

tart
yeni medya mutfağı



Nesneye Yönelik Programlama

Cihangir SAVAS | SW101

Seminer İerięi

01. Class
02. Eriřim Denetleyiciler
03. Method Overloading
04. Yapılandırıcılar
05. İlk deęer ataması
06. This
07. Static / Final
- 08.



Class Nedir?

Nesneler

- Dizi, aynı tipteki değerlerin bir koleksiyonudur.
- Nesne ise farklı tipteki değerlerin bir koleksiyonudur.
 - Aslında nesne bundan biraz daha karmaşıktır ama bu tanımı sadece karşılaştırma için yapıyoruz.
- Örnek:

```
class Person {  
    String name;  
    boolean male;  
    int age;  
    int phoneNumber;  
}
```
- Bu sebeple farklı tipteki nesneler bellekte farklı boyutta yer alacaktır.

Erişim Belirleyiciler

public: Her yerden erişilmeyi sağlayan erişim belirleyicisi.

protected: Aynı paket içerisinde ve bu sınıftan türemiş alt sınıflar tarafından erişilmeyi sağlayan erişim belirleyicisi.

private: Yalnızca kendi sınıfı içerisinde erişilmeyi sağlayan, başka her yerden erişimi kesen erişim belirleyicisi.

private & public alanlar ve metodlar

- **private** alanlar türetilmiş sınıflarda kullanılamazlar.
 - “Bilgi Gizleme” böyle olması gerektiğini söyler
 - Eğer erişmek gerekiyorsa üst sınıfın ilgili metodları kullanılarak gerekli erişim yapılır.
- **private** metodlarda alt sınıflara kalıtımsal olarak geçmezler.
 - Alt sınıflarda kullanılması gerekiyor ise `public/protected` olarak tanımlanmalıdır
 - Sadece yardımcı olmak için bazı metodlar `private` tanımlanabilir

protected Belirleyicisi

Görülebilir özellikler hangi alanların/metodların kalıtılabileceğini veya kalıtlamayacağını gösterirler public olarak tanımlanan değişkenler ve metodlar kalıtılabilirken, private olanlar görünmeyen özellikler olup kalıtılmamaktadır.

Fakat public alanlar/değişkenler kapsama ilkeline ters olan bir belirleyicidir.

Bu sebeple kalıtım durumunda yardımcı olan bir üçüncü belirleyici geliştirilmiştir : **protected**

protected Belirleyicisi - 2

protected belirleyicisi üst sınıfın bir üyesinin (alan veya metod) alt sınıfa geçmesini sağlar.

Protected şu iki özelliği sağlar

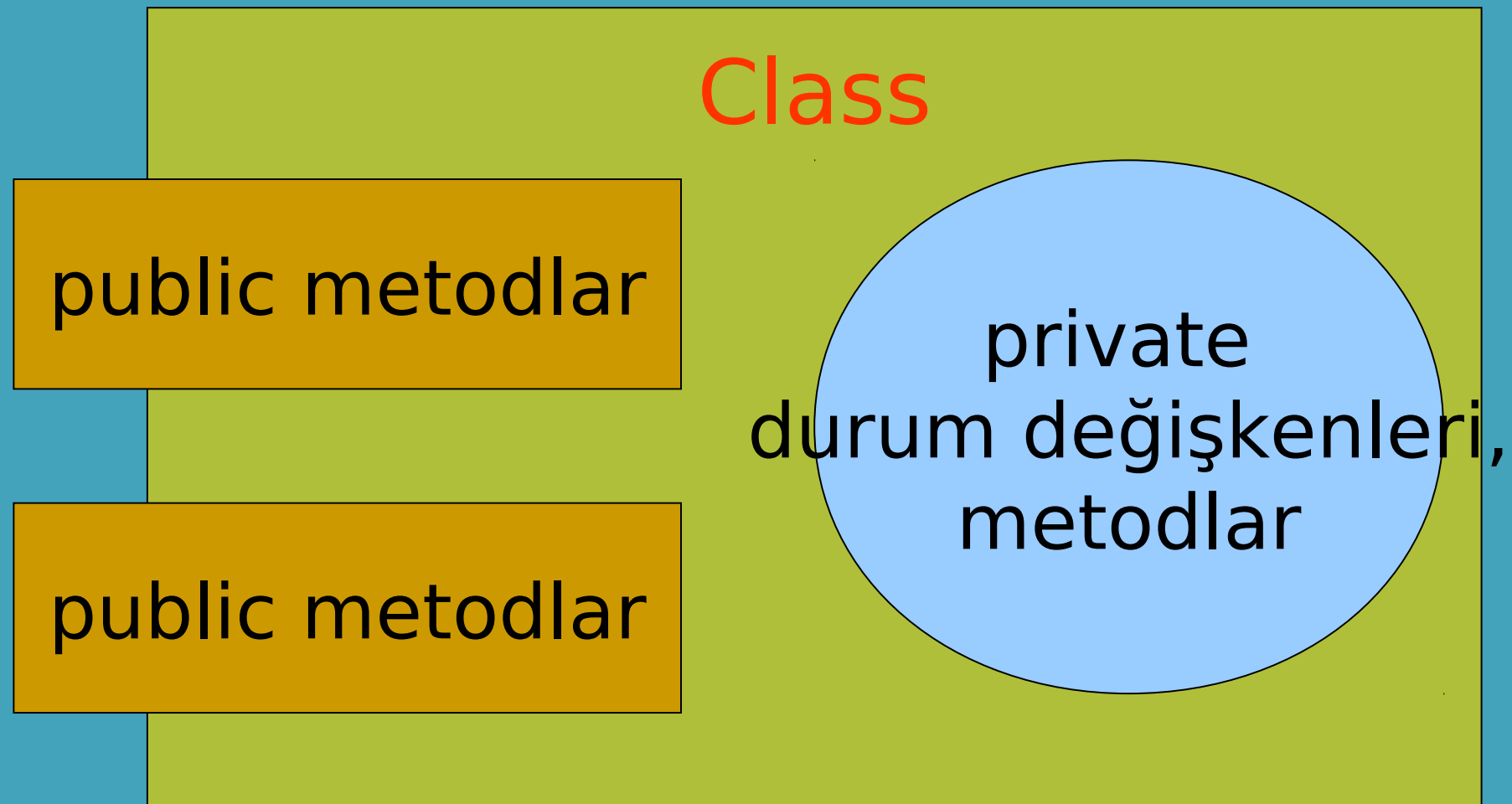
- public ten daha fazla bir kapsama özelliği vardır.
- Kalıtıma izin veren en iyi kapsama imkanını sağlar.

Tasarım Temelleri

- *private* durum değişkenleri kullanılarak özel veriler sınıf yapısı içinde sarmalanır-kapsülленir.
(Encapsulation)
- Sınıf dışından bu verilere erişim public metodlar kullanılarak yapılır.
- *private* metodlar sadece sınıf içinden erişim için kullanılabilir.

Sarmalama-Kapsülleme

Data Encapsulation



Sınıftaki private/public metodlar

```
private String isim, tel;  
private double haftalikmaas, yillikmaas;  
    // aynı sınıf içinden erişim için kullanılır  
private void HesaplaYillikMaas() {  
    yillikmaas = haftalikmaas * 12;  
}  
public void YazdirYillikMaas() {  
    HesaplaYillikMaas(); // aynı sınıf içinden erişim yapılıyor.  
    System.out.println(isim+" nin yıllık maasi " + yillikmaas);  
}  
}
```

public Metodların kullanımı

```
class test {  
    public static void main(String args[]) {  
        Calisan tart=new Calisan();  
        tart.YazdirYillikMaas(); //şeklinde kullanılır  
        tart.HesaplaYillikMaas(); //kullanılmaz  
    }  
}
```

Bilgisayarda Bellek Organizasyonu

- Bir **bit** on/off, yes/no, **0/1** tipinden değerdir
 - Bir **byte** ise sekiz ardışık bitten oluşur
 - Bilgisayar Belleği ise 0 dan başlayan bir dizi byte dan oluşur
 - Bellekteki herhangi bir adresin boş olması diye bir kavram yoktur burada bir şekilde bitler(0/1) saklıdır.
 - Bir byte ilişkilendirilmiş olan numara onun **adresidir**.
-
- Adresler ikili düzendeki sayılardır (genellikle hexadecimal olarak yazılırlar)
 - Bazı büyük bilgisayar sistemleri ise byteları değil wordleri adresler. **Wordler** ise daha fazla sayıda bitten oluşur (such as 32 or 64)

