

Nesneler ve Sınıflar

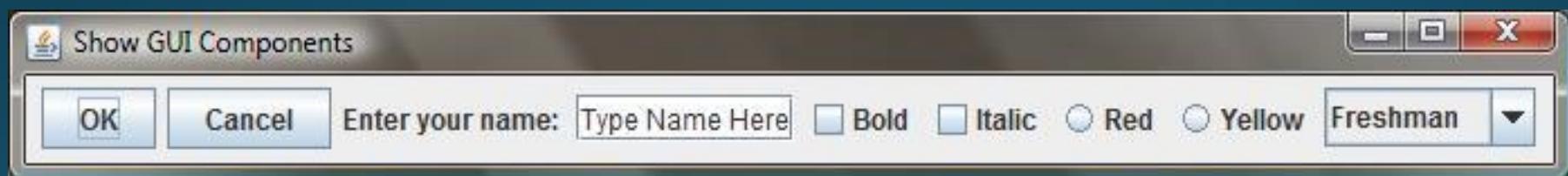
Giriş

Şu ana kadar döngüler, diziler, seçim komutları ve metotlar anlatıldı.

Bu komutlar ile birçok program yapılabilir.

Fakat şu ana kadar görülenler ile grafiksel kullanıcı arabirimini gibi büyük ölçekli yazılım sistemleri geliştirilemez.

Aşağıdaki gibi bir kullanıcı ara yüzü nasıl geliştirilebilir?



