

```

public class Siniflar
{
    public static void main(String []args)
    {
        tamsayi t = new tamsayi(5);
        Sout ("Sayı: " + t.sayi);
        Sout ("Asal: " + t.asalmi());
        Sout ("Tek: " + t.tekmi());
        Sout ("Cift: " + t.ciftmi());
    }
}

```

ÖRNEK

- Nokta isimli bir class tanımla
- Veri alanları double türünde x ve y olsun.
- Kurucu metot boş olsun (x=0, y=0)
- Double türünde değer (parametre) alan kurucu metot olsun.
- Double türünde metot olsun. Uzaklık hesaplasın.

```

class nokta
{
    double x;
    double y;

    nokta()
    {
        x = 0;
        y = 0;
    }
    nokta (double a, double b)
    {
        x = a;
        y = b;
    }
    double uzaklık()
    {
        return Math.sqrt(x*x + y*y);
    }
}

```

Not: "sqrt" fonksiyonu kök alır.

```

public class Ornek
{
    Public static void main ...
    {
        nokta nesne = new nokta();
        Sout(nesne, uzaklık());
    }
}

```

Proje adı

Diyagram

Nokta
x: double y: double
nokta() nokta(double, double) uzaklık(): double

ÖRNEK


```
class diziislemeler{
    int [] dizi;
    diziislemeler (int n)
    {
        dizi = new int[n];
        for( int i=0; i<dizi.length; i++)
            dizi[i] = (int)(Math.random()*100);
    }
    void dizi yaz()
    {
        for(int x:dizi)
            Sout(x+" ");
    }
    int [] enbenk()
    {
        int [] enbenk = { dizi[0], dizi[0] };
        for ( int i=1; i<dizi.length; i++ )
        {
            if ( enbenk[0] < dizi[i] )
                enbenk[0]=dizi[i];
            if ( enbenk[1] > dizi[i] )
                enbenk[1] = dizi[i];
        }
        return enbenk;
    }
}

// ANA PROGRAM

diziislemeler dizi = new diziislemeler(6);
dizi.dizi yaz();
int [] sonu = dizi.enbenk();
Sout( " \n En byk ve en kkler: " + sonu[0] + " " + sonu[1]);
```

STATIC DEĞİŞKENLER, METODLAR VE SABİTLER

class cember

```
{  
    double yarıcap;  static değil  
    cember()  
    {  
        yarıcap = 1.0;  
    }  
    cember(double x)  
    {  
        yarıcap = x;  
    }  
    double alanHesapla ()  
    {  
        return Math.PI*yarıcap*yarıcap;  
    }  
}
```

//Ana Program

ceMBER c = new cember();

//Bir sınıfın özel bir örneğine bağlanır. Aynı sınıfın nesneleri arasında paylaşılmaz. Mesela;

ceMBER c1 = new cember(2.0);

ceMBER c2 = new cember(3.8);

//c1.yarıcap

//c2.yarıcap

 ikisi de var. c1'in değeri 2, c2'nin değeri 3.8. İkisi aynı şeyi ifade etmiyor.

Not: Static değişkenler ortak bir hafızada değişken tutarlar

- Static metodları, sınıftan herhangi bir nesne oluşturmadan çağırabiliyoruz.

ÖRNEK Math.PI



static metod



Class static olamaz. Sadece metodlar ve değişkenler static olabilir.

