

BBS515 Nesneye Yönelik Programlama

Ders 9 - Sınıflar - Kalıtım

Zümra Kavafoğlu
<https://zumrakavafoglu.github.io/>

Soru 1 - Karmaşık Sayılar

Karmaşık sayılar $z = a + b.i$ formunda tanımlanan sayılara verilen addır. Bu tanımda z karmaşık sayı, a sayının Reel bileşeni, b ise imajiner bileşeni olarak tanımlanır. İmajiner bileşeni temsil eden i değeri $i^2 = -1$ olarak tanımlanmıştır. Toplama işleminde karmaşık sayının reel ve imajiner bileşenleri ayrı ayrı toplanır, örnek $z_1 = a + bi$, $z_2 = c + di$ ise $z_1 + z_2 = (a + c) + (b + d)i$ olmaktadır. Çarpma işlemi $z_1 \times z_2 = (bd - ac) + (ad + bc)i$ şeklinde olmaktadır.

KarmasikSayi
- imajiner : double - reel : double
+ KarmasikSayi(double, double)
+ carp(KarmasikSayi) : KarmasikSayi + topla(KarmasikSayi) : KarmasikSayi + esitmi(KarmasikSayi) : boolean

SINIF DEĞİŞKENLERİ (STATİK DEĞİŞKENLER)

- ❧ Bir sınıfın içinde tanımladığımız değişkenler, *sınıf değişkenleri*(*statik değişkenler*) ve *nesne değişkenleri* (*instance variables*) olmak üzere ikiye ayrılır:
- ❧ Bundan önce gördüğümüz değişkenler nesne değişkenleriydi. Yani bu değişkenlerin değerleri, oluşturulan nesneye özeldi.

NESNE DEĞİŞKENİNE ÖRNEK:

☞ Örneğin, şu iki nesneyi oluşturduğumuzu düşünelim:

```
Circle circle1 = new Circle();  
Circle circle2 = new Circle(5);
```

☞ radius değişkeni bir nesne değişkeni olduğundan:

- ☞ circle1 nesnesinin radius değişkeni, circle2 nesnesinin radius değerinden bağımsızdır.
- ☞ İkisi hafızada farklı yerlerde tutulurlar.
- ☞ Birinde yapılan bir değişiklik diğerini etkilemez.

Peki ya ortak deęiřtirilebilir bir deęiřken tanımlamak gerekirse ?



❧ Circle sınıfından kaç tane üretildiğini takip eden bir deęiřken tanımlamak istiyorum. Bir nesne deęiřkeni işime yarar mı?

SINIF DEĞİŞKENLERİ (STATİK DEĞİŞKENLER)

- ❧ Eğer bir veriyi, bir sınıftan ürettiğiniz tüm nesnelerin paylaşmasını istiyorsanız o zaman statik değişken tanımlamalısınız.
- ❧ Statik değişkenler tanımlanırken başlarına *static* yazılır.
- ❧ Statik değişkenleri çağırırken bir nesne ile çağırmaya gerek yoktur, direkt sınıf adıyla çağırılabilir:

```
static int numberOfCircles;
```

```
Circle.numberOfCircles
```

```
package package1;
```

```
public class Circle {
```

```
    private double radius;
```

```
    static int numberOfCircles;
```

```
    public Circle()
```

```
{
```

```
        radius = 1.0;
```

```
        numberOfCircles++;
```

```
}
```

```
    public Circle(double r)
```

```
{
```

```
        radius = r;
```

```
        numberOfCircles++;
```