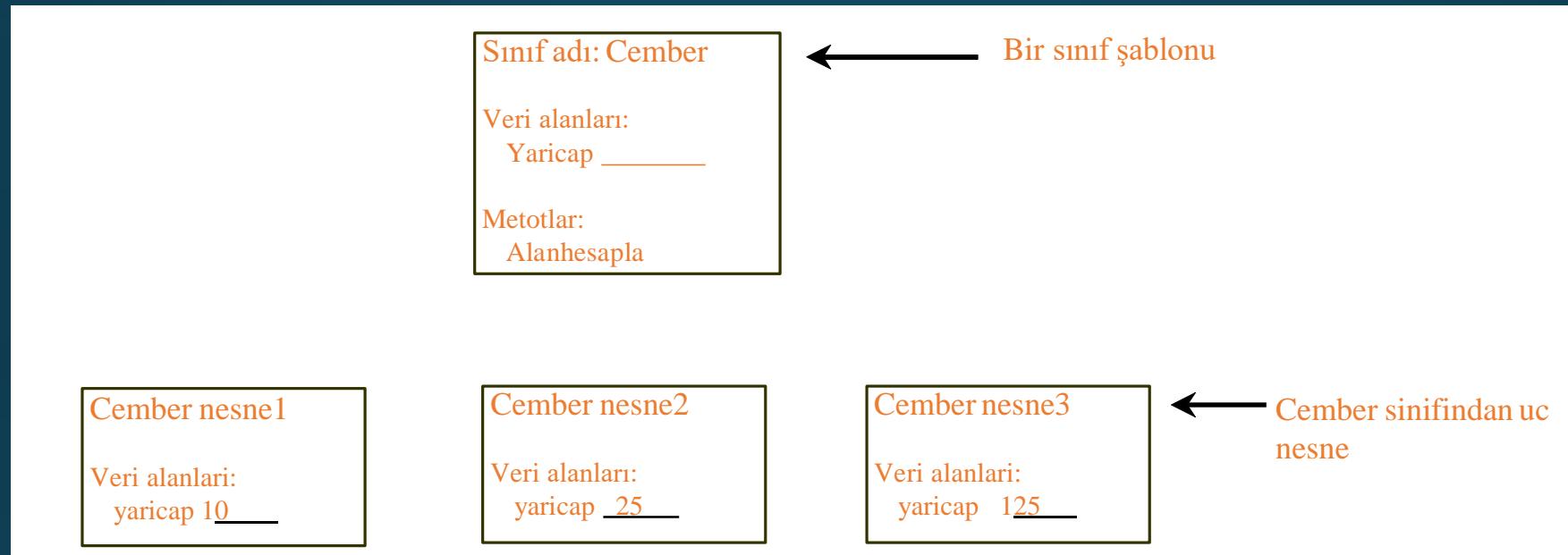
Amaçlar

- Nesne ve sınıfları tanımlamak ve kullanmak
- Nesne ve sınıfları tanımlamak için UML diyagramlarını kullanmak
- Yapılandırmacılar
- Nesne referans değişkenleri ile nesnelere erişmek
- Bir referans tip kullanarak referans değişken tanımlama
- Erişim operatörleri ile nesnelerin veri ve metodlarına erişim
- Referans türünden veri alanlarının tanımlanması
- Nesne değişkenleri ile ilkel değişkenlerin karşılaştırılması
- Java kütüphanesinde bulunan Date, Random, ve JFrame sınıflarının kullanımı.
- Statik değişken ve metodlar
- Get ve set metodları

Nesne Tabanlı Programlama

- Nesne tabanlı programlama nesneleri kullanan programlamayı içerir.
- Bir nesne farklı olarak tanımlanabilen gerçek dünyadaki bir varlıktır.
- Örneğin; bir öğrenci, sıra, çember, bir buton birer nesnedir.
- Bir nesne tek kimlik, durum ve davranışlara sahiptir.
- Bir nesnenin durumu özellikleri olarak bilinen veri alanlarıdır.
- Bir nesnenin davranışı metotlar kümesi ile tanımlanır.

Nesneler



Bir nesne hem durum hem davranışa sahiptir.

Durum nesneyi, davranış ise nesnenin ne yaptığını tanımlar.

Sınıflar

Sınıflar aynı tipten nesneleri tanımlayan yapılardır.

*Bir java sınıfı veri alanları tanımlamak için **değişkenleri** ve davranışları tanımlamak için **metotları** kullanır.*

Ek olarak, bir sınıf yapılandırıcı olarak bilinen özel metotları tanımlar.

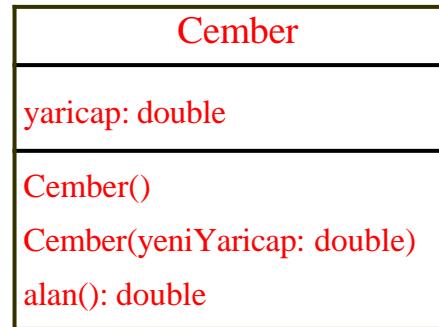
Yapılandırıcılar sınıfın nesnelerini oluşturmak için çağrıılır.

Sınıflar

```
class Cember {  
    /** bu cemberin yarıçapı */  
    double yaricap = 1.0; ← Veri alanı  
    /** bir cember sınıfının yapılandırıcısı */  
    Cember() {  
    }  
  
    /** Cember nesnesinin yapılandırıcısı */  
    Cember(double yeniYaricap) {  
        yaricap = yeniYaricap;  
    } ← Yapıcı  
  
    /** Cemberin alanına dondur */  
    double alan() {  
        return yaricap*yaricap * 3.14159; ← Metot  
    }  
}
```

UML Sınıf Diyagramı

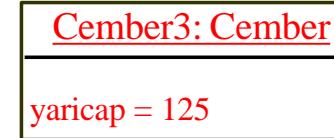
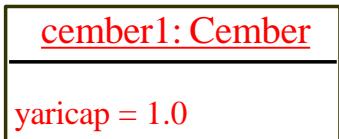
UML Sınıf diyagramı



Sınıf adı

Veri alnaları

Yapilandirici ve
metotlar



Nesnelerin UML
notasyonu

Örnek: Sınıfların tanımlanması ve nesne oluşturma

```
// İki yapılandırıcı ile sınıf tanımı  
class Cember1 {  
    double yaricap;  
    /yaricapi 1 olan çember nesnesi  
    Cember1() {  
        yaricap = 1.0;  
    }  
    // Belirlenen yarıçap ile çember tanımlama  
    Cember1(double yeniyaricap) {  
        yaricap = yeniyaricap;  
    }  
    /** Return the area of this Cember */  
    double getArea()  
    {  
        return yaricap * yaricap * Math.PI; }  
    }
```