ÖRNEK

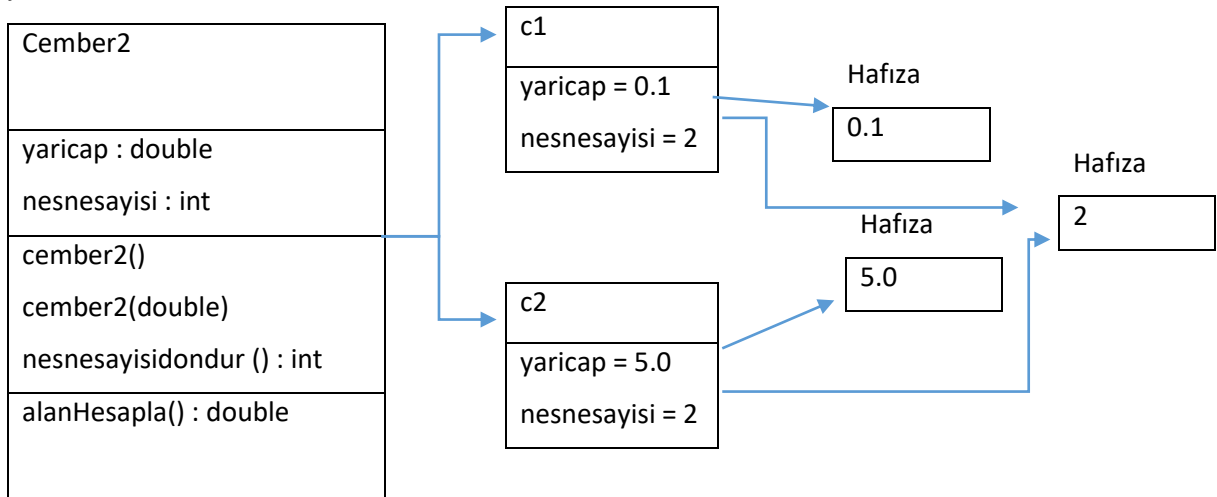
class cember2

```
{  
    double yarıcap;
```

```

static int nesnesayisi =0;
cember2()
{
    yaricap = 1.0;
    nesnesayisi ++;
}
cember2 (double ycap)
{
    yaricap =ycap;
    nesnesayisi ++;
}
static int nesnesayisidondur()
{
    return nesnesayisi;
}
double alanHesapla()
{
    return Math.PI-yaricap*yaricap;
}
}

```



```

Sout ( cember2, nesnesayisi );
    cember2.nesnesayisi = 999;

```

Yazabilirim

NOT: Yandaki örnekte yaricap static olmadığı için nesne oluşturarak yaricipi çağırman gerekiyor.

! Java kitabı sayfa 142'deki örneğe bak!

NOT: Static değişkenler ve metotlar ile static olmayan değişkenler ve metotlar sınıf içerisinde birlikte kullanılabilirler.

*** Static olmayan değişken ve metodlar sadece ve sadece static olmayan metodlar içerisinde çağrılabilir.

144

"Java kitabı sayfa 147-149'a bak!

KISACA STATİC KULLANMAMIZIN AMACI

- Metot ya da değişkene sınıf adıyla ulaşabilmek
 - Sınıftaki ortak hafıza alanını belirlemek
- ! Java kitabı sayfa 144 en alttaki nota örnek

publi class test

```
{
    public static int topla(int a, int b)
    {
        return a+b;
    }
    public static void main (String [] args)
    {
        Sout (topla(3,5));
    }
}
```

Static bir metot, static bir metodun içerisinde doğrudan çağrılabilir.

ÖRNEK

publis class deneme

```
{
    int i=5;
    static int k=2;
    public static void main(String [] args)
    {
        intj =i; //Yanlış
        m1(); // Yanlış
    }
    public void m1()
```

Çünkü static olmayan metodun içerisinde static olmayan değişken ya da metodu çağırabilmem için nesne oluşturmam gerekiyor.

```
{  
    i=i+k+m2(i,k);  
}  
public static int m2(int i, int j)  
{  
    return (int)(Math.Pow(i,j));  
}  
}
```

Erişim Belirleyiciler ! (Public, Private, Protected, friendly)

- 1) Public: Sınıf, metot veya değişkene herhangi bir sınıftan ulaşılabilir.
veri -> public int x
sınıf -> public class deneme
metot -> public int topla (int a, int b)
- 2) Private: Metot veya veri alanlarına sadece ve sadece tanımlandığı sınıf içerisinde ulaşılabilir.
- 3) Friendly: Public veya private yazılmamışsa. Sınıf, metot ve değişkenlere aynı paket içerisinde herhangi bir yerden ulaşılabilir
- 4) Protected: Miras almada (Kalıtım) işimize yarayacak

ÖRNEK

Aynı paketin içerisinde 2 tane class var.