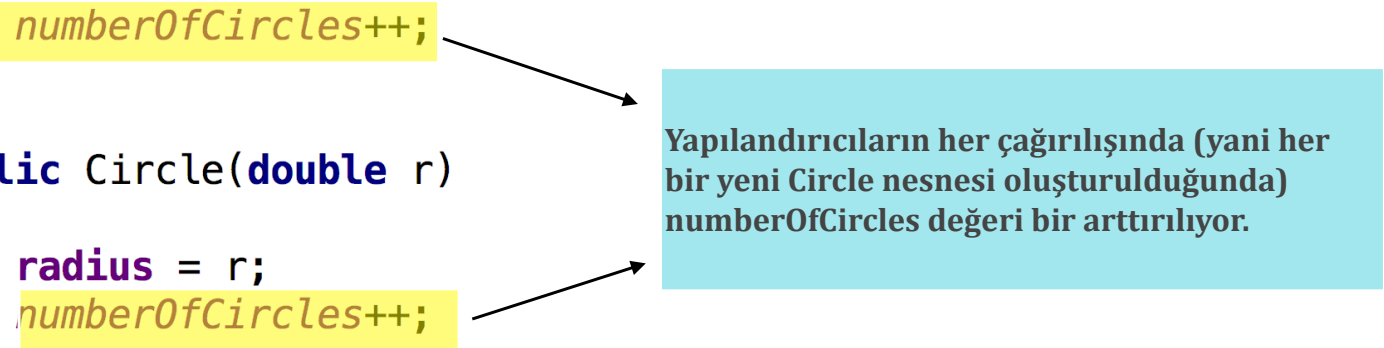
```
}
```

```
    public double getRadius() { return radius; }
```

```
    public void setRadius(double newRadius) { radius = newRadius; }
```

```
    public double findArea() { return radius*radius*Math.PI; }
```

```
}
```



Yapılandırıcıların her çağırılışında (yani her bir yeni Circle nesnesi oluşturulduğunda) numberOfCircles değeri bir arttırılıyor.


```
package package1;
```

```
public class TestCircle {
```

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

```
        Circle c1 = new Circle();
```

```
        System.out.println("Number of circles created: " + Circle.numberOfCircles );
```

```
        Circle c2 = new Circle(5);
```

```
        System.out.println("Number of circles created: " + Circle.numberOfCircles);
```

```
        Circle c3 = new Circle(3);
```

```
        System.out.println("Number of circles created: " + Circle.numberOfCircles);
```

```
    }
```

```
}
```


SINIF METOTLARI (STATİK METOTLAR)

- ❧ Sınıf metotları için metot tanımlamasının başına static niteleyicisi yazılır.

```
public static int getNumOfObects()
```

- ❧ Sınıf metotları da aynen statik değişkenler gibi, sınıfın bir nesnesi üretilmeden çağırılabilirler.

```
Circle.getNumOfObects()
```

ÖNEMLİ NOT



☞ Nesne değişkenleri, nesne metotlarında çağırılabilirler, ancak nesne değişkenleri statik metotların içinde çağırılamazlar.

```
public class Foo {  
  
    int i=5;  
  
    static void aStaticMethod()  
    {  
        System.out.println(i);  
    }  
}
```

Hatalı!!
Çünkü i, statik olmayan bir değişken ama aStaticMethod statik bir metot.

DEĞİŞKEN KAPSAMI (SCOPE OF VARIABLES)

- Bir sınıf ya da nesne değişkeni bir nesnenin belirli bir özelliğini tanımlamak için kullanılır. Bu değişkenler sınıfın herhangi bir yerinde direk tanınır ve kullanılabilir. Bu yüzden bu değişkenlere **global değişken** denir.
- Bir metodun içinde tanımlanan değişken ise sadece o metod içinde kullanılabilir. Bu yüzden bu değişkenlere **yerel(local) değişken** denir.

```
public class Foo {  
  
    int x=0; //nesne deęiřkeni (global deęiřken)  
    int y=0;  
  
    void p()  
    {  
        int x=1; //yerel deęiřken  
        System.out.println("x = " + x);  
        System.out.println("y = " + y);  
    }  
}  
public class TestFoo {  
  
    public static void main(String[] args) {  
        Foo f = new Foo();  
        f.p();  
    }  
}
```

x=1

y=0

this anahtar kelimesi

- Bir önceki örnekte gördüğümüz gibi eğer yerel bir değişken sınıf ya da nesne değişkeniyle aynı isme sahipse, öncelik yerel değişkenindir. Bu karışıklığı engellemek için this anahtar kelimesi kullanılır.
- Bir değişkenin sınıf ya da nesne değişkeni olduğunu belirtmek için this anahtar kelimesi kullanılır.

```
public class Foo {  
  
    int x=0; //nesne değişkeni (global değişken)  
    int y=0;  
  
    void p()  
    {  
        int x=1; //yerel değişken  
        System.out.println("x = " + this.x);  
        System.out.println("y = " + y);  
    }  
}  
public class TestFoo {  
  
    public static void main(String[] args) {  
        Foo f = new Foo();  
        f.p();  
    }  
}
```