TestCember1

```
public class TestCircle1 {  
    /** Main method */  
    public static void main(String[] args) {  
        // Create a circle with radius 5.0  
        Cember1 c1 = new Cember1(5.0);  
        System.out.println("yaricapi " + c1.yaricap +  
                           " olan çemberin alanı " + c1.alan());  
        // Yaricapi 1 olan çember nesnesi olusturma  
        Cember1 c2 = new Cember1();  
        System.out.println("yaricapi " + c2.yaricap +  
                           " olan çemberin alanı: " + c2.alan());  
        // yaricapi degistir  
        c2.yaricap = 100;  
        System.out.println("yaricapi " + c2.yaricap +  
                           " olan çemberin alanı: " + c2.alan());  
    }  
}
```

Örnek:

```
Public class TestTV {  
    public static void main(String[] args) {  
        TV tv1=new TV();  
        Tv1.ac();  
        tv1.kanalayarla(30);  
        tv1.sesAyarla(3);  
        TV tv2=new TV();  
        Tv2.ac();  
        tv2.kanalArtim();  
        tv2.kanalArtim();  
        System.out.Println(«Tv 1'in kanali:»+tv1.kanal + «ve ses  
        seviyesi:»+tv1.sesSeviyesi);  
        System.out.Println(«Tv 2'in kanali:»+tv2.kanal + «ve ses  
        seviyesi:»+tv2.sesSeviyesi);  
    }  
}
```

```
public class TV {
    int kanal = 1; // Varsayılan kanal 1
    int sesSeviyesi = 1; // Varsayılan sesSeviyesi 1
    boolean ac = false; // TV varsayılan olarak kapalı
    public TV() {
    }
    public void ac() {
        ac = true;
    }
    public void kapat() {
        ac = false;
    }
    public void Kanalayarla(int yeniKanal) {
        if (ac && yeniKanal >= 1 && yeniKanal <= 120)
            kanal = yeniKanal;
    }
    public void sesAyarla(int yeniSesSeviyesi) {
        if (ac && yeniSesSeviyesi >= 1 && yeniSesSeviyesi <= 7)
            sesSeviyesi = yeniSesSeviyesi;
    }
    public void kanalArtim() {
        if (ac && kanal < 120)
            kanal++;
    }
    public void kanalAzaltim() {
        if (ac && kanal > 1)
            kanal--;
    }
    public void sesArtim() {
        if (ac && sesSeviyesi < 7)
            sesSeviyesi++;
    }
    public void sesAzaltim() {
        if (ac && sesSeviyesi > 1)
            sesSeviyesi--;
    }
}
```

Örnek:
sınıfların
tanımlanması
ve nesnelerin
oluşturulması

Yapılandırıcılar

```
Cember() {  
}
```

```
Cember(double yeniYaricap) {  
    yaricap = yeniYaricap;  
}
```

Yapılandırıcılar nesneleri oluşturmak için çağrılan özel bir metod türüdür.

Yapılandırıcılar

Parametresi olmayan yapılandırıcı argümana sahip olmayan yapılandırıcı olarak başvurulur.

- Yapılandırıcılar kendi sınıfın ismi ile aynı isme sahiptir.
- Yapılandırıcılar herhangi bir geri dönüş türüne sahip değildir—void bile.
- Yapılandırıcılar nesne oluşturulduğunda new operatörü ile çağrılır
- Yapılandırıcılar nesnelerin başlatılmasında rol oynar

Yapılandırıcı kullanarak nesne oluşturma

```
new sınıfAdı();
```

Örnek:

```
new Çember();
```

```
new Çember(5.0);
```

Varsayılan Yapılandırıcı

Yapılandırıcı tanımlamadan bir sınıf tanımlanabilir.
Bu durumda argümanı olmayan bir yapılandırıcı
dolaylı olarak tanımlanır.

Bu yapılandırıcı varsayılan yapılandırıcıdır.

Nesne referans değişkenlerini tanımlama

Bir nesneyi referans göstermek için bir referans değişkene nesne atanır.

```
sinifAdi NesneRefDeg;
```

Örnek:

```
Cember Cemberim;
```

Tek adımda Nesnelerin oluşturulması ve tanımlanması

```
sinifAdi nesneRefDeg = new sinifAdi();
```

Örnek:

Nesne referansına atama Nesne oluşturma

```
Cember myCember =  new Cember();
```

Accessing Objects

- Referencing the object's data:

`objectRefVar.data`

e.g., myCember.yaricap

- Invoking the object's method:

`objectRefVar.methodName(arguments)`

e.g., myCember.getArea()

