

ÇOK BİÇİMLİLİK POLYMORPHISM

Computer Engineering Department
Java Course

Prof. Dr. Ahmet Sayar
Kocaeli University - Fall 2021

Çok-Biçimlilik (Polymorphism)

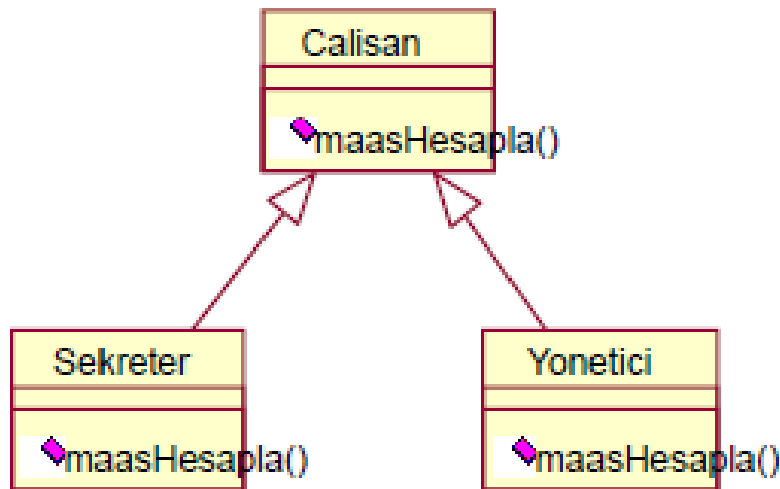
- Dinamik bağlama refer eder.
- Kalıtımda ezilen method mu, yoksa ana sınıfdaki metod mu çağrılacak çalışma anında belirlenir.
- Çok biçimlilik, kapsülleme ve kalıtım nesneye yönelik programlamanın en temel özellikleridir.

Çok-Biçimlilik (Polymorphism)

- Bir nesnenin davranış şekillerinin duruma göre değişebilmesidir.
- Eğer birdençok sınıfın ortak kullanacağı bir metod varsa, bu herbirinin temel alacağı bir anasınıf içerisinde tanımlanabilir.
 - Davranış şekillerindeki farklılıklar her sınıfın kendi yapısı içinde ifade edilir.
 - Örneğin bir selam() metodu ekrana, Turkler icin “selam” İngilizler için “hello” Almanlar için “hallo” yazdıracak biçimde çeşitlendirilebilir.

Çok-Biçimlilik (Polymorphism)

- Bir kalıtım ağacına ait sınıflarda aynı imza (dönüş tipi, ad, parametreler) ile tanımlanmış bir yöntem var ise; Java ortamı çalıştırma zamanında yöntemin hangi sınıfa ait tanımdan çalıştıracağını dinamik olarak belirleyebilir. Bu özelliğe çok-biçimlilik (“polymorphism”) denir.



- Bu özellik, “if” veya “switch” kullanımına gerek bırakmaz.
- Yeni bir işçi alt sınıfı eklendiğinde mevcut kodun değiştirilmesi gerekmez.

Örnek-1

```
interface Konus {  
    String getAd();  
    String merhaba ();  
}  
  
abstract class Insan implements Konus {  
    private final String ad;  
    protected Insan (String pAd) {  
        this.ad = pAd;  
    }  
    public String getAd() {  
        return this.ad;  
    }  
}
```

```
class Turk extends Insan {  
    public Turk (String pAd) {  
        super(pAd);  
    }  
    public String merhaba () {  
        return "Merhaba!";  
    }  
}  
  
class Ingiliz extends Insan {  
    public Ingiliz (String pAd) {  
        super(pAd);  
    }  
    public String merhaba () {  
        return "Hello!";  
    }  
}
```

