Bilgisayar Teknolojileri Bölümü
Bilgisayar Programcılığı Programı

Öğr. Gör. Cansu AYVAZ GÜVEN

NESNE TABANLI PROGRAMLAMA

Java ile Nesne Tabanlı Programlamaya Giriş

Metot-Sınıf-Nesne

METOTLAR

- Metotlar, bir programın ayrılmış küçük parçacıkları olarak adlandırılır.
- Yapılacak işlemler metotlar ile ayrı bir yerde yapılabilir.
- Yapılacak işlemlerden herhangi bir değer dönebilir veya doğrudan işlemler yapılıp bitirilebilir.

```
ErişimBelirleyici    DönüşTipi MetotAdı    (MetotParametreleri)
 {
            MetotİçeriĞi
 }
```

METOTLAR

- **Erişim Belirleyici:** Metoda nasıl erişileceğini belirtir. Yazmak zorunlu değildir.

- **Dönüş Tipi:** Metotdan dönecek olan değerin tipidir. Bu *int, String* gibi tipler olabilir. Metot eğer geriye değer döndürmüyorsa, *void* olarak tanımlanmalıdır.

- **Metot Adı:** Metoda verilecek olan isimdir.

- **Metot Parametreleri:** Metodun yapacağı işler burada yer alır.

Parametresiz Metotlar

```
public class Metotlar {

    public static void main(String[] args) {
        carp();
    }
    static void carp()
    {
         System.out.println(3*5);
    }
}
```

Parametre almayan, fakat geriye değer döndüren metot örneği:

```
public class Metotlar2 {

    public static void main(String[] args) {
        String isim=yaz();
        System.out.println(isim);
    }
    static String yaz(){
    return "Mehmet";
    }
}
```

Parametreli Metotlar

```
public class Metotlar3 {

    public static void main(String[] args) {
        hesapla(5,7); //metoda parametre gönderildi
    }
    static void hesapla(int a, int b)//metot parametre aldı
    {
        System.out.println(a*b); //parametreler çarpılıp ekrana yazıldı
    }
    }
```

Parametre alarak değer döndüren metot örneği:

```
public class Metotlar4 {
    static int buyuksayi;
    public static void main(String[] args)
    {
        int x = hesapla(7,2);
        System.out.println("Büyük olan sayı: "+x);
    }
    static int hesapla (int a, int b)
    {
    if(a>b)
        buyuksayi=a;
    else if(a<b)
        buyuksayi=b;
    return buyuksayi;
    }
}
```

Scanner sınıfı kullanarak metodlara parametre göndermek

```
import java.util.Scanner;
public class Metotlar5 {
    public static void main(String[] args){
    Scanner s = new Scanner(System.in);
        System.out.println("Bir sayı giriniz");
        int deger = s.nextInt();
        long sonuc = hesapla(deger);
        System.out.println("Çarpım değeri: "+sonuc);
    }
    static long hesapla(int sayi)
    {
    int carpim = 1;
    for(int i = 1; i<=sayi; i++)
    {
    carpim*=i;
    }
    return carpim;
    }
```

Static Metotlar

- Static metotların iki kullanım amacı vardır.

1. her nesne için aynı işi yapan static bir metot tanımlanır ve bütün nesneler için ayrı ayrı oluşturulmaz. Bellekten kazanç sağlanır.

2. nesne oluşturmadan sınıf içerisindeki metotlara erişmektedir.