Kod İzleme

```
Cember Cemberim = new Cember(5.0);
```

```
Cember Cember2 = new Cember();
```

```
Cember2.yaricap = 100;
```

Cemberim i tanımla

Cemberim

Değer yok

Kod izleme

```
Cember Cemberim = new Cember(5.0);
```

Cemberim

Değer yok

```
Cember Cember2 = new Cember();
```

```
Cember2.yaricap = 100;
```

:Cember

yaricap 5.0

Cember oluşturma

Kod izleme

```
Cember Cemberim = new Cember(5.0);
```

```
Cember Cember2 = new Cember();
```

```
Cember2.yaricap = 100;
```

Cemberim'e referans
nesnesini ata

Cemberim

Referans değer

:Cember

yaricap: 5.0

Kod İzleme

```
Cember Cemberim = new Cember(5.0);
```

```
Cember Cember2 = new Cember();
```

```
Cember2.yaricap = 100;
```

Cemberim

Referans değer

:Cember

yaricap: 5.0

Cember2

Değer yok

Cember2 yi tanımla

Kod İzleme

```
Cember Cemberim = new Cember(5.0);
```

```
Cember Cember2= new Cember();
```

```
Cember2.yaricap = 100;
```

Cemberim

Referans değer

: Cember

yaricap: 5.0

Cember2

Değer yok

Yeni Cember
nesnesi olustur

: Cember

yaricap: 0.0

Kod izleme, cont.

```
Cember Cemberim = new Cember(5.0);
```

```
Cember Cember2 = new Cember();
```

```
Cember2.yaricap = 100;
```

Cemberim Referans değer

:Cember

yaricap: 5.0

Cember2 Referans değer

:Cember

yaricap: 1.0

Cember2'ye nesne
referansını ata

Kod İzleme

```
Cember Cemberim = new Cember(5.0);
```

```
Cember Cember2 = new Cember();
```

```
Cember2.yaricap = 100;
```

Cemberim Referans değer

:Cember

yaricap: 5.0

Cember2 Referans değer

:Cember

yaricap: 100.0

Cember2'deki
yarıçapı değiştir

Dikkat

Math sınıfında bir metodу çağırırmak için

Math.methodAdı(argümanlar) (Örneğin, Math.pow(3, 2.5))

Formatı kullanılır.

Cember.alan()'ı kullanarak alan() metodunu çağrıabilir miyiz?

Cevap HAYIR

Bu bölüme kadar anlatılan bütün metodlar static tanımlayıcısı ile tanımlanan statik metodlardır.

Bununla birlikte alan() statik olmayan bir metottur.

Dolayısıyla alan() metodу bir nesne değişkeni ile aşağıdaki gibi çağrırlı.

nesneRefDeg.metotAdı(argümanlar) (Örneğin, Cemberim.alan()).

Referans veri alanları

Veri alanları referans tipleri olabilir. Örneğin aşağıdaki `Ogrenci` sınıfı `String` türünden bir veri alanına sahiptir.

```
public class Ogrenci {  
    String isim; // Varsayılan değeri null  
    int yas; // Varsayılan değer 0  
    boolean mezunMu; // Varsayılan değer false  
    char cinsiyet; // Varsayılan değer '\u0000'  
}
```

Null Değeri

Eğer referans tipin veri alanı herhangi bir nesneye referans göstermiyorsa veri alanı null değer alır.

Bir veri Alanı için Varsayılan Değer

Bir referans için veri alanının varsayılan değeri null, nümerik tür için 0, boolen türü için false, ve char için '\u0000' şeklindedir.

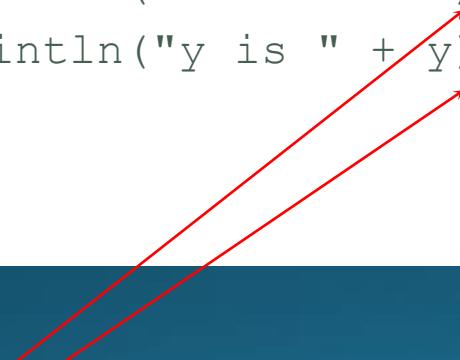
Bununla birlikte Java bir metottaki lokal değişkenlere değer atamaz.

```
public class Test {  
    public static void main(String[] args) {  
        Ogrenci o= new Ogrenci();  
        System.out.println("isim? " + o.isim);  
        System.out.println("yaş? " + o.yas);  
        System.out.println("Mezun mu? " + o.mezunMu);  
        System.out.println("cinsiyet? " + o.cinsiyet);  
    }  
}
```