Örnek-1

```
public class Test {  
    public static void main(String[] args) {  
        İnsan[] insanlar = { new Turk("Ahmet"),  
                               new Ingiliz ("Marry"),  
                               new Turk ("Ayşe")  
                               };  
        for (İnsan n : insanlar) {  
            System.out.println(n.getAd() + ": " + n.merhaba());  
        }  
    }  
}
```

```

• interface Konus {
•     String getAd();
•     String merhaba();
• }

• abstract class Insan implements Konus {
•     private final String ad;
•     protected Insan(String pAd) {
•         this.ad = pAd;
•     }
•     public String getAd() {
•         return this.ad;
•     }
• }

• class Turk extends Insan {
•     public Turk(String pAd) {
•         super(pAd);
•     }
•     public String merhaba() {
•         return "Merhaba!";
•     }
• }

• class Ingiliz extends Insan {
•     public Ingiliz(String pAd) {
•         super(pAd);
•     }
•     public String merhaba() {
•         return "Hello!";
•     }
• }

• public class Main {
•     public static void main(String[] args) {
•         Insan[] insanlar = {
•             new Turk("Ahmet"),
•             new Ingiliz("Marry"),
•             new Turk("Ayse")};
•         for (int i = 0; i < insanlar.length; i++) {
•             System.out.println("****"+insanlar[i].getAd()+"": "
•                                     +insanlar[i].merhaba());
•         }
•     }
• }

```

Örnek-2

```
class AnaSinif
{
    public void Yaz()
    {
        System.out.println("Ana Sınıf");
    }
}

class Tureyen1 extends AnaSinif
{
    public void Yaz()
    {
        System.out.println("Tureyen1");
    }
}

class Tureyen2 extends AnaSinif
{
    public void Yaz()
    {
        System.out.println("Tureyen2");
    }
}

class Tureyen3 extends AnaSinif
{
    public void Yaz()
    {
        System.out.println("Tureyen3");
    }
}

public class Program
{
    public static void Yaz(AnaSinif t)
    {
        t.Yaz();
    }
    public static void main(String[] args)
    {
        Tureyen1 t1=new Tureyen1();
        Tureyen2 t2=new Tureyen2();
        Tureyen3 t3=new Tureyen3();
        Yaz(t1);
        Yaz(t2);
        Yaz(t3);
    }
}
```



Örnek Polymorphism Çalışması

-3-

- abstract public class Calisan {
- private String ad;
- public Calisan(String ad) {
- setAd(ad);
- }
- public String getAd() {
- return new String(ad);
- }
- private void setAd(String ad) {
- this.ad = new String(ad);
- }
- abstract public double ode();
- public String yaz() {
- return "ad: " + ad;
- }
- }

- public class Maasli extends Calisan {
- double maas;
- public Maasli(String ad, double maas) {
- super(ad);
- setMaas(maas);
- }
- public void setMaas(double maas) {
- this.maas = maas;
- }
- public double getMaas() {
- return maas;
- }
- public double ode() {
- return maas;
- }
- public String yaz() {
- return super.yaz() + " (maas: " + maas + ")";
- }
- }