Package P1;

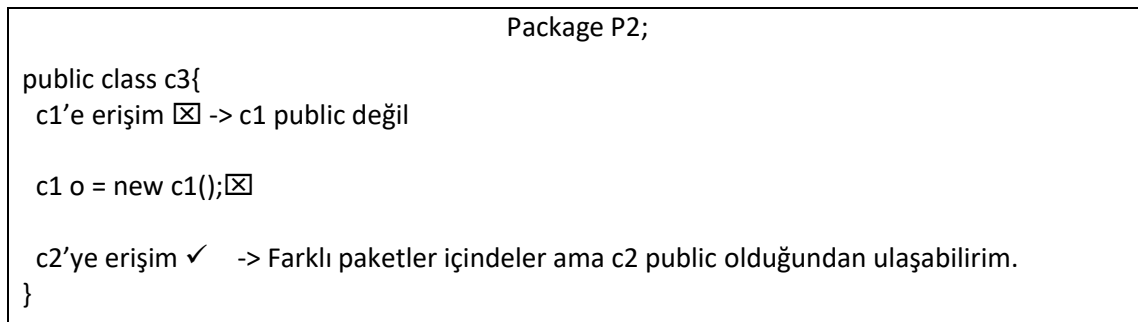
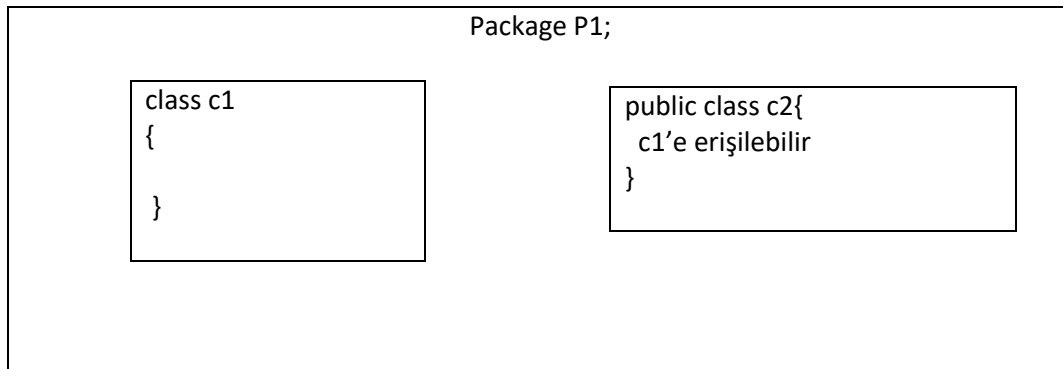
```
public class c1{  
    public static int x;  
    int y;  
    private int z;  
    public void m1()  
    {  
    }  
    friendly void m2()  
    {  
    }  
    private void m3()  
    {  
    }  
}
```

```
public class c2{  
    void metota()  
    {  
        c1 o = new c1(); ✓ (c1 sınıfı Public  
        olduğu için nesne oluşturabilirim)  
        o.x 'e erişim ✓  
        metota static değil.  
        Ama x static. Static olmayan metot içerisinde  
        static olmayan değişkeni çağırabilmem için  
        nesne oluşturmam gerekiyor. Biz de zaten o  
        nesneyi oluşturmuşuz.  
        o.y 'ye erişim ✓  
        public değil (y) ama aynı paket içerisinde  
        olduğu için ulaşabilirim.  
        o.z 'ye erişim ✗ (private olduğu için (z)  
        ulaşamam)  
        o.m1() çağırma ✓  
        o.m2() çağırma ✓ (Aynı paket içinde)  
        o.m3() çağırma ✗
```

Package P2;

```
public class c3{  
    void metot2()  
    {  
        c1 o = new c1(); ✓ (c1 sınıfı Public olduğu için nesne oluşturabilirim. Ama class c1  
        şeklinde olsaydı friendly olurdu ve nesne oluşturamazdım.)  
        o.x 'e erişim ✓  
  
        o.y 'ye erişim ✗ -> P1 ve P2 farklı paketler olduğundan  
  
        o.z 'ye erişim  
        o.m1() çağırma ✓ -> public olduğu için  
        o.m2() çağırma ✗ -> farklı paketler  
        o.m3() çağırma -> ✗
```

ÖRNEK



ÖRNEK

```
public class test
{
    private boolean x;
    public static void main (String [] args)
    {
        test t = new test();
        t.x = true; ✓ (Aynı sınıf içerisinde)
        Sout (t.x);
        Sout (t.donustur()); ✓ (Aynı sınıf içerisinde)
    }
    private int donustur ()
    {
        if(x==true)
            return 1;
        else
            return 0;
    }
}
```

Hatırlatma:

Static olmayan değişken veya metoda ulaşabilmem için static olan metot içerisinde nesne oluşturmam gerekiyor.


```
}  
}
```

NOT: Eer dışarıdan erişimi engellemek istiyorsak alanı private olarak tanımlamam gerekiyor.

