();
    Kapi sol_kapi = new Kapi();
    Tekerlek[] tekerlekler = new Tekerlek[4];

    public Araba() {
        for (int i = 0; i < 4; i++) {
            tekerlekler[i] = new Tekerlek();
        }
    }

    public static void main(String args[]) {
        Araba araba = new Araba();
        araba.sag_kapi.pencere.yukariyaCek();
        araba.tekerlekler[2].havaPompala(70);
    }
}
```

22

Konular

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Overriding

- Ana sınıf içerisinde tanımlanmış **bir yordam**, ana sınıftan **türetilen bir alt sınıfın içerisinde iptal edilebilir** (overriding).
- **Türeyen bir sınıf türetildiği sınıfa ait global alanları ve yordamları kullanabilir.**
- Ana sınıfa ait **private** erişim belirleyicisine sahip olan **alanlara ve yordamlara**, **türeyen alt sınıf tarafından erişilemez.**
- **Türetilen** alt **sınıf, türetildiği** ana **sınıf ile aynı paket içerisinde değilse**, ana sınıfa ait **friendly** olan **alanlara** ve **yordamlara erişemez.**
- **protected** erişim belirleyicisine sahip **alanlara** ve **yordamlara erişebilir.**

24

Overriding

- **Overriding yapılmamış** sınıf yordamlarının kullanımı.

```
class Kitap {
    public int sayfaSayisiOgren() {
        System.out.println("Kitap - sayfaSayisiOgren() ");
        return 440;
    }

    public double fiyatOgren() {
        System.out.println("Kitap - fiyatOgren() ");
        return 2500000;
    }

    public String yazarIsmiOgren() {
        System.out.println("Kitap - yazarIsmiOgren() ");
        return "xy";
    }
}

class Roman extends Kitap {

    public static void main(String args[]) {
        Roman r = new Roman();
        int sayfasayisi = r.sayfaSayisiOgren();
        double fiyat = r.fiyatOgren();
        String yazar = r.yazarIsmiOgren();
    }
}
```

```
Kitap - sayfaSayisiOgren()
Kitap - fiyatOgren()
Kitap - yazarIsmiOgren()
```


Overriding

- **`sayfaSayisiOgren()`** ve **`fiyatOgren()`** yordamları hem ana sınıfta hem de türetilen sınıfta yazılmıştır.
- Bu yordamlardan türetilen sınıftakiler geçerlidir.

```
class Kitap2 {
    public int sayfaSayisiOgren() {
        System.out.println("Kitap2 - sayfaSayisiOgren() ");
        return 440;
    }

    public double fiyatOgren() {
        System.out.println("Kitap2 - fiyatOgren() ");
        return 2500000;
    }

    public String yazarIsmiOgren() {
        System.out.println("Kitap2 - yazarIsmiOgren() ");
        return "xy";
    }
}
```

```
class Roman2 extends Kitap2 {

    public int sayfaSayisiOgren() {
        System.out.println("Roman2 - sayfaSayisiOgren() ");
        return 569;
    }

    public double fiyatOgren() {
        System.out.println("Roman2 - fiyatOgren() ");
        return 8500000;
    }

    public static void main(String args[]) {
        Roman2 r2 = new Roman2();
        int sayfasayisi = r2.sayfaSayisiOgren();
        double fiyat = r2.fiyatOgren();
        String yazar = r2.yazarIsmiOgren();
    }
}
```

```
Roman2 - sayfaSayisiOgren()
Roman2 - fiyatOgren()
Kitap2 - yazarIsmiOgren()
```

Overriding

- Üst sınıftaki (türetilen) fonksiyona veya üyeye **`super`** ile erişilebilir.
- Bu durumda **`super`** kullanımı ilk satırda olmak zorunda değildir.

```
public int sayfaSayisiOgren() {
    super.sayfaSayisiOgren();
    System.out.println("Roman2 - sayfaSayisiOgren() ");
    return 569;
}
```