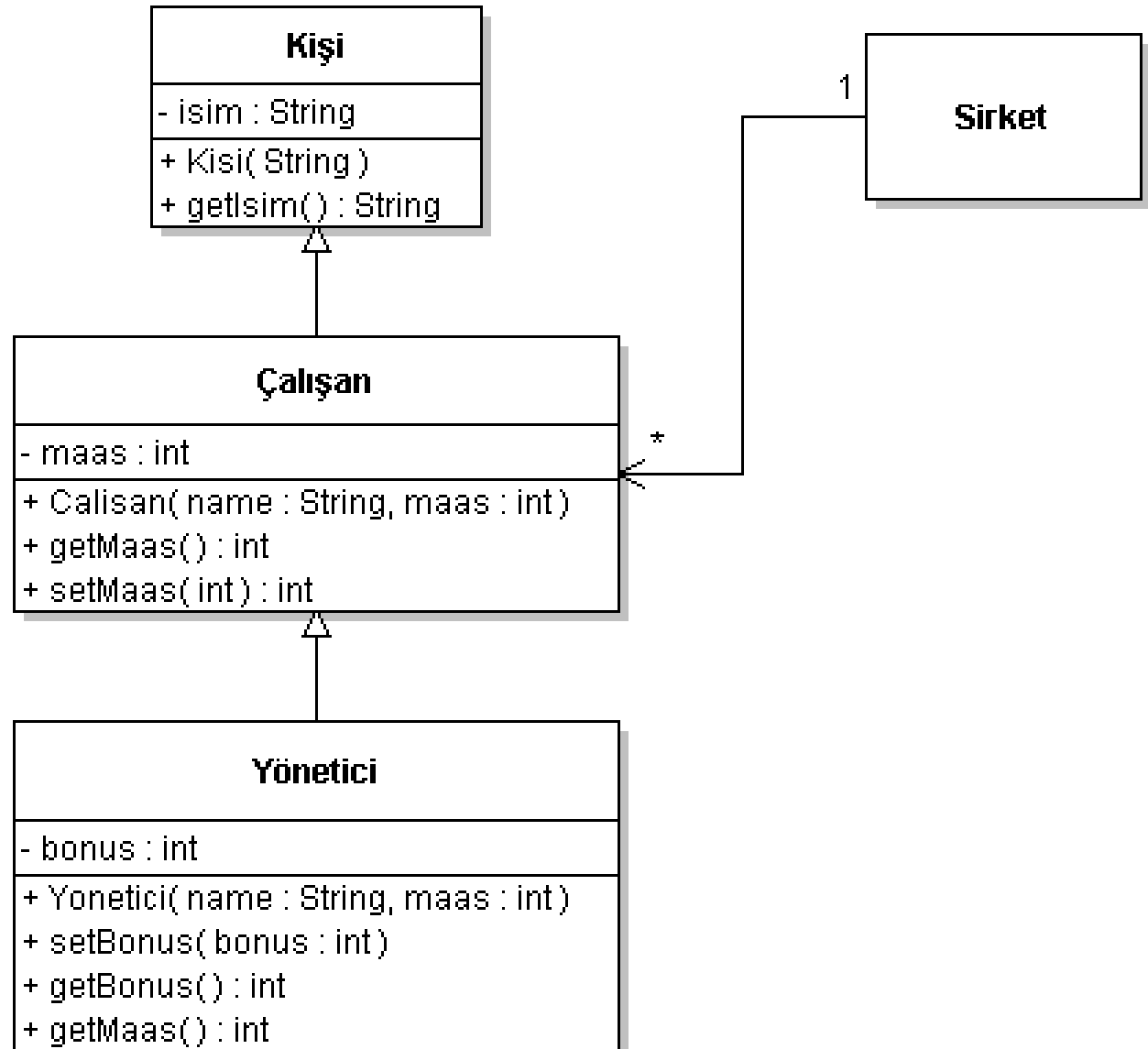
- public class Saatli extends Calisan {
- private double oran;
- private double saat;
- public Saatli(String ad, double oran, double saat) {
- super(ad);
- setOran(oran);
- setSaat(saat);
- }
- public void setOran(double oran) {
- this.oran = oran;
- }
- public void setSaat(double saat) {
- this.saat = saat;
- }
- public double getOran() { return oran; }
- public double getSaat() { return saat; }
- public double ode() { return oran * saat; }
- public String yaz() {
- return super.yaz() + " (oran: " + oran + ", saat: " + saat + ')';
- }
- }

- public class Main {
- public static final int MAX_CALISAN = 10;
-
- public static void main(String[] args) {
- Calisan[] calisanlar = new Calisan[MAX_CALISAN];
- int csayi = 0;
- calisanlar[csayi++] = new Saatli("Ayse Durmuş", 75.00, 2.5);
- calisanlar[csayi++] = new Maasli("Mehmet Yalçın", 125.00);
- calisanlar[csayi++] = new Saatli("Veysel Doğru", 85.00, 3.0);
- calisanlar[csayi++] = new Maasli("Zehra Sümer", 150.00);
- calisanlar[csayi++] = new Saatli("Ahmet Kara", 65.00, 2.0);
-
- for (int i = 0; i < csayi; ++i) {
- System.out.println("calisan: " + calisanlar[i].yaz());
- System.out.println("ode: " + calisanlar[i].ode());
- System.out.println();
- }
- }
- }

