Örnek-2

```
class Arac {
    private String marka;
    private String model;
    private int uretimYili;

    public Arac(String marka, String model, int uretimYili) {
        this.marka = marka;
        this.model = model;
        this.uretimYili = uretimYili;
    }

    public String getMarka() {
        return marka;
    }

    public String getModel() {
        return model;
    }

    public int getUretimYili() {
        return uretimYili;
    }
    public void bilgileriGoster() {
        System.out.println("Marka: " + marka);
        System.out.println("Model: " + model);
        System.out.println("Üretim Yılı: " + uretimYili);
    }
}
```

```
class Otomobil extends Arac {
    private int motorHacmi;

    public Otomobil(String marka, String model, int uretimYili, int
motorHacmi) {
        super(marka, model, uretimYili);
        this.motorHacmi = motorHacmi;
    }

    public int getMotorHacmi() {
        return motorHacmi;
    }

    @Override
    public void bilgileriGoster() {
        super.bilgileriGoster();
        System.out.println("Motor Hacmi: " + motorHacmi + " cc");
    }
}
```

Java'da instanceof un gerçek kullanımını anlamak

```
interface Printable{}
class A implements Printable{
public void a(){System.out.println("a method");}
}
class B implements Printable{
public void b(){System.out.println("b method");}
}
class Call{
void invoke(Printable p){//upcasting
if(p instanceof A){
A a=(A)p;//Downcasting
a.a();
}
if(p instanceof B){
B b=(B)p;//Downcasting
b.b();
}

}
}//end of Call class
```

```
class Test4{
public static void main(String args[]){
Printable p=new B();
Call c=new Call();
c.invoke(p);
}
}
```

Output: b method

Arayüzde Java 8 Varsayılan Yöntemi

```
interface Drawable{
void draw();
default void msg(){
System.out.println("default method");
}
}
class Rectangle implements Drawable{
public void draw(){
System.out.println("drawing rectangle");
}
}
class TestInterfaceDefault{
public static void main(String args[]){
Drawable d=new Rectangle();
d.draw();
d.msg();
}}
```

- Java 8'den beri arayüzde yöntem gövdesine sahip olabiliyoruz. Ancak bunu varsayılan yöntem yapmamız gerekiyor. Bir örnek görelim:

drawing rectangle
default method

```
// Level 1
interface Bank {
        void deposit();
        void withdraw();
        void loan();
        void account();
}

// Level 2
abstract class Dev1 implements Bank {
        public void deposit()
        {
                System.out.println("Your deposit Amount :" + 100);
        }
}
abstract class Dev2 extends Dev1 {
        public void withdraw()
        {
                System.out.println("Your withdraw Amount :" + 50);
        }
}
```

```
// Level 3
class Dev3 extends Dev2 {
        public void loan() {}
        public void account() {}
}
// Level 4
class GFG {
        public static void main(String[] args)
        {
                Dev3 d = new Dev3();
                d.account();
                d.loan();
                d.deposit();
                d.withdraw();
        }
}
```

```
Your deposit Amount :100
Your withdraw Amount :50
```

Örnek-1

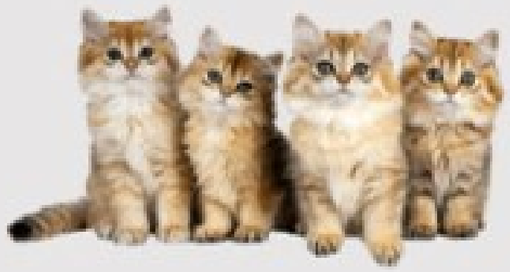
```
public class Car {
    protected int numberOfWheels = 4;
    protected int numberOfSeats = 4;

    protected int length = 10;
    protected int height = 4;

    protected int enginePower = 500;

    public void start() {
    }

    public void stop() {
    }

    public void gear() {
    }

    public void turn() {
    }

    public void brake() {
    }
}
```

Car -> Truck -> FireTruck

```
public Truck extends Car {
    public Truck() {
        numberOfWheels = 8;
        numberOfSeats = 2;
        enginePower = 1500;
        length = 20;
        height = 8;
    }
}
```

```
class FireTruck extends Truck {
    public FireTruck() {
        length = 28;
        height = 12;
        enginePower = 2000;
    }

    public void blowWater() {
    }
}
```