Nesne oluşturmadan sınıf içindeki metotlara erişmek

```
public class StaticMetotlar {
    public static int metot (int a, int b)
    {
    return a*b;
    }
    public static void main(String[] args){
        System.out.println(StaticMetotlar.metot(3,5));
    }
    }
```

Nesne oluşturmadan sınıf içindeki metotlara erişmek

```
package metotlar3;

public class StaticMetotlar3 {
        public static void main(String[] args)
        {
        int a=3, b=4;
            System.out.println((int)Math.pow(a,b));
        }
}
```

Her nesne için aynı işi yapan static bir metot tanımlamak

```
public class StaticMetotlar2 {
    public static int x;
    public static void metod(int a)
    {
        x=a;
    }
       public static void main(String[] args){
           StaticMetotlar2 s1 = new StaticMetotlar2();
           StaticMetotlar2 s2 = new StaticMetotlar2();
           s1.metod(5);
           s2.metod(9);
           System.out.println(s1.x);
           System.out.println(s2.x);
}
}
```

SINIFLAR

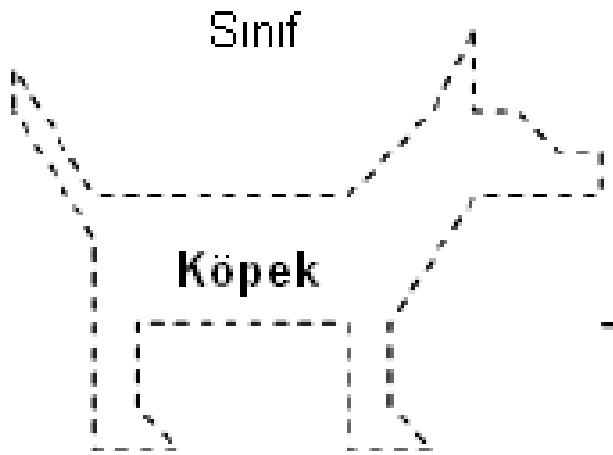
- Nesneler çevremizde bulunan her şeydir. Araba, uçak, kitap, personel, müdür, fatura, öğrenci gibi. Programcılar her zaman gerçek yaşamdaki durumlara yakın senaryolar oluşturmaya çalışırlar. Bu yöndeki ilk adım bilgisayarın yaşadığımız dünyadan nesnelerle ilişki kurmasını sağlamaktır. Hepimizin bildiği gibi, bilgisayar sadece bir elektronik makinedir. Bilgisayarın bizim bildiğimiz nesneleri tanımasını sağlayacak bilgiyi vermek, bizim sorumluluğumuzdadır. İşte bu noktada nesneye dayalı modelleme tekniği devreye girer. Bu modelde gerçek problemlerde karşılaştığımız nesneleri, bilgisayarda benzer nesneler olarak modelleyebiliriz.

SINIFLAR

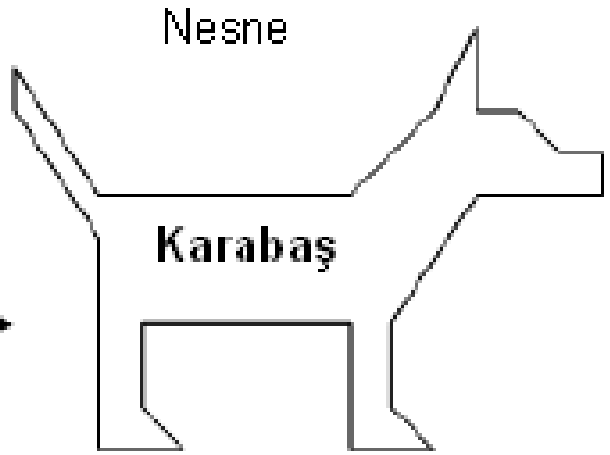
- Nesneye dayalı programlamanın esasını *sınıf* (class) oluşturur. Sınıf aynı cins nesnelerin genel tanımıdır. Örneğin kullandığımız araba bir nesnedir (Ford gibi). Aynı şekilde başkalarının da kullandığı Opel, BMW, Mercedes, Renault arabalarının her biri ayrı birer nesnedir. Bu nesnelerin hepsi araba sınıfı ile tanımlanabilir.

- Aynı sınıfa ait nesneler ortak özelliklere sahiptir.