Örnek:

Java bir metottaki lokal değişkenlere değer atamaz.

```
public class Test {  
    public static void main(String[] args) {  
        int x; // x has no default value  
        String y; // y has no default value  
        System.out.println("x is " + x);  
        System.out.println("y is " + y);  
    }  
}
```



Derleme hatası: Değişkenler başlatılmaz

Nesne türleri ile ilkel veri türleri arasındaki farklar

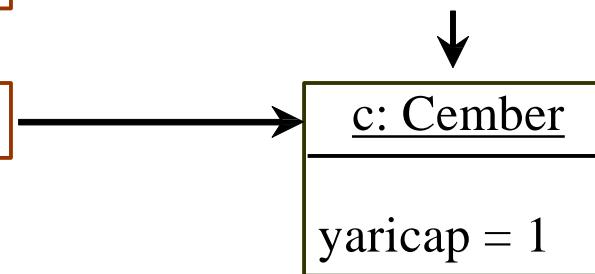
İlkel tip int i = 1 i



new Circle() kullanılarak
oluşturulur

Nesne türü Circle c

c



İllkel veri tipinin ve nesne türlerinin değişkenlerini kopyalama

İllkel tip atama $i = j$

Önce:

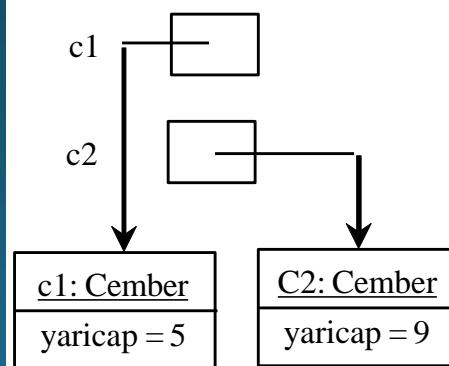


Sonra:

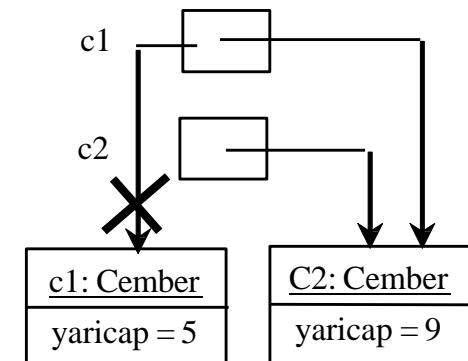


Nesne türü atama $c1 = c2$

Önce:



Sonra:



Çöp Toplayıcısı

Önceki şekilde gösterildiği $c_1=c_2$ atamasından sonra c_1 ve c_2 aynı yeri gösterir.

Daha önce c_1 ile referans gösterilen nesne artık referans gösterilmemektedir.

Bu nesne çöp olarak bilinir ve JVM tarafından otomatik olarak toplanır.

Eğer bir nesneye artık ihtiyaç duyulmayacağıni biliyorsanız nesne için null referans değeri atamanız gereklidir.

Eğer bir nesne herhangi bir değişken ile referans gösterilmiyorsa JVM otomatik olarak boş olanı toplar.

Date Sınıfı

Java Java. Util.Date sınıfında zaman ve tarih bilgisini verir. Zaman ve tarih bilgisini elde etmek için Date sınıfından bir nesne oluşturmalı ve toString ile stringe dönüştürüp String olarak kullanabilirsiniz.

+ public
tanımlayıcıyı
gösterir

The diagram shows a UML class named 'java.util.Date'. An arrow points from the text '+ public tanımlayıcıyı gösterir' to the class box. The class contains five methods:

- +Date()
- +Date(elapseTime: long)
- +toString(): String
- +getTime(): long
- +setTime(elapseTime: long): void

	java.util.Date
→	+Date() +Date(elapseTime: long) +toString(): String +getTime(): long +setTime(elapseTime: long): void

- Geçerli zaman için Date nesnesi oluşturur.
Verilen zaman için January 1, 1970'ten geçen zamanı milisaniye olarak verir.
Zaman ve tarihi string olarak geri döndürür.
January 1, 1970, GMT tarihinden geçen zamanı milisaniye olarak verir.
Nesnede yeni geçen zamanı verir.