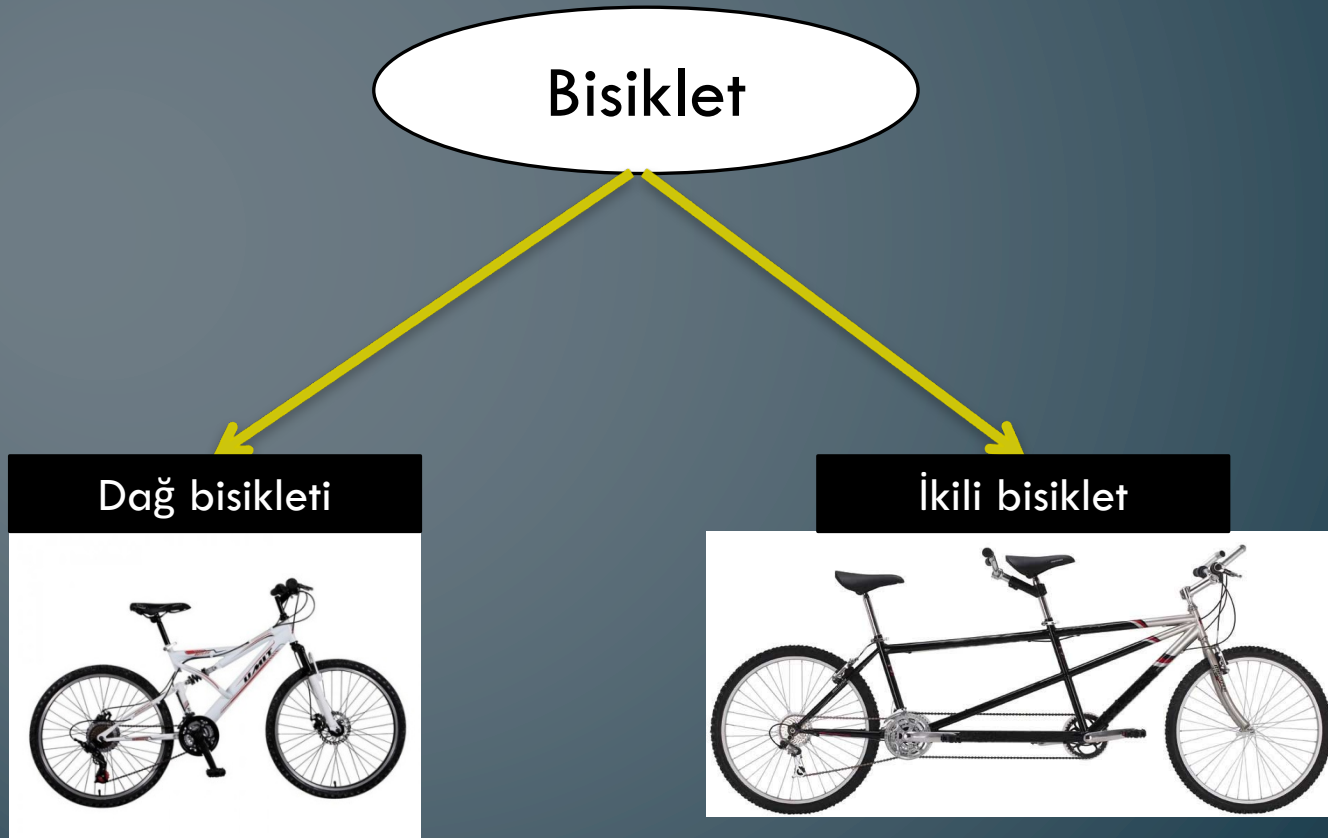
x=0

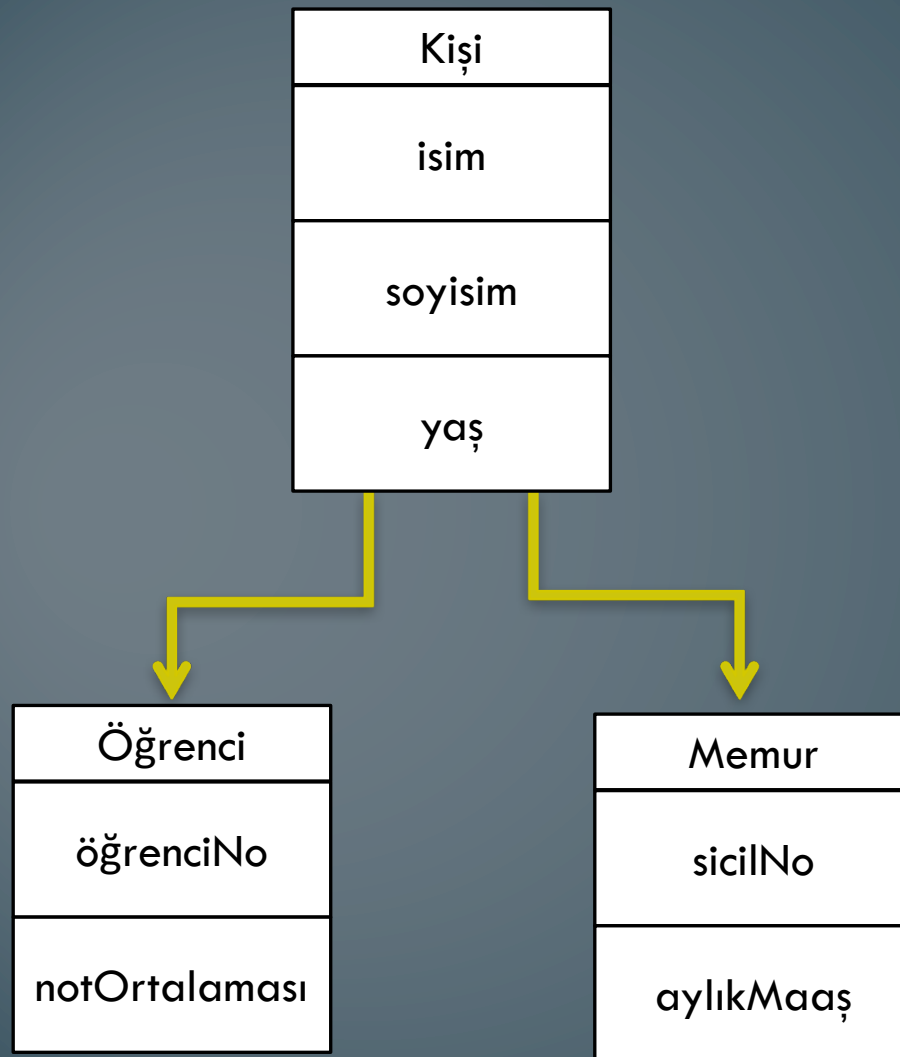
y=0

The background of the slide features a series of vertical lines in various shades of blue and grey, creating a textured, rain-like effect. A solid dark blue horizontal band spans the width of the slide, serving as a background for the title text.

KALITIM(INHERITANCE)

- Farklı nesnelerin ortak bir takım özellikleri olabilir.