VERİ ALANI KAPSÜLLEME



ÖRNEK

```
public class cember {  
    private double r = 1.0;  
    private static int nesnesayisi = 0;  
    public cember ()  
    {  
        nesnesayisini ++;  
    }  
    public cember (double rx)  
    {  
        nesnesayisini ++;  
        r = rx;  
    }  
    public void setr(double rx)  
    {  
        r = rx;  
    }  
    public double getr()  
    {  
        return r;  
    }  
    public int getnesnesayisi()  
    {  
        return nesnesayisi;  
    }  
}
```

Ana Program

```
public class test{  
    psv...{  
        cember c1 = new cember();  
        Sout (c1, r);  
        Sout (c1, getr());  
        c1.setr(3.8);  
    }  
}
```



nesnesayisi
kapsüllenmiş durumda

```
public double alanhesapla()
{
    return Math.PI*r*r;
```

Ya da

```
public class testnesnegecisi
{
    psv...
    {
        cember c = new cember(3.0);
        cemberyazdir( c ); (Static metot içerisinde static metodu doğrudan çağırabiliyordum.)
    }
}

public static void cemberyazdir(cember cember c2)
{
    Sout ("Cemberin yarıçapı : " + c2.getr() + "alanı ise : " + c2.alanhesapla());
}
```



Mesela bu kodu n'e kadar olan her r değerini hesaplaması için yazarsam;

```
public static void cemberyazdir(cember c, int n)
{
    int i = 0;
    while(i < n)
    {
        Sout ("Yarıçapı: " + c.getr() + " olan çemberin alanı : " + c.alanhesapla());
        c.setr(c.getr()+1);
        i++;
    }
}
```

NOT: - ULM diyagramında eğer bir metot veya değişken private ise önüne – konur.

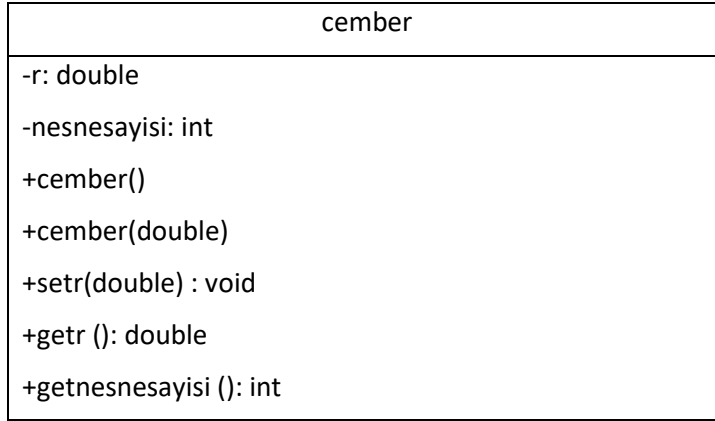
- Eğer public metot veya değişken ise +konur.

- Eğer protected metot veya değişken ise # konur.

-: private plan (değişken) veya metot

+: public olan veya metot

Yandaki örneğin UML diyogramı



METOTLARDA PARAMETRE OLARAK NESNE VERİLMESİ

Bir dizinin geçişi gibi bir nesnede metoda parametre olarak verilebilir. Bu durumda nesnenin geçişi normal değişken geçişinden çok farklı olamaz. Hatta metot, nesne bile döndürebilir.

!Java kitabı sayfa 127!

ÖDEV Dizi tanımla dizinin herbir hücresi

ad: string soyad: string vize: int genel: int ort: double

Bu diziye erişecek sınıfın not ortalamasını, herbir öğrencinin adını, soyadını, ortalamasını göster. En yüksek ve en düşük ortalamaya sahip öğrencinin adını, soyadını döndüren metot yaz.

----- * ----- * ----- * ----- * -----

Get bloğu veri okumaya, set bloğu veri yazmaya (değer atamaya) yarar.

Statik olmayan üyeler sınıftan türetilmiş nesnelerin üyeleridir. Yani sınıftan kaç nesne türetilmişse bu üyelerin o kadar kopyası bulunur. Örneğin;

```
public class Ornek
{
    int x;
}
// static değil
```

Ana Program

```
Ornek nesne1 = new Ornek();
nesne1.x;
Ornek nesne2 = new Ornek();
nesne2.x;
```



2 tane nesne oluşturdum.