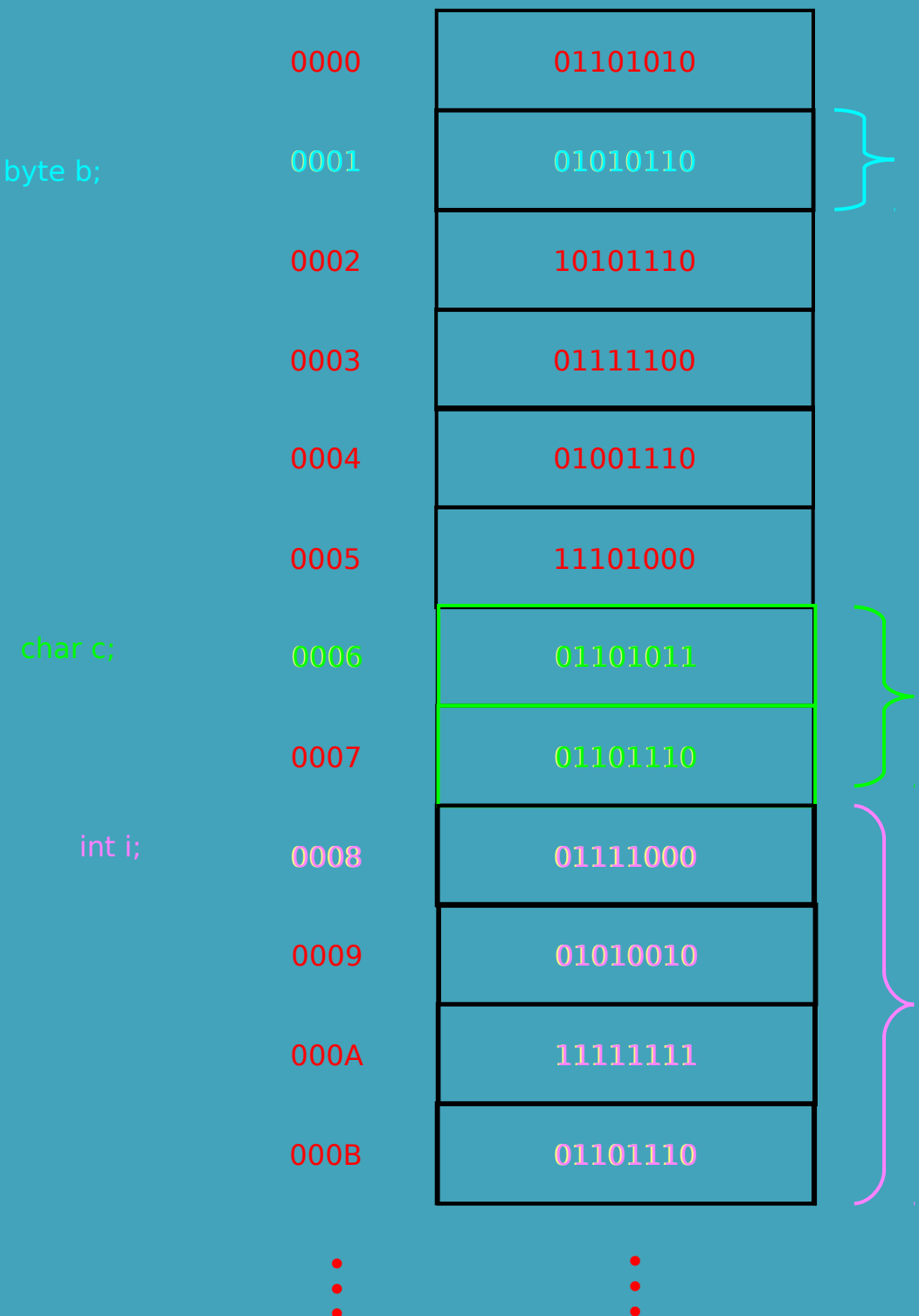
• 0000	• 001101010
• 0001	• 101010110
• 0002	• 10101110
• 0003	• 01111100
• 0004	• 01001110
• 0005	• 11101000
• 0006	• 01101011
• 0007	• 01101110
• 0008	• 01111000
• 0009	• 01010010
• 000A	• 11111111
• 000B	• 01101110

•
•
•

•
•
•

Bellekteki basit değişkenler

- Basit bir değişkeni tanımlayında bellekte ona bir yer ayırırsınız. (Tipine göre)
- Örneğin bir **byte** bellekte tek bir bytelık alana `byte b;` ihtiyaç duyar
- Bir **int** ise dört ardışık byte a ihtiyaç duyar
- Bir **char** iki ardışık byte a ihtiyaç duyar
- Bu yüzden her değişkenin bellekte ayrı bir adresi vardır.



Bellekte Nesne Saklama

- Bir nesneyi tanımladığımız zaman ise Derleyici bu nesnenin ne kadar belleğe ihtiyacı olduğunu bilemez. Bu sebeple bellek ataması yapamaz.
- Gerekli bellek miktarı o nesneyi yarattığınızda bellekte tanımlanır ve atanır.

0000	01101010
0001	01010110
0002	10101110
0003	01111100
0004	01001110
0005	11101000
0006	01101011
0007	01101110
0008	01111000
0009	01010010
000A	11111111
000B	01101110
⋮	⋮

Nesnenin Tanımlanması ve Yaratılması

Person p = new Person("John");

Person p işaretçisi için bellekte bir yer tanımlar.

new Person("John"), Person sınıfının yapısına uygun bellekte bir yer ayırır.

Atama komutu ile p ye gerekli referans ataması gerçekleşir.

p:

07A3

07A3

"John"

Atama İşlemleri

- Basit Değişkenlerde veri atamalarda *değer* ataması yapılır
- Nesnelerin atanmasında ise *referansların kopyalaması* yapılır.
- Bu sebeple iki nesne aynı veri bloğuna işaret edebilir.
 - Bunu başka bir isimle adlandırma olarakta düşünebilirsiniz.
 - Eğer bir nesnenin işaret ettiği bir değeri değiştirirseniz diğer nesnede değişecektir.

Parametrelerde Veri transferi

- Bir method çağrısı şu şekilde olur.

```
result = add( 3 , 5 ) ;
```

Parametre değerleri kopyalanarak gönderilir.

- Bunun method tanımı şu şekildedir.

```
int add ( int number1, int number2 ) {  
    int sum = number1 + number2 ;  
    return sum ;  
}
```

Sonuç geriye döner

Method için bunlar kullanılır

Basit Parametreler Kopyalanır.

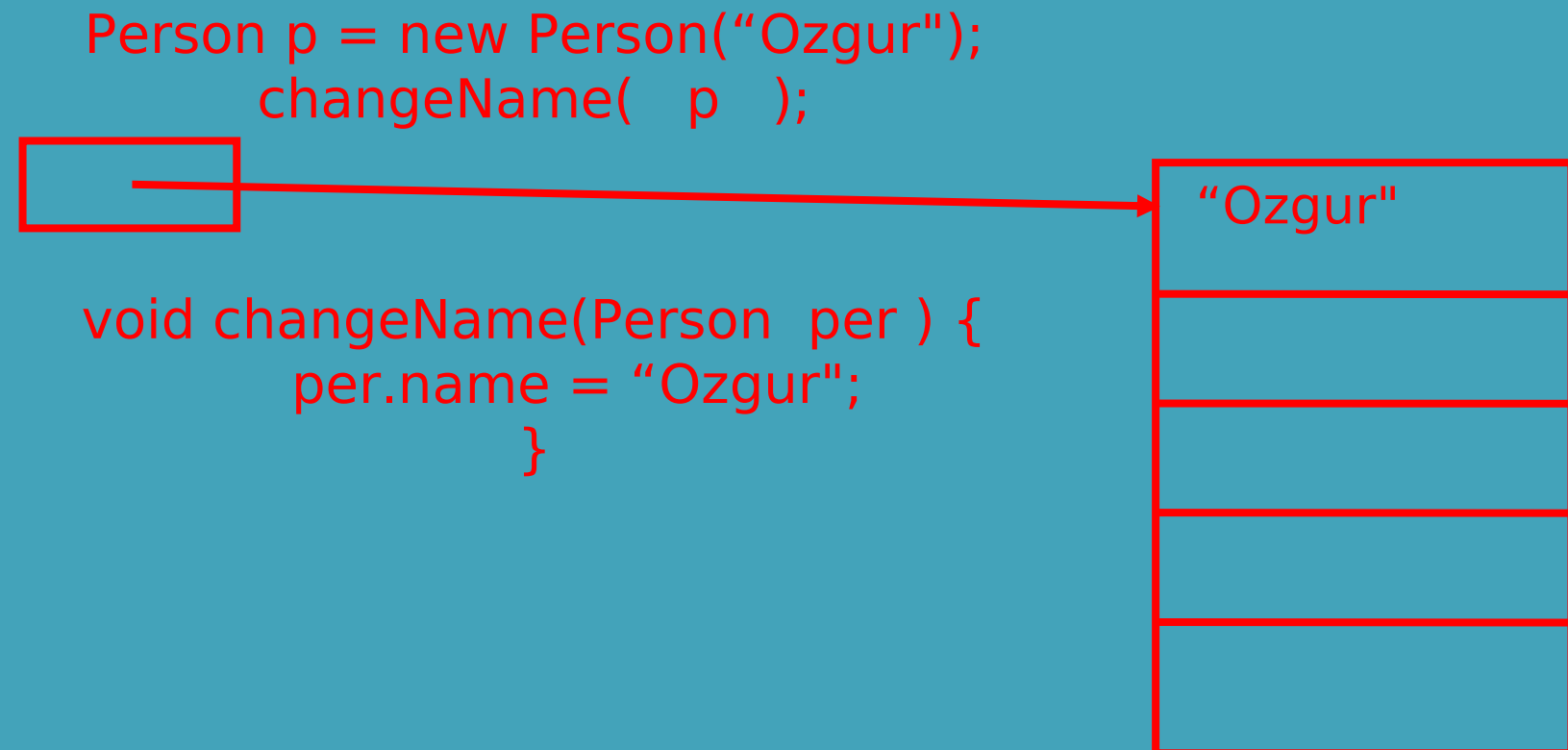
```
int m = 3;  
int n = 5;  
result = add( m , n );
```

```
int add ( int number1 , int number2 ) {  
    while (number1 > 0) {  
        number1--;  
        number2++;  
    }  
    return number2 ;  
}
```