```
public int sayfaSayisiOgren() {
    System.out.println("Roman2 - sayfaSayisiOgren() ");
    super.sayfaSayisiOgren();
    return 569;
}
```

27

Overriding

- **İptal eden yordamın**, iptal **edilen yordamın erişim belirleyicisi** ile **aynı** veya **daha yüksek erişilebilir** bir erişim belirleyicisine sahip **olması gereklidir.**
- En yüksekten en düşüğe doğru erişim belirleyicileri:
  - **public:** Her yerden erişilmeyi sağlayan erişim belirleyicisidir.
  - **protected**: Bu sınıftan türetilmiş alt sınıflar tarafından erişilmeyi sağlayan erişim belirleyicisidir.
  - **friendly:** Sadece aynı paket içerisinden erişilmeyi sağlayan erişim belirleyicisidir.
  - **private:** Sadece kendi sınıfı içerisinden erişilmeyi sağlayan, başka her yerden erişimi olmayan erişim belirleyicisidir.
- Ana sınıfa ait **`public a()`** yordamı varsa, türetilen alt sınıfın **`public a()`** yordamını overriding yapması gerekir.
- Ana sınıftaki **`protected`** için **`public`** veya **`protected`** overriding yapılmalıdır.

28

Overriding

- **protected – private** overriding **yanlıştır**.

```
class Telefon {
    protected void aramaYap() {
        System.out.println("Telefon.aramaYap()");
    }
}

class CepTelefonu extends Telefon {
    private void aramaYap() { // ! hatali !
        System.out.println("CepTelefon.aramaYap()");
    }
}
```

Cannot reduce the visibility of the inherited method from Telefon

- **friendly – protected** overriding **doğrudur.**

```
class HesapMakinesi {
    void hesapla(double a, double b) {
        System.out.println("HesapMakinesi.hesapla()");
    }
}

class Bilgisayar extends HesapMakinesi {
    protected void hesapla(double a, double b) {
        System.out.println("HesapMakinesi.hesapla()");
    }
}
```

29

Konular

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Overload

- Ana sınıftaki bir yordama, **türetilen bir alt sınıfın içerisinde adaş yordam yazılabilir** (overload).
- Adaş yordamlarda parametre türleri/sayıları farklı olmalıdır.

```
class Calisan {
    public void isYap(double a) {
        System.out.println("Calisan.isYap()");
    }
}

class Kalitim extends Calisan {
    public void isYap(int a) { // adas yordam (overloaded)
        System.out.println("Mudur.isYap()");
    }

    public static void main(String args[]) {
        Kalitim m = new Kalitim();
        m.isYap(3.3);
        m.isYap(3);
    }
}
```

```
Calisan.isYap()
Mudur.isYap()
```

31

Overload

- Constructor'larda overload yapılabilir.

```
class Calisan {
    public Calisan(int a) {
        System.out.println(a);
    }
    public void isYap(double a) {
        System.out.println("Calisan.isYap()");
    }
}

class Mudur extends Calisan {
    public Mudur(int a, int b) { // adas constructor (overloaded)
        super(a);
        System.out.println(a + b);
    }
    public void isYap(int a) { // adas yordam (overloaded)
        System.out.println("Mudur.isYap()");
    }

    public static void main(String args[]) {
        Mudur m = new Mudur(10, 20);
        m.isYap(3.3);
        m.isYap(3);
    }
}
```

```
10
30
Calisan.isYap()
Mudur.isYap()
```

32

Konular

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Upcasting

- **Türetilen bir sınıf nesnesi türetildiği sınıf nesnesi yerine kullanılabilir** (upcasting).

```
Sporcu.calis()
Futbolcu.calis()
```

```
class KontrolMerkezi {
    public static void checkUp(Sporcu s) {
        // ..
        s.calis();
    }
}

class Sporcu {
    public void calis() {
        System.out.println("Sporcu.calis()");
    }
}

class Futbolcu extends Sporcu {
    public void calis() { // iptal etti (Overriding)
        System.out.println("Futbolcu.calis()");
    }

    public static void main(String args[]) {
        Sporcu s = new Sporcu();
        Futbolcu f = new Futbolcu();
        KontrolMerkezi.checkUp(s);
        KontrolMerkezi.checkUp(f); // dikkat
    }
}
```

Konular

- Kompozisyon
- Kalıtım
- Gizli Kalıtım
- Kalıtım ve İlk Değer Alma Sırası
- Parametre Alan Yapılandırıcılar ve Kalıtım
- Kompozisyon ve Kalıtım
- Overriding
- Overload
- Upcasting
- Final Özelliği

Final Özelliği

- Java programlama dilinde **final** anahtar kelimesi **değiştirilemez olmayı gösterir.**
- **Global alanlara**, **yordamlara** ve **sınıflara** final özelliği uygulanabilir.
- **Global alanlar için** final özelliği **sabit değer özelliğini sağlar.**
- Global sabit alanlar (statik dahi olsa) final özelliğine sahip olabilir.
- **Final global alanlara** sadece **bir kez değer atanabilir.**
- **Global olan final alanlar** ikiye ayrılır:
  - **Derleme anında** değerleri atanan final global alanlar.
  - **Çalışma anında** değerleri atanan final global alanlar.

Final Özelliği

- **`X_SABIT_DEGER`** ve **`Y_SABIT_DEGER`** alanlarına **derleme anında**, **`A_SABIT_DEGER`** alanına **çalışma anında** değeri atanır.
- Nesne final yapılırsa adresi sabitlenir ve başka bir nesneye bağlanamaz.
- **`fo.k = new Kutu();`** hatalıdır.