Örnek-2



```
class Arac {
    private String marka;
    private String model;
    private int uretimYili;

    public Arac(String marka, String model, int uretimYili) {
        this.marka = marka;
        this.model = model;
        this.uretimYili = uretimYili;
    }

    public String getMarka() {
        return marka;
    }

    public String getModel() {
        return model;
    }

    public int getUretimYili() {
        return uretimYili;
    }
    public void bilgileriGoster() {
        System.out.println("Marka: " + marka);
        System.out.println("Model: " + model);
        System.out.println("Üretim Yılı: " + uretimYili);
    }
}
```

```
class Otomobil extends Arac {
    private int motorHacmi;

    public Otomobil(String marka, String model, int uretimYili, int
motorHacmi) {
        super(marka, model, uretimYili);
        this.motorHacmi = motorHacmi;
    }

    public int getMotorHacmi() {
        return motorHacmi;
    }

    @Override
    public void bilgileriGoster() {
        super.bilgileriGoster();
        System.out.println("Motor Hacmi: " + motorHacmi + " cc");
    }
}
```

Örnek-3

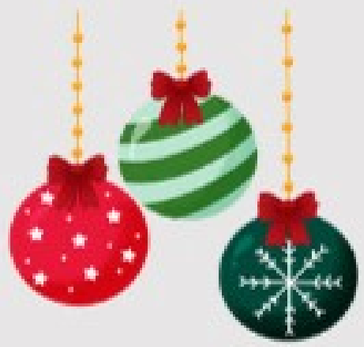
```
class Çalışan {
    private String ad;
    private String soyad;
    private int maas;

    public Çalışan(String ad, String soyad, int maas) {
        this.ad = ad;
        this.soyad = soyad;
        this.maas = maas;
    }
    public String getAd() {
        return ad;
    }

    public String getSoyad() {
        return soyad;
    }
    public int getMaas() {
        return maas;
    }
    public void bilgileriGoster() {
        System.out.println("Ad: " + ad);
        System.out.println("Soyad: " + soyad);
        System.out.println("Maaş: " + maas);
    }
}
```

```
class Müdür extends Çalışan {
    private String bölüm;

    public Müdür(String ad, String soyad, int maas, String bölüm)
{
        super(ad, soyad, maas);
        this.bölüm = bölüm;
    }

    public String getBölüm() {
        return bölüm;
    }

    @Override
    public void bilgileriGoster() {
        super.bilgileriGoster();
        System.out.println("Bölüm: " + bölüm);
    }
}
```


```
class İşçi extends Çalışan {
    private int çalışmaSaatleri;

    public İşçi(String ad, String soyad, int maas, int
çalışmaSaatleri) {
        super(ad, soyad, maas);
        this.çalışmaSaatleri = çalışmaSaatleri;
    }

    public int getÇalışmaSaatleri() {
        return çalışmaSaatleri;
    }

    @Override
    public void bilgileriGoster() {
        super.bilgileriGoster();
        System.out.println("Çalışma Saatleri: " + çalışmaSaatleri);
    }
}
```

```
public class KalitimOrnegi {
    public static void main(String[] args) {
        Müdür müdür = new Müdür("Ahmet", "Yılmaz", 10000
"Pazarlama");
        İşçi işçi = new İşçi("Ayşe", "Demir", 3000, 40);

        System.out.println("Müdür Bilgileri:");
        müdür.bilgileriGoster();

        System.out.println("\nİşçi Bilgileri:");
        işçi.bilgileriGoster();
    }
}
```

```
// Level 1
interface Bank {
        void deposit();
        void withdraw();
        void loan();
        void account();
}

// Level 2
abstract class Dev1 implements Bank {
        public void deposit()
        {
                System.out.println("Your deposit Amount :" + 100);
        }
}
abstract class Dev2 extends Dev1 {
        public void withdraw()
        {
                System.out.println("Your withdraw Amount :" + 50);
        }
}
```



```
// Level 3
class Dev3 extends Dev2 {
        public void loan() {}
        public void account() {}
}
// Level 4
class GFG {
        public static void main(String[] args)
        {
                Dev3 d = new Dev3();
                d.account();
                d.loan();
                d.deposit();
                d.withdraw();
        }
}
```

```
Your deposit Amount :100
Your withdraw Amount :50
```

Arayüzde Java 8 Varsayılan Yöntemi

```
interface Drawable{
void draw();
default void msg(){
System.out.println("default method");
}
}
class Rectangle implements Drawable{
public void draw(){
System.out.println("drawing rectangle");
}
}
class TestInterfaceDefault{
public static void main(String args[]){
Drawable d=new Rectangle();
d.draw();
d.msg();
}}
```

- Java 8'den beri arayüzde yöntem gövdesine sahip olabiliyoruz. Ancak bunu varsayılan yöntem yapmamız gerekiyor. Bir örnek görelim:

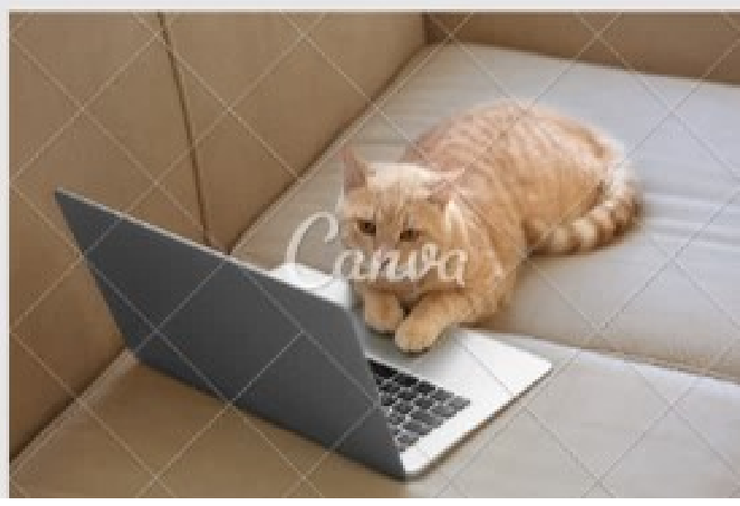
drawing rectangle
default method

Java'da instanceof un gerçek kullanımını anlamak

```
interface Printable{}
class A implements Printable{
public void a(){System.out.println("a method");}
}
class B implements Printable{
public void b(){System.out.println("b method");}
}
class Call{
void invoke(Printable p){//upcasting
if(p instanceof A){
A a=(A)p;//Downcasting
a.a();
}
if(p instanceof B){
B b=(B)p;//Downcasting
b.b();
}

}
}//end of Call class
```

```
class Test4{
public static void main(String args[]){
Printable p=new B();
Call c=new Call();
c.invoke(p);
}
}
```

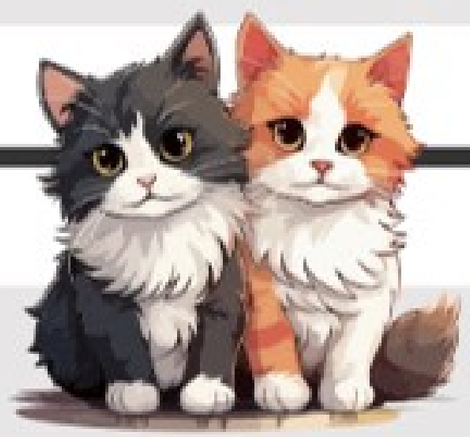
Output: b method



Örnek-3

```
class Çalışan {
    private String ad;
    private String soyad;
    private int maas;

    public Çalışan(String ad, String soyad, int maas) {
        this.ad = ad;
        this.soyad = soyad;
        this.maas = maas;
    }
    public String getAd() {
        return ad;
    }

    public String getSoyad() {
        return soyad;
    }
    public int getMaas() {
        return maas;
    }
    public void bilgileriGoster() {
        System.out.println("Ad: " + ad);
        System.out.println("Soyad: " + soyad);
        System.out.println("Maaş: " + maas);
    }
}
```

```
class Müdür extends Çalışan {
    private String bölüm;

    public Müdür(String ad, String soyad, int maas, String bölüm)
{
        super(ad, soyad, maas);
        this.bölüm = bölüm;
    }

    public String getBölüm() {
        return bölüm;
    }

    @Override
    public void bilgileriGoster() {
        super.bilgileriGoster();
        System.out.println("Bölüm: " + bölüm);
    }
}
```

```
class İşçi extends Çalışan {
    private int çalışmaSaatleri;

    public İşçi(String ad, String soyad, int maas, int
çalışmaSaatleri) {
        super(ad, soyad, maas);
        this.çalışmaSaatleri = çalışmaSaatleri;
    }

    public int getÇalışmaSaatleri() {
        return çalışmaSaatleri;
    }

    @Override
    public void bilgileriGoster() {
        super.bilgileriGoster();
        System.out.println("Çalışma Saatleri: " + çalışmaSaatleri);
    }
}
```

```
public class KalitimOrnegi {
    public static void main(String[] args) {
        Müdür müdür = new Müdür("Ahmet", "Yılmaz", 10000
"Pazarlama");
        İşçi işçi = new İşçi("Ayşe", "Demir", 3000, 40);

        System.out.println("Müdür Bilgileri:");
        müdür.bilgileriGoster();

        System.out.println("\nİşçi Bilgileri:");
        işçi.bilgileriGoster();
    }
}
```