Lab Çalışması

Örnek-4: Kalıtım ve Çok Biçimlilik

- Örnek kalıtım ağacı:
 - Kişi
 - Çalışan
 - Yönetici.
- ve bu sınıfları kullanan bir Şirket sınıfı



Örnek-4:

Kişi sınıfı:

```
package cokbicim3;

public class Kisi {
    private String isim;

    public Kisi( String name ) {
        this.isim = name;
    }

    public String getIsim( ) {
        return isim;
    }
}
```

Örnek-4:

Çalışan sınıfı:

```
package cokbicim3;
public class Calisan extends Kisi {
    private int maas;

    public Calisan( String name, int maas ) {
        super( name );
        this.maas = maas;
    }
    public int getMaas( ) {
        return maas;
    }
    public void setMaas( int salary ) {
        this.maas = salary;
    }
}
```

- Bir çalışan nesnesinin ismini nasıl belirleyeceğiz?
 - İsimsiz kişi olmaz. Kişinin isim üyesi private. setIsim metodu da yok.
- Çözüm: Üst sınıfın yapılandırıcısına erişmek.
 - Bunun için super anahtar kelimesi kullanılır.

Örnek-4:

- Benzer şekilde, yöneticinin maaşının doğru hesaplanması için tekrar super kullanarak, bu kez üst sınıfın normal bir üye metodunu çağırdık.


```
package cokbicim3;
public class Yonetici extends Calisan {
    private int bonus;

    public Yonetici( String name, int maas ) {
        super( name, maas );
        bonus = 0;
    }
    public void setBonus( int bonus ) {
        this.bonus = bonus;
    }
    public int getBonus() {
        return bonus;
    }
    public int getMaas( ) {
        return super.getMaas( ) + bonus;
    }
}
```

Örnek-4:

Şirket sınıfı:

```
package cokbicim3;
public class Sirket {
    private Calisan[] calisanlar;
    public Sirket() {
        calisanlar = new Calisan[3];
        Yonetici mudur = new Yonetici( "Oktay Orcun", 8000 );
        mudur.setBonus( 1500 );
        calisanlar[0] = mudur;
        calisanlar[1] = new Calisan( "Ali Ucar", 7500 );
        calisanlar[2] = new Calisan( "Veli Kacar", 6000 );
    }
    public void calisanlariGoster( ) {
        for( Calisan calisan : calisanlar )
            if( calisan != null )
                System.out.println( calisan.getIsim() + " " + calisan.getMaas( ) );
    }
    public static void main(String[] args) {
        Sirket sirket = new Sirket( );
        sirket.calisanlariGoster( );
    }
}
```



Çokbiçimlilik örneği

- Yöneticilere de Çalışan gibi erişilebilmesi, çokbiçimlilik örneğidir.

OBJECT SINIFI

- java.lang.Object sınıfı, tüm sınıfların üst sınıfıdır.
 - Siz isterseniz de, istemeseniz de. Yazsanız da, yazmasanız da.
- toString() : String metodunu yeniden tanımlayarak, nesneleri komut satırına daha kolay yazdırabilirsiniz.
- Örnek:

```
package cokbicim3;
public class Calisan extends Kisi {
    //önceki koda ek olarak:
    public String toString( ) {
        return getIsim() + " " + getMaas( ) ;
    }
}

public class Sirket {
    //önceki kodda değişen kısım:
    public void calisanlariGoster( ) {
        for( Calisan calisan : calisanlar )
            if( calisan != null )
                System.out.println( calisan );
    }
}
```