Bir nesneye ait üyenin değerinin değişmesi o üyenin diğer nesnelerdeki değerini etkilemez. Örneğin;

```
public class Ornek
{
    String ad;
}
```

Ana Program

```
Ornek nesne1 = new Ornek();
nesne1.ad = "Büşra";
Ornek nesne2 = new Ornek();
nesne2.ad = "Damla";
```



2 tane nesne oluşturdum.

x'in 2 tane kopyası var.

```
Sout(nesne1.ad); //Ekranda Büşra yazar.
```

```
Sout(nesne2.ad); //Ekranda Damla yazar.
```

Static üyeler ise doğrudan sınıfın kendi üyeleri olup, sınıftan üretilen nesneden bağımsız hareket ederler. Tüm nesnelerin ulaşılacağı ortak bir bellek alan alanında tutunur. Bu nedenle static üyelere erişmek için sınıftan nesne üretmeye gerek yoktur. Bu da static üyelerde işlem yapmayı hızlandırır. Static üyelerin ilk değerleri atamamışsa çalışma anında üyenin tüne göre değer verilir.

SET – GET NEDEN KULLANILIR?

Bir değişken eğer private ise ona ulaşabilmek için set – get kullanırız.

- Static sınıfları nesneye bağımlı olmayan işlemlerin yapıldığı durumlarda tercih ederiz.
- İçerisinde herhangi bir yapıcı tanımlamaz, olamaz



```
new bisiklet()
```

bu yoksa yapıcıda yok.

ERİŞİM BELİRLEYİCİLER

Public: Tüm erişimlere açık

Private: Sadece üyesi olduğu sınıftan erişilebilir. Sınıf içerisindeki bir değişken veya yordamın private olarak işaretlenmesi ona dışarıdan hatta bu sınıftan türetilmiş sınıflardan bile erişilemeyeceği anlamına gelir.

Protected: Üyesi olduğu sınıftan ve bu sınıftan türetilmiş sınıflarlardan ulaşılır.

ÖRNEK

```
package ornek;
```

```
import java.util.Scanner;
```

```
class cember
```

```
{
```

```
    static int ns=0;
```

```
    double r;
```

```
    cember()
```

```
    {
```

```
        r=1.0;
```

```
        ns++;
```

```
    }
```

```
    cember(double rx)
```

```
    {
```

```
        r=rx;
```

```
    }
```

```
    double alanHesapla()
```

```
    {
```

```
        return Math.PI*r*r;
```

```
    }
```

```
}
```

```
public class Ornek
```

```
{
```

```
    public static void main (String []args)
```

```
    {
```

```
        cember c = new cember();
```

```
        Sout("Yaricap: " + c.r);
```

```
        Sout("Yaricapi : " + c.r + " olan cember alanı : " + c.alanHesapla());
```

```
        Sout("Nesne Sayısı : " + c.ns);
```

```
        cember c1 = new cember();
```

```
        Sout("Yaricapi : " + c1.r + " olan cemberin alanı : " + c1.alanHesapla());
```

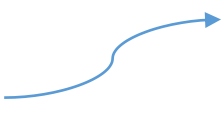
```
        Sout("Nesne sayısı : " + c1.ns);
```

****NOT :** Aynı dosyanın içinde iki tane public class olamaz.

ÖRNEK

Class içinde sayı alsın parametre olarak. Asal olup olmadığını tek mi? çift mi? olduğunu bulsun.

ns'yi static yapmazsak nesne sayısını 1 olarak verilir.



```

class sinif3
{
    int a;
    sinif3(int x)
    {
        a=x;
    }
    String tekmi()
    {
        if (a % 2 == 1)
            return "tek";
        else
            return "çift";
    }
    String asalmi()
    {
        String asal = "asal";
        for (int i=1; i<a/2; i++)
        {
            if( a% i == 0 )
            {
                asal = "Asal değil";
            }
        }
        return asal;
    }
}

public class Test{
    public static void main(String []args)
    {
        sinif3 [] s = new sinif3[10];
        for ( int i=0; i<10; i++)
        {
            int a = (int) (Math.random()*100);