- Buna **değer ile çağırma** (call by value) denir

Nesnelerde ise *referenslar* gönderilir

p , Person in referansını tutar.



- *p* ve *per* aynı nesneyi işaret etmektedir.!
- Bu sebeple *p* de yapılacak değişiklik diğerini de etkileyecektir.
- Buna referans ile çağırma (*call by reference*) denir

NullPointerException

- Bir nesne tanımladığınız zaman onun ilk değeri null (0000) dır.
- null her nesne tanımlaması için kullanılabilen bir değerdir.
 - Bir nesnenin değerinin null olup olmadığını kontrol edebilirsiniz (if (x==null))
a*****)
 - Bir nesneye null değer atayabilirsiniz. (x=null;)
 - null değere sahip bir nesneyi başka şekilde kullanamazsınız.
- Örnek:

Person ozgur;

ozgur.name = "OzgurKoray"; // NullPointerException

Eşitlik

- İki basit değişkenin eşitliğini değerleri üzerinden yapabilirsiniz.

x 24

y 24

- İki nesnenin eşitliğini ise referansları üzerinden yapmanız gerekir.

- $a == b$, fakat $a != c$
- *Stringler için ise equals* metodunu kullanmanız gerekecektir. `s1.equals(s2)`

a

b

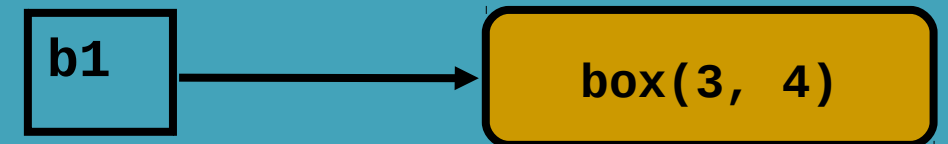
c

OzgurKoray
true
24
5551212

OzgurKoray
true
24
5551212

Nesne Kopyalama

```
Box b1 = new Box(3, 4);
```



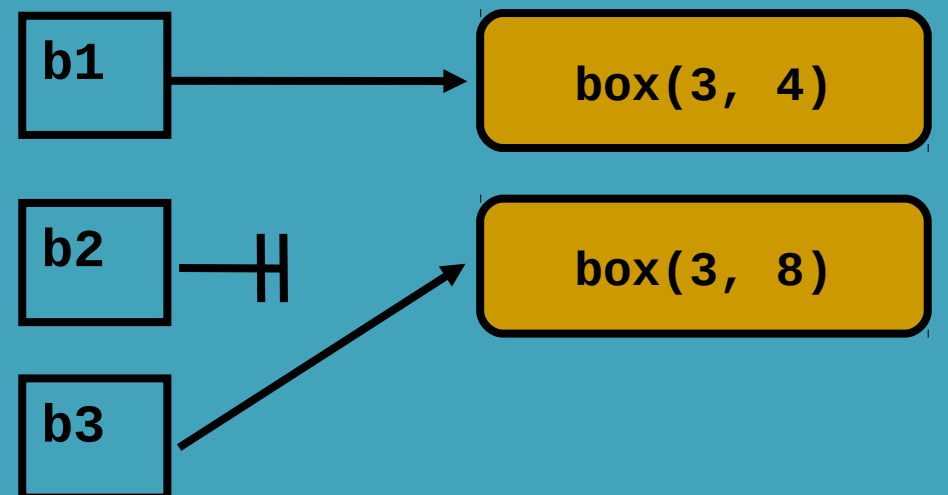
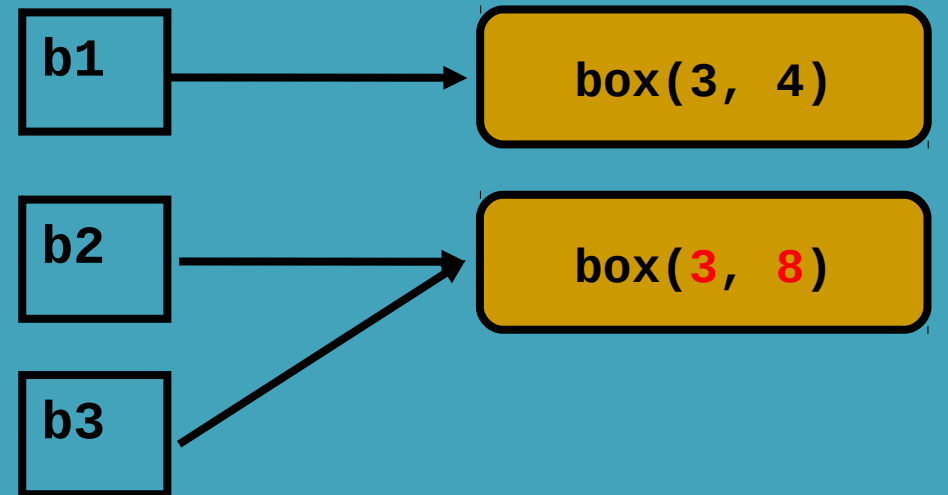
```
Box b2 = b1.duplicate();
```

```
Box b3 = b2;
```

```
b2.setWid(8);
```

```
System.out.println(b3.getWid());
```

```
b2 = null;
```

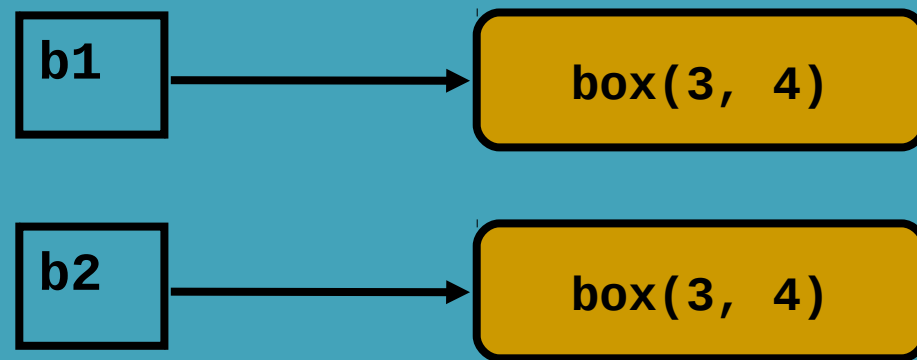


Soru ?

```
Box b1 = new Box(3, 4);  
Box b2 = new Box(3, 4);  
if(b1 == b2)  
    System.out.println("Bu Kutu Nesneleri Eşittir!");  
else  
    System.out.println("Nasıl Eşit Olsunlar ki!.... ");
```


Eşitlik

- Aynı Bellek alanına işaret etmedikleri için eşit değildir.