Date sınıfı örnek:

Örneğin aşağıdaki kod

```
Date tarih = new Date();  
System.out.println(tarih.toString());
```

Sun Mar 09 13:50:19 EST 2013 gibi bir mesaj verir.

Random Sınıfı

0.0 ile 1.0 aralığında rastgele değer üretmek için
Math.random() kullanılır

Daha kullanışlı bir rastgele değişken üretme metodu
java.util.Random sınıfıdır.

java.util.Random	
+Random()	Kendi başlangıcı ile rastgele sınıfı oluşturur.
+Random(seed: long)	Belirlenen başlangıç ile rastgele nesne oluşturur.
+nextInt(): int	Rastgele bir tam sayı üretir
+nextInt(n: int): int	0 ile n arası rastgele bir tamsayı üretir.
+nextLong(): long	Long türünden rastgele sayı üretir.
+nextDouble(): double	0.0 ile 1.0 arasında double türünden rastgele bir değer döndürür.
+nextFloat(): float	0.0F ile 1.0F aralığında float türünden bir değer üretir.
+nextBoolean(): boolean	Rastgele Boolean bir değer üretir.

The Random Class Example

If two Random objects have the same seed, they will generate identical sequences of numbers. For example, the following code creates two Random objects with the same seed 3.

```
Random random1 = new Random(3);
System.out.print("From random1: ");
for (int i = 0; i < 10; i++)
    System.out.print(random1.nextInt(1000) + " ");
Random random2 = new Random(3);
System.out.print("\nFrom random2: ");
for (int i = 0; i < 10; i++)
    System.out.print(random2.nextInt(1000) + " ");
```

From random1: 734 660 210 581 128 202 549 564 459 961

From random2: 734 660 210 581 128 202 549 564 459 961

Static Değişkenler, sabitler ve metotlar

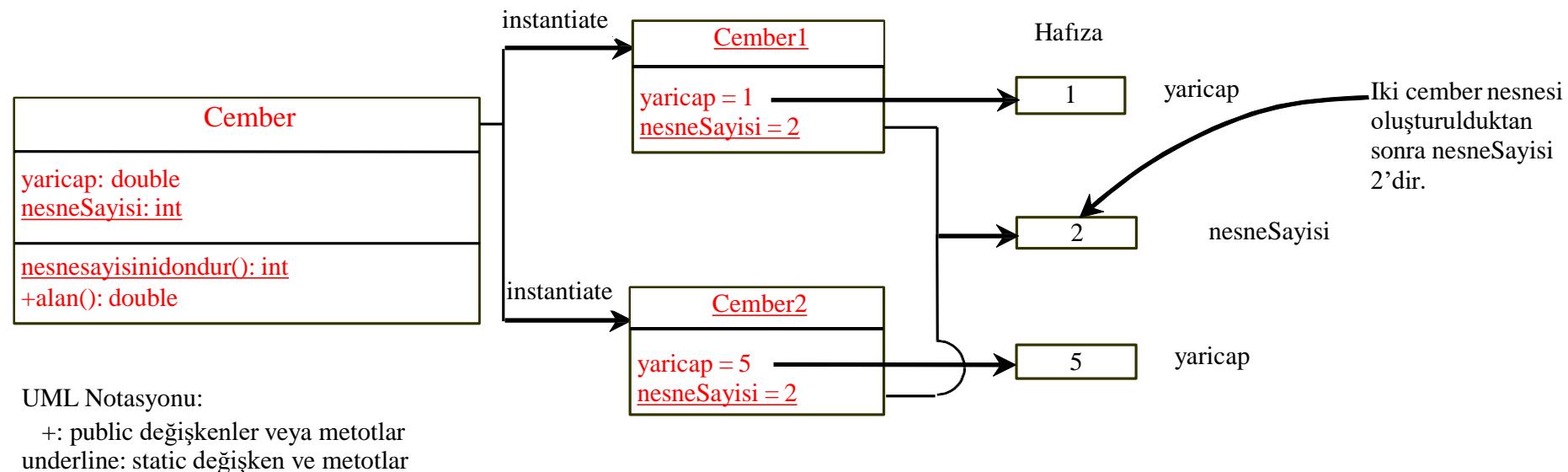
Statik değişkenler sınıfın bütün örnekleri tarafından paylaşılır.