Kişi: Superclass
(Parent class ya da Base class)

Öğrenci – Memur:
Subclass (child class, extended class ya da derived class)

Person süper sınıfı

```
public class Person {  
    protected String name, surname;  
  
    public Person() { name=surname="unknown"; }  
  
    public Person(String _name, String _surname)  
    {  
        name = _name;  
        surname = _surname;  
    }  
  
    public void setName(String _name) { name = _name; }  
  
    public void setSurname(String _surname) { surname = _surname; }  
  
    public String getName() { return name; }  
  
    public String getSurname() { return surname; }  
}
```

```
public class Student extends Person {

    private String studentNumber;
    private double average;

    public Student(){
        super();
        studentNumber = "unknown";
        average = 0;
    }

    public Student(String _name, String _surname, String _studentNumber, double _average)
    {
        super(_name,_surname);
        studentNumber = _studentNumber;
        setAverage(_average);
    }

    public void setStudentNumber(String _studentNumber) { studentNumber = _studentNumber; }

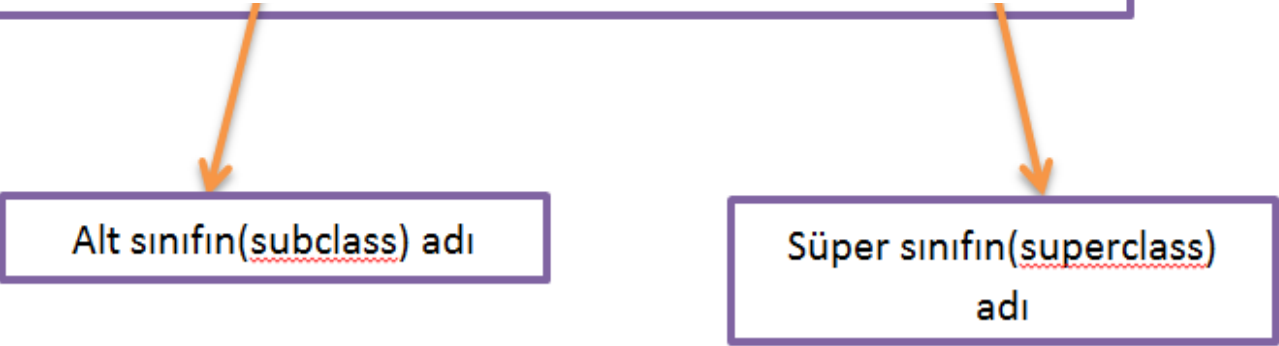
    public void setAverage(double _average)
    {
        if(average<0)
        {
            System.out.println("Hatalı girdi! Ortalama otomatik olarak 0 yapıldı");
            average = 0;
        }
        else
            average = _average;
    }

    public String getStudentNumber() { return studentNumber; }

    public double getAverage() { return average; }
```

Alt sınıf tanımlanması

```
public class Student extends Person
```



Alt sınıfın(subclass) adı

Süper sınıfın(superclass)
adı

- **NOT:** Bir sınıfın birden fazla alt sınıfı olabilir ancak sadece bir tane süper sınıfı olabilir.
- Bir alt sınıf, süper sınıfının private olmayan veri ve metotlarını doğrudan çağırabilir.

protected görünürlük niteleyicisi

```
public class Person {  
    protected String name, surname;  
  
    public Person() { name=surname="unknown"; }  
  
    public Person(String _name, String _surname)  
    {  
        name = _name;  
        surname = _surname;  
    }  
  
    public void setName(String _name) { name = _name; }  
  
    public void setSurname(String _surname) { surname = _surname; }  
  
    public String getName() { return name; }  
  
    public String getSurname() { return surname; }  
}
```

protected görünürlük niteleyicisi

- protected niteleyicisine sahip veri ve metotlar,
 - aynı paket içindeki diğer tüm sınıflar
 - sınıfın altsınıfları(subclasses) (aynı paket içinde olmasalar bile) tarafından erişilebilirdir.