Eşitlik-String

```
class test {  
    public static void main (String args[]) {  
        String str1= new String("Ozgur");  
        String str2= new String("Ozgur");  
  
        System.out.println ("Ozgur" == "Ozgur");  
        System.out.println (str1 == "Ozgur");  
        System.out.println (str1 == str2);  
        System.out.println (str1.equals(str2));  
        System.out.println (str1.equalsIgnoreCase(str2));  
    }  
} // Çıktı ? ? ?
```

Cıktı

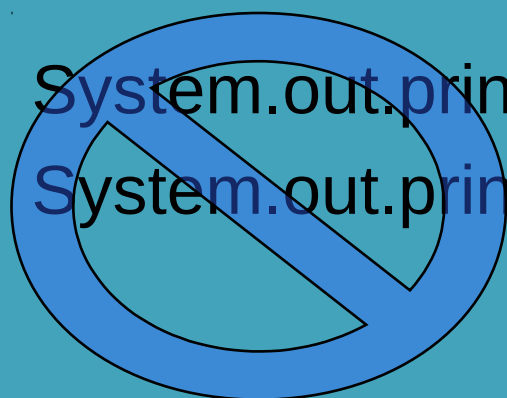
- tue
- false
- false
- true
- true

Adaş Yordamlar-(Metod Overloading)

- Aynı metod ismi vardır. Fakat farklı parametreler gönderilir.
- Örneğin:
 - ❑ bilgiDegistir(String t);
 - ❑ bilgiDegistir(double s);
 - ❑ bilgiDegistir(String t, double s);

Örnek

- `System.out.println(12);`
- `System.out.println("Özgür Koray");`
- `System.out.println('E');`
- `System.out.println(12.0f);`



`System.out.printString("Ozgur Koray");`
`System.out.println(12);`

Adaş Yordamlar

```
public class OverLoad {  
    public void same()  
        { System.out.println( "No arguments" ); }  
    public void same( int firstArgument )  
        { System.out.println( "One int arguments" ); }  
    public void same( char firstArgument )  
        { System.out.println( "One char arguments" ); }  
    public int same( int firstArgument ) // Derleme Hatası  
        { System.out.println( "One char arguments" ); return 5; }  
    public void same( char firstArgument, int secondArgument )  
        { System.out.println( "char + int arguments" ); }  
    public void same( int firstArgument, char secondArgument )  
        { System.out.println( "int + char arguments" ); } }  
}
```

CalisanOverloading

```
public class CalisanOverloading {  
    private String isim;  
    private String tel;  
    private double haftalikmaas;  
    public void bilgiYazdir() {  
        System.out.println( isim + "' nin telefonu " + tel + ", Haftalik Maasi " + haftalikmaas);  
    }  
    public void bilgiDegistir(String t) {  
        tel = t;  
    }  
    public void bilgiDegistir(double s) {  
        haftalikmaas = s;  
    }  
    public void bilgiDegistir(String t, double s) {  
        tel = t;    haftalikmaas = s;  
    }  
}  
  
CalisanOverloading ozgur=new CalisanOverloading("Ozgur Koray SAHINGOZ", "6632490", 1234.50);  
ozgur.bilgiDegistir("6632491");  
ozgur.bilgiDegistir(1500.00);  
ozgur.bilgiDegistir("555-3824", 1750.80);
```

Örnek?

Gönderilen integer değerlerin en büyüğünü dönderen Max metodu nasıl geliştirilir.

```
int x= max(12,24);
```

```
int x= max(12,24,35);
```

```
int[] dizi = {12,24,34,45,56,67,78}
```

```
int x=max(dizi);
```


Adaş Yapılandırıcı

- Adaş metodların özel bir durumudur.
- Farklı parametreler ile birden fazla tanımlanabilir.

```
public class CalisanOverCons {  
    CalisanOverCons (String n);  
    CalisanOverCons (String n, String t);  
    CalisanOverCons (String n, String t, double s);  
}
```

Adaş Yapılandırıcı

```
public CalisanOverCons(String n) {  
    isim = n; tel = "000-0000"; haftalikmaas = 0.0;  
}  
public CalisanOverCons(String n, String t) {  
    isim = n; tel = t; haftalikmaas = 0.0;  
}  
public CalisanOverCons(String n, String t, double s) {  
    isim = n; tel = t; haftalikmaas = s;  
}
```

Kullanımı

```
CalisanOverCons ozgur = new CalisanOverCons("Ozgur Koray SAHINGOZ");  
ozgur.chanageinfo("555-1234", 1500.00);
```

```
CalisanOverCons jordan = new CalisanOverCons("Michael Jordan", "555-4548");  
jordan.changeinfo(2500.00);
```

```
CalisanOverCons gates = new CalisanOverCons("Bill Gates", "555-4647", 3000.00);
```

Örnek-2

```
public class Point
{
    public int x = 0;
    public int y = 0;
    public Point(int a, int b)    // yapılandırıcı
    {
        x = a;
        y = b;
    }
    public Point(int a)    // yapılandırıcı
    {
        x = a;
        y = 0;
    }
}
```

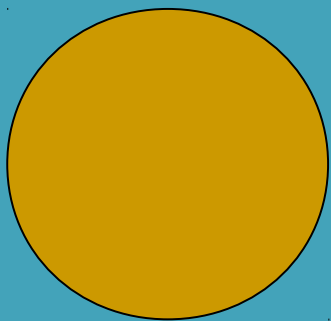
?? Method overloading

Circle Sınıfı ile Kullanım Örneği

```
public class Circle {  
    public double x,y,r;  
    // Constructor  
    public Circle(double centreX, double centreY, double radius)  
    {  
        x = centreX;  
        y = centreY;  
        r = radius;  
    }  
    //Methods to return circumference and area  
    public double circumference() { return 2*3.14*r; }  
    public double area() { return 3.14 * r * r; }  
}
```

Constructor kullanmaz ise

```
Circle aCircle = new Circle();  
aCircle.x = 10.0; // initialize center and radius  
aCircle.y = 20.0  
aCircle.r = 5.0;
```



İlk yaratılma zamanında
çemberin merkezi ve çapı
tanımlı değildir

Bu değerler sonradan atanır.

Çoklu Yapılandırıcılar

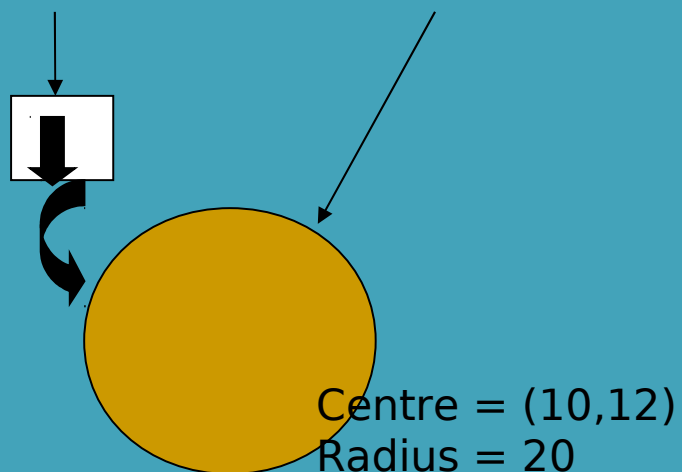
```
public class Circle {  
    public double x,y,r; //instance variables  
    // Constructors  
    public Circle(double centreX, double centreY, double radius) {  
        x = centreX; y = centreY; r = radius;  
    }  
    public Circle(double radius) { x=0; y=0; r = radius; }  
    public Circle() { x=0; y=0; r=1.0; }  
  
    //Methods to return circumference and area  
    public double circumference() { return 2*3.14*r; }  
    public double area() { return 3.14 * r * r; }  
}
```

Kullanım

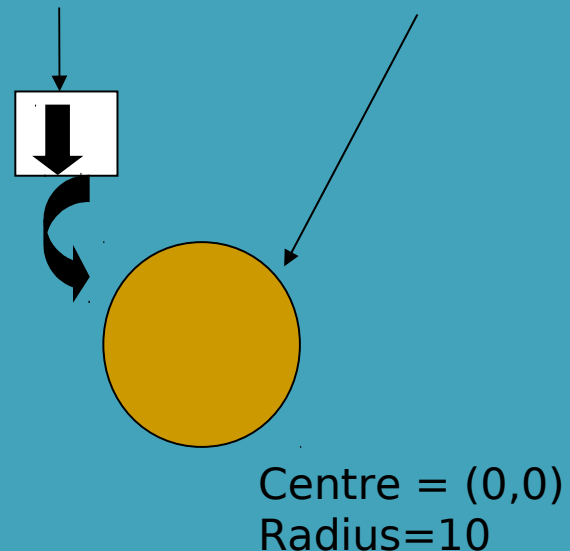
```
public class TestCircles {
```

```
    public static void main(String args[]){  
        Circle circleA = new Circle( 10.0, 12.0, 20.0);  
        Circle circleB = new Circle(10.0);  
        Circle circleC = new Circle();  
    }  
}
```