```

```

        s[i] = new sinif3(a); (s[i] = 0 gibi düşün)
        Sout ( s[i].a + " " + s[i].tekmi() + " " + s[i].asalmi() + "\n");
    }
}

```

ÖRNEK

class öğrenci

```

{
    int no;
    String isim;
    static String bolum = "bilgisayar";
    static void degis()
    {
        bolum ="elektrik";
    }
    öğrenci(int r, String n)
    {
        no = r;
        isim = n;
    }
    void göster(){
        degis();
        Sout(no + " " + isim + " " + bolum);
    }
}

```

public class ornek2{

```

    public static void main (String [] args)
    {
        öğrenci.degis();
        öğrenci S1 = new öğrenci (111, " Busra");
        öğrenci S2 = new öğrenci (222, " Damla");
        öğrenci S1 = new öğrenci (333, " Sevgi");
        S1.goster();
        S2.goster();
    }
}

```

```

        S3.goster();
    }
}

```

Proje Adı: Erişimler

ÖRNEK Paket ve class oluşturma !!!

Paket ve class oluşturmak için sol taraftaki projects ya da files kısmındaki “src” kısmını kullanmamız gerekiyor.

sinif1 kısmı

```

public class sınıf1{
    int a;

    sınıf1 ( int ax)
    {
        a=ax;
    }

    int faktoryel()
    {
        int f=1;
        for(int i=1; i<a; i++)
        {
            f=f*i;
        }
        return f;
    }

    private boolean tekmi( int i )
    {
        return ( i % 2 == 1);
    }

    int tektoplam()
    {
        int t=0;
        for(int i=0; i<a; i++)
        {

```

Erisimler.java kısmı

```

package erisimler;

import erisimler.sinif1;

public class Erisimler{
    psvm ...
    {
        sınıf1 s = new sınıf1 (5);
        Sout (s.Faktoryel());
        Sout(s.tektoplam());
    }
}

```

Projects Kısmı

+ erisim2

sinif3.java

+ erisimler

sinif1.java

Erisimler.java


```

        if(terim(i))
        {
            t=t+i;
        }
    }
    return t;
}
}

```

DEĞİŞKENLERİN TANIM ARALIĞI

ÖRNEK

```
for ( int i=0; i<10; i++)
```

```

{
    Sout(i);    ☒ Hatalı
}

```

i=5; // Hata verir. Çünkü i'nin tanım aralığı _
kısımında

ÖRNEK

```

int i;
for(int i=0; i<10; i++)
{
    Sout(i);    ✓Doğru
}

```

i=8; // Çalışır. Çünkü i'nin tanım aralığı dışarıda

```
Sout(i);
```

SINIF DEĞİŞKENLERİ

(VERİ PLANI)

```

public class cember{
    public cember(){
        r=1.0;
    }
    public cember(double rx) {
        r=rx;
    }
    public double alanHesapla() {
        return Math.PI*r*r;
    }
    public double r;
}

```

```

public class test
{
    private int i=2;
    private int j=i+5;
}

```

Böyle bir durum varsa
i'nin önceden tanımlanması gerekiyor.
Çünkü j'nin değeri i'ye bağlı

ÖRNEK

```
public class fx
```

```
{
```

```
    int x=0; //Sınıf referans değişkeni
```

```
    int y=0;
```

```
    public fx()
```

```
    {
```

```
    },
```

```
    void P()
```

```
    {
```

```
        int x=1; // Lokal değişken
```

```
        Sout("x:" + x);
```

```
        Sout("y:" + y);
```

```
    }
```

```
    public class test
```

```
    {
```

```
        psvm()
```

```
        {
```

```
            fx f = new fx();
```

```
            f.P(); -> Yazdığımızda ekrana x:1
```

```
                    y:0
```

yazar

Metot içerisinde bir tanımlama yapılmışsa lokal değişkendir.