Statik metotlar belirli bir nesneye bağlı değildir.

Statik sabitler sınıfın bütün örnekleri tarafından paylaşılan son değerlerdir.

Statik değişken, metot veya sabiti tanımlamak için static belirleyicisi kullanılır.

Statik Değişkenler, sabitler ve metotlar



Example of Using Instance and Class Variables and Method

Objective: Demonstrate the roles of instance and class variables and their uses. This example adds a class variable `numberOfObjects` to track the number of `Cember` objects created.

Cember2

TestCember2

Run

Tanımlayıcıların görünürügü

Varsayılan olarak bir sınıf, değişken veya metoda yanı paketteki herhangi bir sınıftan erişilebilir.



`public` Sınıf, metod veya veri herhangi bir paketteki herhangi bir sınıfa görülebilirdir.

`private`

Veri veya metodlara sadece tanımlanan sınıftan erişilebilir.

Get ve set metodları `private` özellikleri değiştirmek veya okumak için kullanılır.

```
package p1;
```

```
public class C1 {  
    public int x;  
    int y;  
    private int z;  
  
    public void m1() {}  
    void m2() {}  
    private void m3() {}  
}
```

```
public class C2 {  
    void aMethod() {  
        C1 o = new C1();  
        can access o.x;  
        can access o.y;  
        cannot access o.z;  
  
        can invoke o.m1();  
        can invoke o.m2();  
        cannot invoke o.m3();  
    }  
}
```

```
package p2;
```

```
public class C3 {  
    void aMethod() {  
        C1 o = new C1();  
        can access o.x;  
        cannot access o.y;  
        cannot access o.z;  
  
        can invoke o.m1();  
        cannot invoke o.m2();  
        cannot invoke o.m3();  
    }  
}
```

```
package p1;
```

```
class C1 {  
    ...  
}
```

```
public class C2 {  
    can access C1  
}
```

```
package p2;
```

```
public class C3 {  
    cannot access C1;  
    can access C2;  
}
```

Private tanımlayıcısı bir sınıf içinde erişimi sınırlar. Varsayılan tanımlayıcı bir paket içinde erişimi sınırlar. public tanımlayıcı ise sınırlayıcısız erişimi aktifleştirir.

NOT

Bir nesne kendi private üyelerine erişemez. (b)

Fakat, eğer nesne kendi sınıfında tanımlanmış ise, erişilebilir.

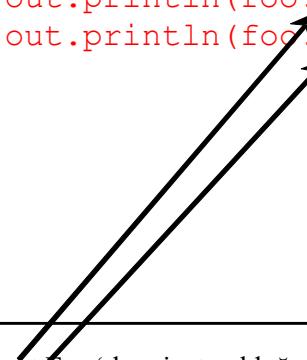
(a)

```
public class Foo {  
    private boolean x;  
  
    public static void main(String[] args) {  
        Foo foo = new Foo();  
        System.out.println(foo.x);  
        System.out.println(foo.convert());  
    }  
  
    private int convert(boolean b) {  
        return x ? 1 : -1;  
    }  
}
```

(a) foo nesnesi Foo sınıfının içinde tanımlandığı için kullanılabilir

```
public class Test {  
    public static void main(String[] args) {  
        Foo foo = new Foo();  
        System.out.println(foo.x);  
        System.out.println(foo.convert());  
    }  
}
```

(b) x ve convert Foo ‘da private olduğu için kullanım yanlışır



Veri alanı niçin private olması
gerekir?

Veriyi korumak

Sınıfı korumayı kolaylaştırmak için

Örnek

- işaretli private
tanımlayıcıyı
gosterir

	Cember	
	<u>-yaricap: double</u> <u>-nesneSayisi: int</u>	The radius of this circle (default: 1.0). The number of circle objects created.
	+Cember() +Cember(yaricap: double) +yaricapAl(): double +yaricapAta(yaricap: double): void + <u>nesneSayisiAl(): int</u> +Alan(): double	Constructs a default circle object. Constructs a circle object with the specified radius. Returns the radius of this circle. Sets a new radius for this circle. Returns the number of circle objects created. Returns the area of this circle.