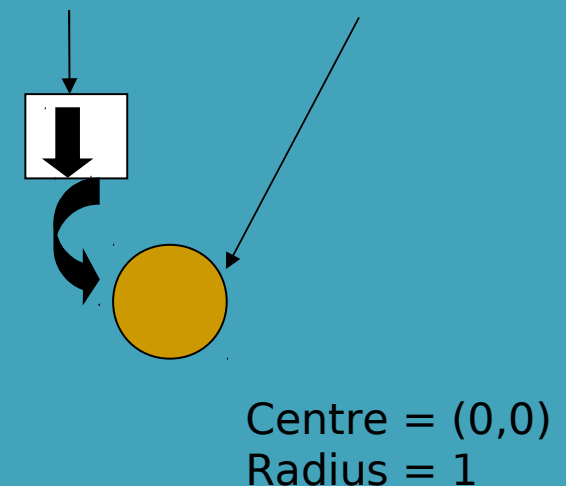
circleA = new Circle(10, 12, 20)



circleB = new Circle(10)



circleC = new Circle()



İlk Değerlerin Atanması

- Uygulamalarında üç tür değişken çeşidi bulunur:
 - yerel (*local*) değişkenler,
 - nesneye ait global alanlar ve
 - son olarak sınıfa ait global alanlar (*statik alanlar*).
- Bu değişkenlerin tipleri temel (*primitive*) veya herhangi bir sınıf tipi olabilir

Değişken Gösterimi

```
public class DegiskenGosterim {  
    int x ;           //nesneye ait global alan  
    static int y ;    // sınıfa ait global alan  
    public void metod () {  
        int i ; //yerel degisken  
    }  
}
```

İlk Değer Alma Sırası

- Nesnelere ait global alanlara başlangıç değerleri hemen verilir; üstelik, yapılandırıcılardan (*constructor*) bile önce...
- Belirtilen alanların konumu hangi sırada ise başlangıç değeri alma sırasında aynı olur.

Örnek

```
class Kagit {  
    public Kagit(int i) {  
        System.out.println("Kagit (" + i + ") ");  
    }  
}
```

Örnek (devam)

```
public class Defter {  
    Kagit k1 = new Kagit(1);           // dikkat  
    public Defter() {  
        System.out.println("Defter() yapilandirici ");  
        k2 = new Kagit(33);  
    } //artık başka bir Kagit nesnesine bağlı  
  
    Kagit k2 = new Kagit(2);           //dikkat  
  
    public void islemTamam() {  
        System.out.println("Islem tamam"); }  
  
    Kagit k3 = new Kagit(3);           //dikkat  
}
```

Çıktı

Kagit (1)

Kagit (2)

Kagit (3)

Defter() yapilandirici

Kagit (33)

Islem tamam

This

```
public class BankAccount {  
    public float balance;  
    public BankAccount( float initialBalance ) {  
        this.balance = initialBalance; } // this şart değil  
    public BankAccount( int balance ) {  
        this.balance = balance; } // this şart  
    public void deposit( float amount ) {  
        balance += amount ; }  
    public String toString() {  
        return "Account Balance: " + balance; }  
}
```

This-2

```
public class Point {  
    public int x = 0;  
    public int y = 0;  
    public Point(int x, int y)    // constructor  
    { this.x = x;  
      this.y = y; }  
    public Point(int x)    // constructor  
    { this.x = x;  
      this.y = 0; }  
}
```


Nesne Olarak “This”

```
public class BankAccount {  
    public float balance;  
    public BankAccount( float initialBalance ) {  
        this.balance = initialBalance;  
    }  
    public BankAccount deposit( float amount ) {  
        balance += amount ;  
        return this;  
    }  
}
```

This

```
public class RunBank {  
    public static void main( String args[] ) {  
        BankAccount richStudent = new BankAccount( 10000F );  
        richStudent.deposit( 100F ).deposit( 200F ).deposit( 300F );  
        System.out.println( "Student: " + richStudent.balance );  
    }  
} //?? Çıktı ne olur
```

Yapılandırıcılarda This

```
public class Tost {  
    int sayi ;  
    String malzeme ;  
  
    public Tost() {  
        this(5);  
        // this(5,"sucuklu");   Hata!-iki this kullanılamaz  
        System.out.println("parametresiz yapılandırıcı");  
    }  
  
    public Tost(int sayi) {  
        this(sayi,"Sucuklu");  
        this.sayi = sayi ;  
        System.out.println("Tost(int sayi) " );  
    }  
}
```

Örnek

```
public Tost(int sayi ,String malzeme) {  
    this.sayi = sayi ;  
    this.malzeme = malzeme ;  
    System.out.println("Tost(int sayi ,String malzeme) " );    }  
  
public void siparisGoster() {  
    // this(5,"Kasarli");      !Hata!-sadece yapılandırıcılarda  
    System.out.println("Tost sayisi="+sayi+ "=" + malzeme );  
}  
  
public static void main(String[] args) {  
    Tost t = new Tost();  
    t.siparisGoster();  
}          // main  
}          // class
```

Yapılandırıcılarda This

- Yapılandırıcılar içerisinde this ifadesi ile her zaman başka bir yapılandırıcı çağrılabilir.
- Yapılandırıcı içerisinde, diğer bir yapılandırıcıyı çağırırken this ifadesi her zaman ilk satırda yazılmalıdır.
- Yapılandırıcılar içerisinde birden fazla this ifadesi ile başka yapılandırıcı çağrılmaz.

Statik Yordamlar (Static Methods)

```
public class StatikTest {  
  
    public static void hesapla(int a , int b) {  
        islemYap(a,b);  
        // !Hata!  
    }  
  
    public void islemYap(int a , int b) {  
        // nesneye ait bir yordam, static bir yordamı çağırabilir  
        hesapla(a,b);  
    }  
}
```

Örnek

```
public class MutluAdam {  
    private String ruh_hali = "Mutluyum" ;  
  
    public void ruhHaliniYansit() {  
        System.out.println( "Ben " + ruh_hali );    }  
  
    public void tokatAt() {  
        if( ruh_hali.equals("Mutluyum" ) )  
            ruh_hali = "Sinirlendim";    }  
  
    public void kucakla() {  
        if( ruh_hali.equals( "Sinirlendim" ) )  
            ruh_hali = "Mutluyum";    }  
}
```

Örnek-2

```
public static void main(String[] args) {  
    MutluAdam obj1 = new MutluAdam();  
    MutluAdam obj2 = new MutluAdam();  
  
    obj1.ruhHaliniYansit();  
    obj2.ruhHaliniYansit();  
  
    obj1.kucakla();  
    obj2.tokatAt();  
  
    obj1.ruhHaliniYansit();  
    obj2.ruhHaliniYansit();  
}  
}
```


Static Metod

```
public class Toplama {  
  
    public static double topla(double a , double b ) {  
        double sonuc = a + b ;  
        return sonuc ;  
    }  
} // Sınıfın durum değişkenlerine bağımlı değil
```

Nasıl Kullanılır

```
public class ToplamIslemi {  
    public static void main(String args[]) {  
  
        String s1=Klavye.stringOku();  
        String s2=Klavye.stringOku();  
  
        double a = Double.parseDouble(s1);  
        double b = Double.parseDouble(s2);  
  
        double sonuc = Toplama.topla(a,b);    // dikkat  
        System.out.println("Sonuc : " + sonuc );  
    }  
}
```

Static Metodlar

- Aynı zamanda sınıf metodları olarakta bilinirler.
- Static metodlar doğrudan sınıfın ismiylede çağrılabilirler.
 - ▣ `double rand = Math.random();`
- Bir static metod, static olmayan bir değere veya metoda erişemez.
 - ▣ Static metodlar nesne örneklerinden bağımsız olarak çalışırlar.

Final Değişkenler

- Sabit değişkenler olup değerlerini değiştiremezsiniz

```
public class CityName {  
    final static public String name = "Kayseri";  
    public static void main(String args[]) {  
        System.out.println("Sehrin Adi " + name );  
    }  
}
```

- CityName.java, sınıfının herhangi bir yerinde *name* değişkeninin değerini değiştiremezsiniz. Çünkü *name* bir final değişkendir.

FINAL

final anahtar sözcüğünün kullanımı ile tanımlanır.

- ❑ `final int INCREMENT = 5;`
- ❑ `INCREMENT` değişkeninin değeri 5 tir
- ❑ final değişkenlere sadece bir kez değer atanabilir.

final int d;

d=5; //olur

d=9; //olmaz

Örnek (Final)

```
class FinalDegisken {  
    public static int x ;  
    public final int y =5;  
    public void ekranaBas(FinalDegisken sd ) {  
        System.out.println(" x = " + x + " y = " + y );  
    }  
public static void main(String args[]) {  
    FinalDegisken sd1 = new FinalDegisken ();  
    sd1.x = 50 ;  
    sd1.y = 2 ;  
    sd1.ekranaBas(sd1);  
    }  
}
```

// Derleme Hatası

Sorular ?