package p1

```
public class Class1
```

```
    protected int x
```

```
public class C3
```

```
    Class1 c1;  
    c1.x erişilebilirdir.
```

package p2

```
public class Class2 extends Class1
```

```
    x bu sınıfta  
    erişilebilirdir.
```

```
public class Class4
```

```
    Class1 c1;  
    c1.x erişilebilir değildir.
```


super anahtar kelimesi

- Süper sınıfın yapılandırıcısını çağırmak için *super* anahtar kelimesi kullanılır
- Eğer alt sınıfta bir metodun ya da değişkenin superclass'a ait olduğu belirtilmek isteniyorsa, o zaman *super.değişkenAdı* ya da *super.metotAdı* biçiminde yazılır.

```
public class Student extends Person {

    private String studentNumber;
    private double average;

    public Student(){
        super();
        studentNumber = "unknown";
        average = 0;
    }

    public Student(String _name, String _surname, String _studentNumber, double _average)
    {
        super(_name,_surname);
        studentNumber = _studentNumber;
        setAverage(_average);
    }

    public void setStudentNumber(String _studentNumber) { studentNumber = _studentNumber; }

    public void setAverage(double _average)
    {
        if(average<0)
        {
            System.out.println("Wrong input!");
            average = 0;
        }
        else
            average = _average;
    }

    public String getStudentNumber() { return studentNumber; }

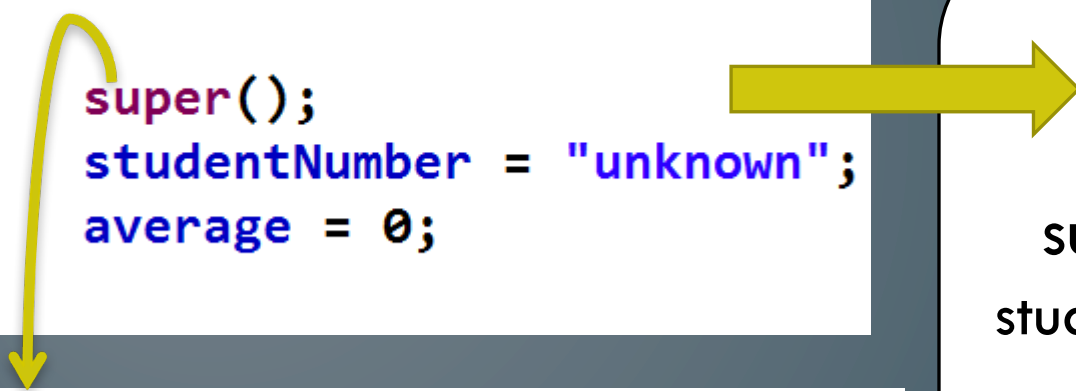
    public double getAverage() { return average; }
```

SUPERCLASS YAPILANDIRICISININ ÇAĞIRILMASI

- Superclass varsayılan yapılandırıcısı **super()** komutu ile çağırılır.

```
Student()  
{  
    super();  
    studentNumber = "unknown";  
    average = 0;  
}
```

```
Person()  
{  
    name = surname = "unknown";  
}
```



name=unknown
surname=unknown
studentNumber=unknown
average = 0

```
public class Student extends Person {

    private String studentNumber;
    private double average;

    public Student(){
        super();
        studentNumber = "unknown";
        average = 0;
    }

    public Student(String _name, String _surname, String _studentNumber, double _average)
    {
        super(_name, _surname);
        studentNumber = _studentNumber;
        setAverage(_average);
    }

    public void setStudentNumber(String _studentNumber) { studentNumber = _studentNumber; }

    public void setAverage(double _average)
    {
        if(average<0)
        {
            System.out.println("Wrong input!");
            average = 0;
        }
        else
            average = _average;
    }

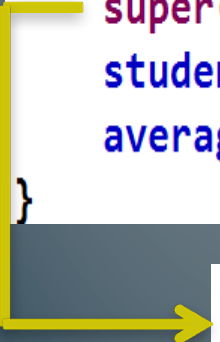
    public String getStudentNumber() { return studentNumber; }

    public double getAverage() { return average; }
```

SUPERCLASS YAPILANDIRICISININ ÇAĞIRILMASI

- Süper sınıfın varsayılan olmayan yapılandırıcısı ise `super(parametre listesi)` komutu ile çağırılır.

```
Student(String _name, String _surname, String _studentNumber, double _average)
{
    super(_name, _surname);
    studentNumber = _studentNumber;
    average = _average;
}
```



```
Person(String _name, String _surname)
{
    name = _name;
    surname = _surname;
}
```

```
public class TestPerson {  
  
    public static void main(String[] args) {  
        Person p = new Person("Ahmet", "Demir");  
        System.out.println(p.getName());  
  
        Student s = new Student("Mehmet", "Demir", "20217898", 70);  
        System.out.println(s.getName());  
        System.out.println(s.getStudentNumber());  
  
    }  
}
```