```
X_SABIT_DEGER = 34
Y_SABIT_DEGER = 35
A_SABIT_DEGER = 49
Kutu.i = 35
Kutu.i = 55
```

```
class Kutu {
    int i = 0;
}

public class Kalitim{
    final int X_SABIT_DEGER = 34;
    final static int Y_SABIT_DEGER = 35;
    final int A_SABIT_DEGER = (int) (Math.random() * 50);
    final Kutu k = new Kutu();

    public static void main(String args[]) {
        Kalitim  fo = new Kalitim ();
        // fo.X_SABIT_DEGER = 15 ! Hata !
        // fo.Y_SABIT_DEGER = 16 ! Hata !
        // fo.A_SABIT_DEGER = 17 ! Hata !
        System.out.println("X_SABIT_DEGER = " + fo.X_SABIT_DEGER);
        System.out.println("Y_SABIT_DEGER = " + fo.Y_SABIT_DEGER);
        System.out.println("A_SABIT_DEGER = " + fo.A_SABIT_DEGER);

        fo.k.i = 35; // doğru
        System.out.println("Kutu.i = " + fo.k.i);
        fo.k.i = 55; // doğru
        System.out.println("Kutu.i = " + fo.k.i);
        // fo.k = new Kutu() ! hata !
    }
}
```

Final Özelliği (final alanlar)

- **Boş final alanlara** ve **nesnelere ilk değerleri yapılandırıcıların içerisinde verilmelidir.**

```
class Kalem {
}

public class BosFinal {
    final int a = 0;
    final int b; // Bos final
    final Kalem k; // Blank final nesne alanı

    // Bos final alanlar ilk değerlerini yapılandırıcılarda içerisinde alırlar
    BosFinal() {
        k = new Kalem();
        b = 1; // bos final alanına ilk değeri ver
    }

    BosFinal(int x) {
        b = x; // bos final alanına ilk değeri ver
        k = new Kalem();
    }

    public static void main(String[] args) {
        BosFinal bf = new BosFinal();
    }
}
```

38

Final Özelliği (final yordamlar)

- **Final yordamlar overriding yapılamazlar.**

```
class A {
    public final void ekranaYaz() {
        System.out.println("A.ekranaYaz()");
    }
}

class B extends A {
    public void ekranaYaz() {
        System.out.println
    }
}
```

Cannot override the final method from A

1 quick fix available:

Remove 'final' modifier of 'A.ekranaYaz'(..)

Press 'F2' for focus

39

Final Özelliği (final yordamlar)

- **private final yordamlar overriding yapılmazlar.**

```
class Polis {
    private final void sucluYakala() { // erişilemez gizli yordam
        System.out.println("Polis.sucluYakala()");
    }
}

class SivilPolis extends Polis {
    public void sucluYakala() { // iptal etme söz konusu değildir
        System.out.println("SivilPolis.sucluYakala()");
    }
}

public class Kalitim {
      public static void main(String[] args) {
          SivilPolis sp = new SivilPolis();
          sp.sucluYakala();
      }
}
```

SivilPolis.sucluYakala()

40

Final Özelliği (final sınıflar)

- **final sınıflardan yeni sınıf türetilemez.**

```
final class Televizyon {
    public void kanalBul() {
    }
}

// final sınıf'tan türetme yapılamaz !!!
class SuperTelevizyon extends Televizyon{

}

class Ev {
    int oda_sayisi = 5;
    Televizyon tv = new Televizyon();

    public static void main(String args[]) {
        Ev e = new Ev();
        e.tv.kanalBul();
    }
}
```

The type SuperTelevizyon cannot subclass the final class Televizyon

1 quick fix available:

Remove 'final' modifier of 'Televizyon'

Press 'F2' for focus