Teşekkürler

www.tart.com.tr / cihangir.savas@tart.com.tr

Web

cihangirsavas.com.tr

GitHub

github.com/siesta

Linkedin

tr.linkedin.com/in/cihangirsavas

tart
yeni medya nuntagi



Nesneye Yönelik Programlama

Cihangir SAVAS | SW101

Seminer İeriĐi

01. Class
02. Eriřim Denetleyiciler
03. Method Overloading
04. Yapılandırıcılar
05. İlk deĐer ataması
06. This
07. Static / Final
- 08.



Nesneler

- Dizi, aynı tipteki değerlerin bir koleksiyonudur.
- Nesne ise farklı tipteki değerlerin bir koleksiyonudur.
 - Aslında nesne bundan biraz daha karmaşıktır ama bu tanımı sadece karşılaştırma için yapıyoruz.
- Örnek:

```
class Person {  
    String name;  
    boolean male;  
    int age;  
    int phoneNumber;  
}
```
- Bu sebeple farklı tipteki nesneler bellekte farklı boyutta yer alacaktır.

Eriřim Belirleyiciler

public: Her yerden erişilmeyi sağlayan erişim belirleyicisi.

protected: Aynı paket içerisinde ve bu sınıftan türemiş alt sınıflar tarafından erişilmeyi sağlayan erişim belirleyicisi.

private: Yalnızca kendi sınıfı içerisinde erişilmeyi sağlayan, başka her yerden erişimi kesen erişim belirleyicisi.

private & public alanlar ve metodlar

- **private** alanlar türetilmiş sınıflarda kullanılamazlar.
 - "Bilgi Gizleme" böyle olması gerektiğini söyler
 - Eğer erişmek gerekiyorsa üst sınıfın ilgili metodları kullanılarak gerekli erişim yapılır.
- **private** metotlarda alt sınıflara kalıtsal olarak geçmezler.
 - Alt sınıflarda kullanılması gerekiyor ise public/protected olarak tanımlanmalıdır
 - Sadece yardımcı olmak için bazı metodlar private tanımlanabilir

protected Belirleyicisi

Görülebilir özellikler hangi alanların/metodların kalıtılabileceğini veya kalıtılamayacağını gösterirler `public` olarak tanımlanan değişkenler ve metodlar kalıtılabilirken, `private` olanlar görünmeyen özellikler olup kalıtılmamaktadır.

Fakat `public` alanlar/değişkenler kapsama ilkeline ters olan bir belirleyicidir.

Bu sebeple kalıtım durumunda yardımcı olan bir üçüncü belirleyici geliştirilmiştir : `protected`

protected Belirleyicisi - 2

protected belirleyicisi üst sınıfın bir üyesinin (alan veya metod) alt sınıfa geçmesini sağlar.

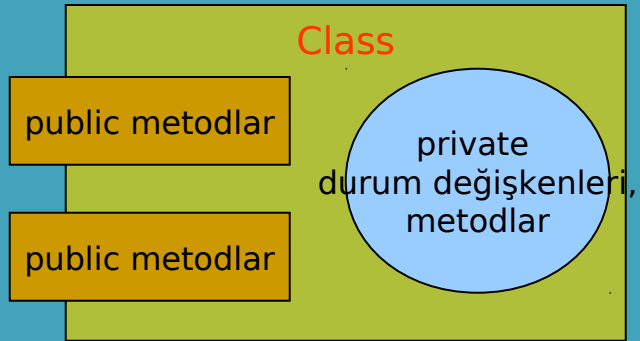
Protected şu iki özelliği sağlar

- public ten daha fazla bir kapsama özelliği vardır.
- Kalıtıma izin veren en iyi kapsama imkanını sağlar.

Tasarım Temelleri

- *private* durum değişkenleri kullanılarak özel veriler sınıf yapısı içinde sarmalanır-kapsülленir. (Encapsulation)
- Sınıf dışından bu verilere erişim public metodlar kullanılarak yapılır.
- *private* metodlar sadece sınıf içinden erişim için kullanılabilir.

Sarmalama-Kapsülleme Data Encapsulation



Sınıftaki private/public metodlar

```
private String isim, tel;  
private double haftalikmaas, yillikmaas;  
    // aynı sınıf içinden erişim için kullanılır  
private void HesaplaYillikMaas() {  
    yillikmaas = haftalikmaas * 12;  
}  
public void YazdirYillikMaas() {  
    HesaplaYillikMaas(); // aynı sınıf içinden erişim yapılıyor.  
    System.out.println(isim+" nin yillik maasi " + yillikmaas);  
}  
}
```

public Metodların kullanımı

```
class test {  
    public static void main(String args[]) {  
        Calisan tart=new Calisan();  
        tart.YazdirYillikMaas(); //şeklinde kullanılır  
        tart.HesaplaYillikMaas(); //kullanılmaz  
    }  
}
```

Bilgisayarda Bellek Organizasyonu

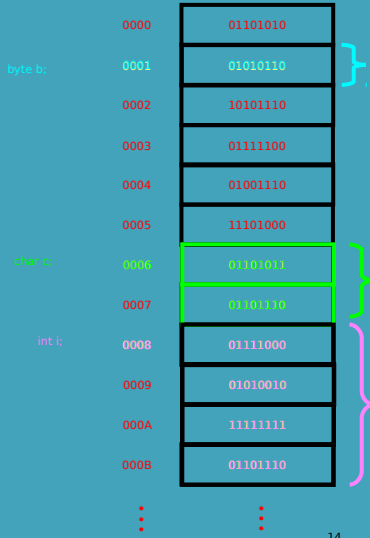
- Bir **bit** on/off, yes/no, **0/1** tipinden değerdir
- Bir **byte** ise sekiz ardışık bitten oluşur
- Bilgisayar Belleği ise 0 dan başlayan bir dizi byte dan oluşur
 - Bellekteki herhangi bir adresin boş olması diye bir kavram yoktur burada bir şekilde bitler(0/1) saklıdır.
- Bir byte ilişkilendirilmiş olan numara onun **adresidir**.
- Adresler ikili düzendeki sayılardır (genellikle hexadecimal olarak yazılırlar)
- Bazı büyük bilgisayar sistemleri ise byteları değil wordleri adresler. **Wordler** ise daha fazla sayıda bitten oluşur (such as 32 or 64)

• 000001101010
• 000101010110
• 000210101110
• 000301111100
• 000401001110
• 000511101000
• 000601101011
• 000701101110
• 000801111000
• 000901010010
• 000A11111111
• 000B01101110

• •
• •
• •

Bellekteki basit deęişkenler

- Basit bir deęişkeni tanımlayında bellekte ona bir yer ayırırsınız. (Tipine göre)
- Örneęin bir **byte** bellekte tek bir bytelik alana ihtiyaç duyar
- Bir **int** ise dört ardışık byte a ihtiyaç duyar
- Bir **char** iki ardışık byte a ihtiyaç duyar
- Bu yüzden her deęişkenin bellekte ayrı bir adresi vardır.



Bellekte Nesne Saklama

- Bir nesneyi tanımladığımız zaman ise Derleyici bu nesnenin ne kadar belleğe ihtiyacı olduğunu bilemez. Bu sebeple bellek ataması yapamaz.
- Gerekli bellek miktarı o nesneyi yarattığınızda bellekte tanımlanır ve atanır.

0000	01101010
0001	01010110
0002	10101110
0003	01111100
0004	01001110
0005	11101000
0006	01101011
0007	01101110
0008	01111000
0009	01010010
000A	11111111
000B	01101110
...	...

Nesnenin Tanımlanması ve Yaratılması

```
Person p = new Person("John");
```

Person p işaretçisi için bellekte bir yer tanımlar.

new Person("John"), Person sınıfının yapısına uygun bellekte bir yer ayırır.

Atama komutu ile p ye gerekli referans ataması gerçekleşir.

p:

07A3

07A3

John

Atama İşlemleri

- Basit Değişkenlerde veri atamalarda *değer* ataması yapılır
- Nesnelerin atanmasında ise *referansların kopyalaması* yapılır.
- Bu sebeple iki nesne aynı veri bloğuna işaret edebilir.
 - Bunu başka bir isimle adlandırma olarak düşünebilirsiniz.
 - Eğer bir nesnenin işaret ettiği bir değeri değiştirirseniz diğer nesnede değişecektir.

Parametrelerde Veri transferi

- Bir method çağırısı şu şekilde olur.

```
result = add( 3 , 5 ) ;
```

- Method ismi ve parametre isimleri şu şekilde olur.

```
int add ( int number1, int number2 ) {  
    int sum = number1 + number2 ;  
    return sum ;  
}
```

Parametre değerleri kopyalanarak gönderilir.