Süper sınıf Person'ın
getName metodu

```
public class TestPerson {  
  
    public static void main(String[] args) {  
        Person p = new Person("Ahmet", "Demir");  
        System.out.println(p.getName());  
  
        Student s = new Student("Mehmet", "Demir", "20217898", 70);  
        System.out.println(s.getName());  
        System.out.println(s.getStudentNumber());  
    }  
}
```

Süper sınıf Person'ın
getName metodu

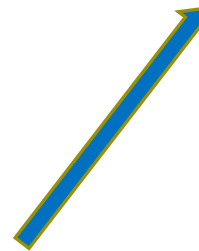
Student sınıfının
getStudentNumber
metodu


```
public class Circle {  
    private double radius;  
  
    public Circle() { radius = 1.0; }  
  
    public Circle(double r) { setRadius(r); }  
  
    public double getRadius() { return radius; }  
  
    public void setRadius(double newRadius)  
    {  
        if(newRadius<0)  
        {  
            System.out.println("Wrong radius value!");  
            radius=1;  
        }  
        else  
            radius = newRadius;  
    }  
  
    public double findArea() { return radius*radius*Math.PI; }  
}
```

```
public class Cylinder extends Circle {  
  
    private double height;  
  
    public Cylinder()  
    {  
        super();  
        height = 1.0;  
    }  
  
    public Cylinder(double _radius, double _height)  
    {  
        super(_radius);  
        setHeight(_height);  
    }  
  
    public void setHeight(double _height)  
    {  
        if(_height < 0)  
        {  
            System.out.println("Wrong height value");  
            height = 1;  
        }  
        else  
            height = _height;  
    }  
  
    public double getHeight() { return height; }  
  
    public double findArea() { return 2*super.findArea()+2*Math.PI*getRadius()*height; }  
  
    public double findVolume()  
    {  
        return super.findArea()*height;  
    }  
}
```

```
public class Cylinder extends Circle {  
    private double height;  
  
    public Cylinder()  
    {  
        super();  
        height = 1.0;  
    }  
  
    public Cylinder(double _radius, double _height)  
    {  
        super(_radius);  
        setHeight(_height);  
    }  
  
    public void setHeight(double _height)  
    {  
        if(_height < 0)  
        {  
            System.out.println("Wrong height value");  
            height = 1;  
        }  
        else  
            height = _height;  
    }  
  
    public double getHeight() { return height; }
```

findArea metodunun
yeniden yazılması
(method overriding)



```
public double findArea() { return 2*super.findArea()+2*Math.PI*getRadius()*height; }
```

```
public double findVolume()  
{  
    return super.findArea()*height;  
}
```

METHOD OVERRIDING

- Bazen subclass'ın superclass'da tanımlanmış metotları değiştirmesi gerekebilir. Buna *method overriding* denir.


Circle

```
public double findArea()  
{  
    return radius*radius*Math.PI;  
}
```

Cylinder

```
public double findArea()  
{  
    return 2*getRadius()*getRadius()*Math.PI+(2*getRadius()*Math.PI)*height;  
}
```

```
Cylinder myCylinder = new Cylinder(5,2);  
myCylinder.findArea()
```



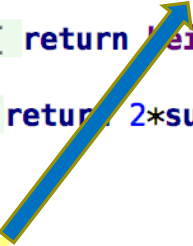
Circle sınıfının
findArea metodu

```
public class TestCylinder {  
    public static void main(String[] args){  
        Circle myCircle = new Circle(3);  
        System.out.printf("Area of circle is: %.2f \n", myCircle.findArea());  
  
        Cylinder myCylinder = new Cylinder(5,2);  
        System.out.printf("Area of cylinder is: %.2f \n", myCylinder.findArea());  
    }  
}
```