SINIFLAR



Örnek Oluşturma



Özellikler	Metotlar
Renk	Oturmak
Göz Rengi	Koşmak
Yükseklik	Havlamak
Ağırlık	Birşey Getirmek

Özellikler	Metotlar
Siyah-Gri	Oturmak
Kahverençi	Koşmak
45 cm	Havlamak
25 kg	Birşey Getirmek

SINIFLAR

Nesneye dayalı programlama,

1. Nesnelerin ortak özelliklerinin sınıf (class) yapıları kullanılarak tanımlanması,
2. Bu sınıfları kullanarak nesnelerin oluşturulması,
3. Bu nesnelerle uygulamaların gerçekleştirilmesi

aşamalarından oluşur.

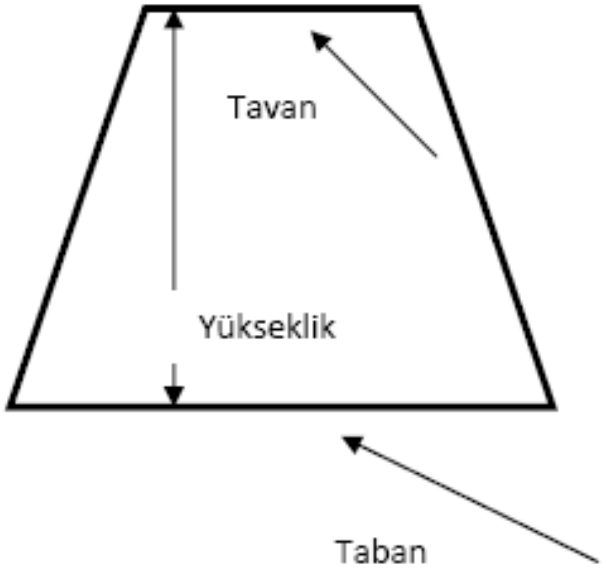
Sınıf yeni bir tip veri tanımlar. Bu yeni tip, bu tipte yeni nesneler oluşturmada kullanılır. Bu yüzden sınıf nesne için bir şablondur. Ve bir nesne sınıfın bir örneğidir.

Sınıfın Genel Biçimi : Bir sınıf, class anahtar sözcüğü ile tanımlanır. Sınıf içinde değişkenler ve metotlar yer alır.

Basit bir sınıf uygulaması

- Yamuk geometrik elemanının alanını hesaplayacak bir program geliştireceğiz.

- **Alan =** (taban+tavan)/2*yukseklik



```
package sinifnesneornek;
//Yamuk sınıfı
public class YamukAlan {
    double taban;
    double tavan;
    double yukseklik;

//Yamuk sınıfından nesne oluşturulur
public static void main (String args[]){
YamukAlan ymk1 = new YamukAlan();
double alan;

//ymk1 nesnesine değer atama
ymk1.taban=40;
ymk1.tavan=20;
ymk1.yukseklik=10;

//yamuğun alanı hesaplanır
alan=(ymk1.taban + ymk1.tavan) /2* ymk1 .yukseklik;

    System.out.println("Yamuğun alanı: "+alan);
}
}
```

```
public class YamukAlan2 {
    double taban;
    double tavan;
    double yukseklik;

//Yamuk sınıfından 2 tane nesne oluşturulur
public static void main (String args[]){
YamukAlan2 ymk1 = new YamukAlan2();
YamukAlan2 ymk2 = new YamukAlan2();
double alan;

ymk1.taban=40;
ymk1.tavan=20;
ymk1.yukseklik=10;

ymk2.taban=25;
ymk2.tavan=10;
ymk2.yukseklik=12;

//ymk1 yamuğunun alanı hesaplanır
alan=(ymk1.taban + ymk1.tavan) /2* ymk1. yukseklik;
    System.out.println("1. Yamuğun alanı: "+alan);
//ymk2 yamuğunun alanı hesaplanır
alan=(ymk2.taban + ymk2.tavan) /2* ymk2. yukseklik;
    System.out.println("2. Yamuğun alanı: "+alan);
```

```
public class YamukAlan3 {
    double taban;
    double tavan;
    double yukseklik;

//Sınıfa bir metot ekleme
 void alan(){
     double alanh =(taban+tavan)/2*yukseklik;
     System.out.println("Alan= "+alanh);

 }
public static void main (String args[]){
YamukAlan3 ymk1 = new YamukAlan3();
YamukAlan3 ymk2 = new YamukAlan3();

ymk1.taban=12;
ymk1.tavan=5;
ymk1.yukseklik=8;

ymk2.taban=8;
ymk2.tavan=5;
ymk2.yukseklik=7;

//ymk1 yamuğunun alanı yazdırılıyor
ymk1.alan();
//ymk2 yamuğunun alanı yazdırılıyor
ymk2.alan();
}
}
```