Sonuç geriye döner

Method için bunlar kullanılır

Basit Parametreler Kopyalanır.

```
int m = 3;  
int n = 5;  
result = add( m , n );
```

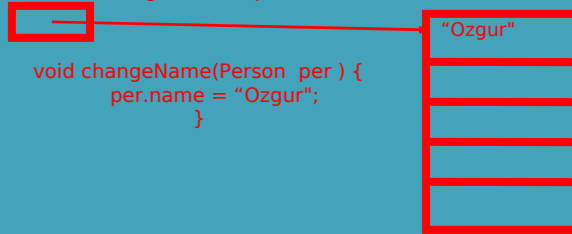
```
int add ( int number1 , int number2 ) {  
    while (number1 > 0) {  
        number1--;  
        number2++;  
    }  
    return number2 ;  
}
```

- Buna **değer ile çağırma (call by value)** denir

Nesnelerde ise *referenslar* gönderilir

p , Person in referansını tutar.

```
Person p = new Person("Ozgur");  
changeName( p );
```



- **p** ve **per** aynı nesneyi işaret etmektedir.!
- Bu sebeple **p** de yapılacak değişiklik diğerini de etkileyecektir.
- Buna referans ile çağırma (**call by reference**) denir

NullPointerException

- Bir nesne tanımladığınız zaman onun ilk değeri null (0000) dır.
- null her nesne tanımlaması için kullanılabilen bir değerdir.
 - Bir nesnenin değerinin null olup olmadığını kontrol edebilirsiniz (if (x==null) a****)
 - Bir nesneye null değer atayabilirsiniz. (x=null;)
 - null değere sahip bir nesneyi başka şekilde kullanamazsınız.
- Örnek:

```
Person ozgur;
```

```
ozgur.name = "OzgurKoray"; // NullPointerException
```

Eşitlik

- İki basit değişkenin eşitliğini değerleri üzerinden yapabilirsiniz.

x 24

y 24

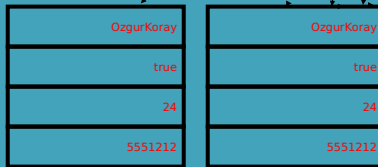
- İki nesnenin eşitliğini ise referansları üzerinden yapmanız gerekir.

- `a == b`, fakat `a != c`
- *Stringler için ise `equals` metodunu kullanmanız gerekecektir. `s1.equals(s2)`*

a

b

c



Nesne Kopyalama

```
Box b1 = new Box(3, 4);
```



```
Box b2 = b1.clone();
```

```
Box b3 = b2;
```

```
b2.setWid(8);
```

```
System.out.println(b3.getWid());
```



```
b2 = null;
```

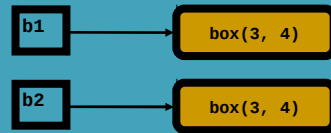


Soru ?

```
Box b1 = new Box(3, 4);  
Box b2 = new Box(3, 4);  
if(b1 == b2)  
    System.out.println("Bu Kutu Nesneleri Eşittir!");  
else  
    System.out.println("Nasıl Eşit Olsunlar ki!.... ");
```


Eşitlik

- Aynı Bellek alanına işaret etmedikleri için eşit değildir.



Eşitlik-String

```
class test {  
    public static void main (String args[]) {  
        String str1= new String("Ozgur");  
        String str2= new String("Ozgur");  
  
        System.out.println ("Ozgur" == "Ozgur");  
        System.out.println (str1 == "Ozgur");  
        System.out.println (str1 == str2);  
        System.out.println (str1.equals(str2));  
        System.out.println (str1.equalsIgnoreCase(str2));  
    }  
}  
// Çıktı ? ? ?
```

Cıktı


- tue
- false
- false
- true
- true

Adaş Yordamlar-(Metod Overloading)

- Aynı metod ismi vardır. Fakat farklı parametreler gönderilir.
- Örneğin:
 - bilgiDegistir(String t);
 - bilgiDegistir(double s);
 - bilgiDegistir(String t, double s);

Örnek

- `System.out.println(12);`
- `System.out.println("Özgür Koray");`
- `System.out.println('E');`
- `System.out.println(12.0f);`

 `System.out.printString("Ozgur Koray");`
`System.out.printInt(12);`

Adaş Yordamlar

```
public class OverLoad {  
    public void same()  
        { System.out.println( "No arguments" ); }  
    public void same( int firstArgument )  
        { System.out.println( "One int arguments" ); }  
    public void same( char firstArgument )  
        { System.out.println( "One char arguments" ); }  
    public int same( int firstArgument ) // Derleme Hatasi  
        {System.out.println( "One char arguments" ); return 5; }  
    public void same( char firstArgument, int secondArgument)  
        { System.out.println( "char + int arguments" ); }  
    public void same( int firstArgument, char secondArgument )  
        { System.out.println( "int + char arguments" ); } }  
}
```

CalisanOverloading

```
public class CalisanOverloading {
    private String isim;
    private String tel;
    private double haftalikmaas;
    public void bilgiYazdir() {
        System.out.println( isim + " nin telefonu " + tel + " , Haftalik Maasi " + haftalikmaas);
    }
    public void bilgiDegistir(String t) {
        tel = t;
    }
    public void bilgiDegistir(double s) {
        haftalikmaas = s;
    }
    public void bilgiDegistir(String t, double s) {
        tel = t;    haftalikmaas = s;
    }
}

CalisanOverloading ozgur=new CalisanOverloading("Ozgur Koray SAHINGOZ", "6632490", 1234.50);
ozgur.bilgiDegistir("6632491");
ozgur.bilgiDegistir(1500.00);
ozgur.bilgiDegistir("555-3824", 1750.80);
```

Örnek?

Gönderilen integer değerlerin en büyüğünü döndüren Max metodu nasıl geliştirilir.

```
int x= max(12,24);
```

```
int x= max(12,24,35);
```

```
int[] dizi = {12,24,34,45,56,67,78}
```

```
int x=max(dizi);
```


Adaş Yapılandırıcı

- Adaş metodların özel bir durumudur.
- Farklı parametreler ile birden fazla tanımlanabilir.

```
public class CalisanOverCons {  
    CalisanOverCons (String n);  
    CalisanOverCons (String n, String t);  
    CalisanOverCons (String n, String t, double s);  
}
```

Adaş Yapılandırıcı

```
public CalisanOverCons(String n) {  
    isim = n; tel = "000-0000"; haftalikmaas = 0.0;  
}  
public CalisanOverCons(String n, String t) {  
    isim = n; tel = t; haftalikmaas = 0.0;  
}  
public CalisanOverCons(String n, String t, double s) {  
    isim = n; tel = t; haftalikmaas = s;  
}
```

Kullanımı

```
CalisanOverCons ozgur = new CalisanOverCons("Ozgur Koray SAHINGOZ");  
ozgur.chanageinfo("555-1234", 1500.00);
```

```
CalisanOverCons jordan = new CalisanOverCons("Michael Jordan", "555-4548");  
jordan.changeinfo(2500.00);
```

```
CalisanOverCons gates = new CalisanOverCons("Bill Gates", "555-4647", 3000.00);
```

Örnek-2

```
public class Point
{
    public int x = 0;
    public int y = 0;
    public Point(int a, int b)    // yapılandırıcı
    {
        x = a;
        y = b;
    }
    public Point(int a)    // yapılandırıcı
    {
        x = a;
        y = 0;
    }
}
```

?? Method overloading

Circle Sınıfı ile Kullanım Örneği

```
public class Circle {  
    public double x,y,r;  
    // Constructor  
    public Circle(double centreX, double centreY, double radius)  
    {  
        x = centreX;  
        y = centreY;  
        r = radius;  
    }  
    //Methods to return circumference and area  
    public double circumference() { return 2*3.14*r; }  
    public double area() { return 3.14 * r * r; }  
}
```

Constructor kullanmaz ise

```
Circle aCircle = new Circle();  
aCircle.x = 10.0; // initialize center and radius  
aCircle.y = 20.0  
aCircle.r = 5.0;
```



İlk yaratılma zamanında
çemberin merkezi ve çapı
tanımlı değildir

Bu değerler sonradan atanır.