Cylinder sınıfının
findArea metodu

```
public class Cylinder extends Circle {  
  
    private double height;  
  
    public Cylinder()  
    {  
        super();  
        height = 1.0;  
    }  
  
    public Cylinder(double _radius, double _height)  
    {  
        super(_radius);  
        setHeight(_height);  
    }  
  
    public void setHeight(double _height)  
    {  
        if(_height < 0)  
        {  
            System.out.println("Wrong height value");  
            height = 1;  
        }  
        else  
            height = _height;  
    }  
  
    public double getHeight() { return height; }  
  
    public double findArea() { return 2*super.findArea()+2*Math.PI*getRadius()*height; }  
  
    public double findVolume()  
    {  
        return super.findArea()*height;  
    }  
}
```

Circle superclass'in
findArea metodu



Subclass of a subclass

Kişi
isim
soyisim
yaş



Öğrenci
öğrenciNo
notOrtalaması



Lisans Öğrencisi
devamsızlık

```
public class UnderGradStudent extends Student{

    private int absentDays;

    public UnderGradStudent()
    {
        super();
        absentDays = 0;
    }

    public UnderGradStudent(String _name, String _surname,String _studentNumber, double _average, int _absentDays)
    {
        super(_name,_surname,_studentNumber,_average);
        setAbsentDays(_absentDays);
    }

    public int getAbsentDays() {
        return absentDays;
    }

    public void setAbsentDays(int _absentDays){
        if(_absentDays < 0){
            System.out.println("Wrong input!");
            absentDays = 0;
        }else{
            absentDays = _absentDays;
        }
    }

}
```


Subclass of a subclass

```
public class TestPerson {  
    public static void main(String[] args) {  
        Person p = new Person("Ahmet", "Demir");  
        System.out.println(p.getName());  
  
        Student s = new Student("Mehmet", "Demir", "20217898", 70);  
        System.out.println(s.getName());  
        System.out.println(s.getStudentNumber());  
  
        UnderGradStudent u = new UnderGradStudent("Ali", "Demir", "2098767", 80, 10);  
        System.out.println(u.getSurname());  
        System.out.println(u.getAbsentDays());  
    }  
}
```

final niteleyicisi

- final niteleyicisinin değişkenler için kullanımını görmüştük.
- final niteleyicisini bir sınıfı nitelemek için kullanırsak, o sınıfın herhangi bir alt sınıfı oluşturulamaz.

```
public final class Circle
```

```
public class Cylinder extends Circle
```



Hatalı! Circle sınıfı final olduğundan, Cylinder Circle'ın alt sınıfı olamaz.

Object Sınıfı

- Javadaki her sınıf `java.lang.Object` sınıfından kalıttır ve `Object` sınıfının Javada yazılan her sınıf tarafından kalıtsal olarak kullanılabilen metotları vardır.

Object Sınıfı: toString metodu

Bu metot birlikte çağırıldığı nesneyi temsil eden bir String döndürür. Bu metodun default implementasyonu nesnenin sınıfının ismi, sınıfın içinde bulunduğu paket ismi ve hashcode yazdırır.

Bu metot override edilerek nesneyi temsil eden başka bir String de döndürülebilir.

Object Sınıfı: toString metodu

Örneğin inheritanceExamples paketinin içindeki package1 paketinin içindeki Person sınıfının bir nesnesini aşağıdaki koddaki gibi yazdırırsak:

```
public static void main(String[] args) {  
    Person p = new Person("Ahmet", "Demir");  
    System.out.println(p);  
}
```

ekrana şu yazılır:

```
inheritanceExamples.package1.Person@5fe5c6f
```

Object Sınıfı: toString metodu

Ancak Person sınıfının içinde metodu aşağıdaki gibi override edersem

```
public String toString() {  
    return name + " " + surname;  
}
```

Aşağıdaki kodun sonucu olarak

```
public static void main(String[] args) {  
    Person p = new Person("Ahmet", "Demir");  
    System.out.println(p);  
}
```

ekrana şu yazılır:

```
Ahmet Demir
```