This Anahtar Kelimesi

```
class cember{
```

```
    double r;
```

```
    cember()
```

```
    {
```

```
        r=1.0;
```

```
    }
```

```
    cember(double r)
```

```
    {
```

```
        this.r=r;
```

```
    }
```

```
    double alanHesapla()
```

```
    {
```

```
        return Math.PI*r*r;
```

```
        // ya da
```

```
        return Math.PI*this.r*this.r
```

```
    }
```

```
}
```

```
public class test{
```

```
    psvm(){
```

```
        cember c = new cember(3.2);
```

```
        Sout ( this.r); ☒
```

```
        Sout (c.r); ✓
```

```
}
```

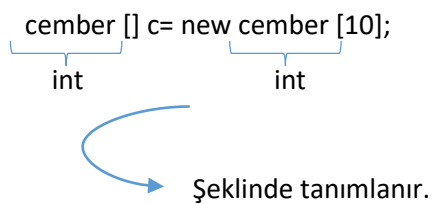
ÖRNEK

```
class sinif1
{
    int i=5;
    static double k=0;
    void set2(int i)
    {
        this.i=i;
    }
    static void setk(double k)
    {
        sinif1.k=k;
        //this.k yerine yazabiliriz.
    }
}

class test{
    psvm()
    {
        sinif1 s1 = new sinif1();
        sinif1 s2 = new sinif2();
        s1.set2(10);
        s2.i=25;
        sinif1.setk(15);
    }
}
```

-NESNE DİZİLERİ-

```
cember [] c= new cember [10];
```



Şeklinde tanımlanır.

c

referans

c[0]
c[1]
.
.
.
c[9]

→ cember nesne 0

→ cember nesne 1



1. eleman string

2. eleman int

→ cember nesne 9

olabilir

ÖRNEK

```
cember [] c= new cember [10];
```

int int

```
int s=1;
```

```
for ( int i=0, i<10; i++)
```

```
{
```

```
    c[i] = new cember(s);
```

```
    s++;
```

```
}
```

cember nesne 0 ->

c[0]
r:1

cember nesne 1 ->

c[1]
r:2

.

.

.

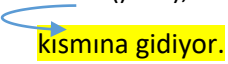
cember nesne 3 ->

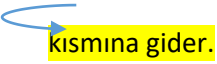
c[3]
r:3

ÖRNEK

cember sınıfından bir nesne tanımla. Yarıçap değerlerini, nesnelerin alanları ve alanlar toplamını bulalım.

```
class cember
{
    double r;
    cember()
    {
        r=1.0;
    }
    cember (double r)
    {
        this.r =r;
    }
    double alanHesapla()
    {
        return Math.PI*r*r;
    }
    public static cember[] olusturCemberdizi()
    {
        cember[] c = new cember[10];
        for( int i=0; i< c.length; i++ )
        {
            c[i] = new cember (Math.random()*20);
        }
        return c;
    }
    public static void yazCemberdizi( cember [] c)
    {
        Sout( "r \t\t\t " + "Alan");
        for ( int i=0; i<c.length; i++)
        {
            Sout(c[i].r + c[i].alanhesapla() + "\n");
        }
    }
}
```

kismına gidiyor.

kismına gider.

-KONUYU ÖZETLEYEN ÖRNEK-

ÖRNEK Rasyonel Sayılar Üzerinde İşlem Yapan Bir Sınıf Oluşturalım.

Rasyonel ->Sınıf Adı

Rasyonel () -> pay=0 payda=1 (ikisi de private)

Rasyonel(pay, payda) -> this.pay = pay, this.payda = payda

public int obeb (int x, int y) ->obebi geri döndürür.

public void sadeleştir()

public void uzerineekle (Rasyonel s)

public static Rasyonel topla (Rasyonel s1, Rasyonel s2)

```
.....

public class Rasyonel
{
    public int pay;
    public int payda;
    public Rasyonel()
    {
        pay=0;
        payda =1;
    }
    public Rasyonel(int pay, int payda)
    {
        this.pay= pay;
        this. payda = payda;
    } public Rasyonel()
    {
        pay=0;
        payda =1;
    }
    public void yaz()
    {
        Sout(pay +"\" +payda);
    }
    private int obebAl( int x, int y)
    {
        int kucuk;
        if(x>y) kucuk=x;
        else kucuk=y;

        for (int i=kucuk;i>2;i--)
        {
            if(x%i == 0 && y%i==0)
            {
                return i;
            }
        }
    }
}
```

```
    }  
}  
    return 1;  
}
```

```
public void sadeleştir()  
{  
    int o = obebAl(pay, payda);  
    pay = pay/o;  
    payda = payda/o;  
}  
public void uzerekle(Rasyonel s)  
{  
    this.pay = this.pay*s.payda + this.payda*s.pay;  
    this.payda = this.payda*s.payda;  
}
```