//Metotdan Değer Döndürmek

```
public class YamukAlan4 {
    double taban;
    double tavan;
    double yukseklik;
//Yamuğun alanını hesapla ve sonucu gönder
double alan(){
    return (taban+tavan)/2*yukseklik;
 }
public static void main (String args[]){
YamukAlan4 ymk1 = new YamukAlan4();
YamukAlan4 ymk2 = new YamukAlan4();

ymk1.taban=12;
ymk1.tavan=5;
ymk1.yukseklik=8;

ymk2.taban=8;
ymk2.tavan=5;
ymk2.yukseklik=7;

//ymk1 yamuğunun alanı yazdırılıyor
System.out.println("1.Yamuğun Alanı: "+ymk1.alan());
//ymk2 yamuğunun alanı yazdırılıyor
System.out.println("1.Yamuğun Alanı: "+ymk2.alan());
}
}
```

Kurucular (Constructors)

- Bir nesnenin oluşturulduğu anda otomatik olarak çalıştırılan metotlardır. Bir sınıf içinde birden fazla kurucu olabilir. Nesne oluşturulurken hangi kurucu çağrılacağı parametrelerle belirlenir.

```
public class YamukAlan5 {
    double taban;
    double tavan;
    double yukseklik;
   //Yamuk constructor
    YamukAlan5(){
        taban=6;
        tavan=4;
        yukseklik=5;
    }
//Yamuğun alanını hesapla ve sonucu gönder
double alan(){
     return (taban+tavan)/2*yukseklik;
 }
public static void main (String args[]){
YamukAlan5 ymk1 = new YamukAlan5();

//ymk1 yamuğunun alanı yazdırılıyor
System.out.println("Yamuğun Alanı: "+ymk1.alan());
}
}
```

```
public class YamukAlan6 {
double taban;
double tavan;
double yukseklik;
// Yamuk constructor
YamukAlan6() {
taban=6;
tavan=4;
yukseklik=7;
}
// parametreli constructor
YamukAlan6(double tab, double tav, double yuk) {
taban=tab;
tavan=tav;
yukseklik=yuk;
}
// yamuğun alanını hesapla ve sonucu gönder
double alan(){
return (taban*tavan)/2 * yukseklik;
}
// Yamuk sınıfından nesne oluşturacaktır
public static void main(String args[]) {
YamukAlan6 ymk1 = new YamukAlan6();
YamukAlan6 ymk2 = new YamukAlan6(8,3,6);

System.out.println("1. Yamuğun alanı = " + ymk1.alan());
System.out.println("2. Yamuğun alanı = " + ymk2.alan());
}
```

```
package sinifnesneornek;
// Aşırı yüklenem (overloading)
class AsiriYukleme {
// tamsayı toplama
void topla(int a, int b) {
System.out.println("toplam: " + (a+b) );
}
// reel sayı toplama
void topla(double a, double b) {
System.out.println("toplam: " + (a+b) );
}

void topla(int a , int b, int c) {
System.out.println("toplam: " + (a+b+c) );
}
public static void main(String args[]) {
AsiriYukleme t = new AsiriYukleme();
t.topla(3,5);
t.topla(23.45,56.71);
t.topla(5, 12, 23);
}
}
```