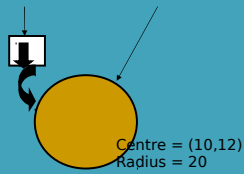
Çoklu Yapılandırıcılar

```
public class Circle {  
    public double x,y,r; //instance variables  
    // Constructors  
    public Circle(double centreX, double centreY, double radius) {  
        x = centreX; y = centreY; r = radius;  
    }  
    public Circle(double radius) { x=0; y=0; r = radius; }  
    public Circle() { x=0; y=0; r=1.0; }  
  
    //Methods to return circumference and area  
    public double circumference() { return 2*3.14*r; }  
    public double area() { return 3.14 * r * r; }  
}
```

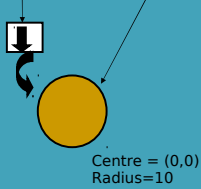
Kullanım

```
public class TestCircles {  
    public static void main(String args[]){  
        Circle circleA = new Circle( 10.0, 12.0, 20.0);  
        Circle circleB = new Circle(10.0);  
        Circle circleC = new Circle();  
    }  
}
```

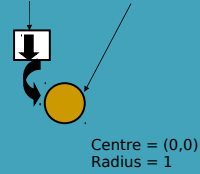
circleA = new Circle(10, 12, 20)



circleB = new Circle(10)



circleC = new Circle()



İlk Değerlerin Atanması

- Uygulamalarında üç tür değişken çeşidi bulunur:
 - yerel (*local*) değişkenler,
 - nesneye ait global alanlar ve
 - son olarak sınıfa ait global alanlar (*statik alanlar*).
- Bu değişkenlerin tipleri temel (*primitive*) veya herhangi bir sınıf tipi olabilir

Değişken Gösterimi

```
public class DegiskenGosterim {  
    int x ;           //nesneye ait global alan  
    static int y ;    // sınıfa ait global alan  
    public void metod () {  
        int i ; //yerel degisken  
    }  
}
```

İlk Değer Alma Sırası

- Nesnelere ait global alanlara başlangıç değerleri hemen verilir; üstelik, yapılandırıcılardan (*constructor*) bile önce...
- Belirtilen alanların konumu hangi sırada ise başlangıç değeri alma sırasında aynı olur.

Örnek

```
class Kagit {  
    public Kagit(int i) {  
        System.out.println("Kagit (" + i + ") ");  
    }  
}
```

Örnek (devam)

```
public class Defter {  
    Kagit k1 = new Kagit(1);        // dikkat  
    public Defter() {  
        System.out.println("Defter() yapilandirici ");  
        k2 = new Kagit(33);  
    } //artık başka bir Kagit nesnesine bağlı  
  
    Kagit k2 = new Kagit(2);        //dikkat  
  
    public void islemTamam() {  
        System.out.println("Islem tamam"); }  
  
    Kagit k3 = new Kagit(3);        //dikkat  
}
```

Çıktı

Kagit (1)

Kagit (2)

Kagit (3)

Defter() yapilandirici

Kagit (33)

Islem tamam

This

```
public class BankAccount {  
    public float balance;  
    public BankAccount( float initialBalance ) {  
        this.balance = initialBalance; } // this şart değil  
    public BankAccount( int balance ) {  
        this.balance = balance; } // this şart  
    public void deposit( float amount ) {  
        balance += amount ; }  
    public String toString() {  
        return "Account Balance: " + balance; }  
}
```

o an kullanılan nesneye belirtir. O an kullanılan nesneden bir parametre gönderilmesi gerektiğinde kullanılır.

This-2

```
public class Point {  
    public int x = 0;  
    public int y = 0;  
    public Point(int x, int y)    // constructor  
    { this.x = x;  
      this.y = y; }  
    public Point(int x)    // constructor  
    { this.x = x;  
      this.y = 0; }  
}
```


Nesne Olarak “This”

```
public class BankAccount {  
    public float balance;  
    public BankAccount( float initialBalance ) {  
        this.balance = initialBalance;  
    }  
    public BankAccount deposit( float amount ) {  
        balance += amount ;  
        return this;  
    }  
}
```

This

```
public class RunBank {  
    public static void main( String args[] ) {  
        BankAccount richStudent = new BankAccount( 10000F );  
        richStudent.deposit( 100F ).deposit( 200F ).deposit( 300F );  
        System.out.println( "Student: " + richStudent.balance );  
    }  
} /// Çıktı ne olur
```

Yapılandırıcılarda This

```
public class Tost {  
    int sayi ;  
    String malzeme ;  
  
    public Tost() {  
        this(5);  
        // this(5,"sucuklu");   Hata!-iki this kullanılamaz  
        System.out.println("parametresiz yapılandırıcı");  
    }  
  
    public Tost(int sayi) {  
        this(sayi,"Sucuklu");  
        this.sayi = sayi ;  
        System.out.println("Tost(int sayi) " );  
    }  
}
```

Örnek

```
public Tost(int sayi ,String malzeme) {  
    this.sayi = sayi ;  
    this.malzeme = malzeme ;  
    System.out.println("Tost(int sayi ,String malzeme) " );  }  
  
public void siparisGoster() {  
    // this(5,"Kasarli");    !Hata!-sadece yapılandırıcılarda  
    System.out.println("Tost sayisi="+sayi+ "="+ malzeme );  
}  
  
public static void main(String[] args) {  
    Tost t = new Tost();  
    t.siparisGoster();  
}      // main  
}      // class
```

Yapılandırıcılarda This

- Yapılandırıcılar içerisinde this ifadesi ile her zaman başka bir yapılandırıcı çağrılabilir.
- Yapılandırıcı içerisinde, diğer bir yapılandırıcıyı çağırırken this ifadesi her zaman ilk satırda yazılmalıdır.
- Yapılandırıcılar içerisinde birden fazla this ifadesi ile başka yapılandırıcı çağrılmaz.

Statik Yordamlar (Static Methods)

```
public class StatikTest {  
  
    public static void hesapla(int a , int b) {  
        islemYap(a,b);  
        // !Hata!  
    }  
  
    public void islemYap(int a , int b) {  
        // nesneye ait bir yordam, static bir yordamı çağırabilir  
        hesapla(a,b);  
    }  
}
```

Örnek

```
public class MutluAdam {  
    private String ruh_hali = "Mutluyum" ;  
  
    public void ruhHaliniYansit() {  
        System.out.println( "Ben " + ruh_hali );    }  
  
    public void tokatAt() {  
        if( ruh_hali.equals("Mutluyum" ) )  
            ruh_hali = "Sinirlendim";    }  
  
    public void kucakla() {  
        if( ruh_hali.equals( "Sinirlendim" ) )  
            ruh_hali = "Mutluyum";    }  
}
```

Örnek-2

```
public static void main(String[] args) {  
    MutluAdam obj1 = new MutluAdam();  
    MutluAdam obj2 = new MutluAdam();  
  
    obj1.ruhHaliniYansit();  
    obj2.ruhHaliniYansit();  
  
    obj1.kucakla();  
    obj2.tokatAt();  
  
    obj1.ruhHaliniYansit();  
    obj2.ruhHaliniYansit();  
}  
}
```


Static Metod

```
public class Toplama {  
  
    public static double toplama(double a , double b ) {  
        double sonuc = a + b ;  
        return sonuc ;  
    }  
} // Sınıfın durum değişkenlerine bağımlı değil
```

Nasıl Kullanılır

```
public class ToplamIslemi {  
    public static void main(String args[]) {  
  
        String s1=Klavye.stringOku();  
        String s2=Klavye.stringOku();  
  
        double a = Double.parseDouble(s1);  
        double b = Double.parseDouble(s2);  
  
        double sonuc = Toplama.topla(a,b);    // dikkat  
        System.out.println("Sonuc : " + sonuc );  
    }  
}
```

Static Metodlar

- Aynı zamanda sınıf metodları olarakta bilinirler.
- Static metodlar doğrudan sınıfın ismiylede çağrılabilirler.
 - `double rand = Math.random();`
- Bir static metod, static olmayan bir değere veya metoda erişemez.
 - Static metodlar nesne örneklerinden bağımsız olarak çalışırlar.

Final Değişkenler

- Sabit değişkenler olup değerlerini değiştiremezsiniz

```
public class CityName {  
    final static public String name = "Kayseri";  
    public static void main(String args[]) {  
        System.out.println("Sehrin Adi " + name );  
    }  
}
```

- CityName.java, sınıfının herhangi bir yerinde *name* değişkeninin değerini değiştiremezsiniz. Çünkü *name* bir final değişkendir.

FINAL

final anahtar sözcüğünün kullanımı ile tanımlanır.

- final int INCREMENT = 5;
- INCREMENT değişkeninin değeri 5 tir
- final değişkenlere sadece bir kez değer atanabilir.

final int d;

d=5; //olur

d=9; //olmaz

Örnek (Final)

```
class FinalDegisken {  
    public static int x ;  
    public final int y =5;  
    public void ekranaBas(FinalDegisken sd ) {  
        System.out.println(" x = " + x + " y = " + y );  
    }  
    public static void main(String args[]) {  
        FinalDegisken sd1 = new FinalDegisken ();  
        sd1.x = 50 ;  
        sd1.y = 2 ;  
        sd1.ekranaBas(sd1);  
    }  
}  
// Derleme Hatası
```

Sorular ?

1. Sorular

61



Teşekkürler

www.tart.com.tr / cihangir.savas@tart.com.tr

Web

cihangirsavas.com.tr

GitHub

github.com/cihangir

LinkedIn

linkedin.com/in/cihangirsavas