

JAVA PROGRAMLAMA

Metodlar ve Paketler

Sunum İçeriği

⚡ İÇİNDEKİLER

1. Metod Tanımlama ve Çağırma
2. Parametreler ve Return Değerleri
3. Method Overloading
4. Paket (Package) Kavramı
5. Import İfadesi
6. Erişim Belirleyiciler

1. METOD TANIMLAMA VE ÇAĞIRMA

Metod Nedir?

Metod, belirli bir görevi yerine getiren, yeniden kullanılabilir kod bloklarıdır. Metodlar, programın daha düzenli, okunabilir ve sürdürülebilir olmasını sağlar.

Kısa tanım olarak “Bir kere yaz, defalarca kullan” demektir.

Metodların Faydaları:

- Kod Tekrarını Önler: Aynı kodu defalarca yazmak yerine, bir kez yazıp birçok yerde kullanabilirsiniz.
- Okunabilirlik: Kodun daha anlaşılır ve organize olmasını sağlar.
- Bakım Kolaylığı: Bir değişiklik yapmak istediğinizde sadece metodu güncellemeniz yeterlidir.
- Modülerlik: Programı küçük parçalara böler ve her parçanın bağımsız çalışmasını sağlar.

Metodların Yapısı:

```
erişimBelirleyici geriDönüşTipi metodAdı() {  
    // yapılacak işlemler  
}
```

Metod Bileşenleri:

Bileşen	Açıklama
Erişim Belirleyici	Metodun görünürlük seviyesini belirler (public, private, protected, default)
Geri Dönüş Tipi	Metodun döndüreceği veri tipi (int, String, void vb.)
Metod Adı	Metodun benzersiz ismi (camelCase kullanılır)
Parametreler	Metoda gönderilen değerler (opsiyonel)

Parametresiz Metod Örneği

```
© Main.java © ParametresizMetodlar.java ×

1 package Metodlar;
2
3 public class ParametresizMetodlar { 4 usages
4
5
6                                     // TODO 1) Void Metod
7
8     public void selamla() { 1 usage
9
10
11         System.out.println("Arkadaşlar sunum Hoş Geldiniz!");
12
13         // todo void = geri değer döndürmez demektir.
14     }
15
16                                     // TODO 2) String Metod
17
18     public String mesajGetir() { // parametre yok 1 usage
19         return "Merhaba Dünya!";
20     }
21
22                                     // TODO 3) int metod
23
24     public int sayiGetir() { 1 usage
25
26         return 10;
27     }
28
29                                     // TODO static ile tanımlama
30
31     public static void bilgileriGoster() { 1 usage
32
33         System.out.println("Benim Adım Emirsah");
34     }
35 }
```

Parametresiz Metodları çağırma

The screenshot shows a Java code editor with two tabs: 'Main.java' and 'ParametresizMetodlar.java'. The 'Main.java' tab is active, displaying the following code:

```
9  public class Main {  
10     public static void main(String[] args) {  
11         // TODO PARAMETRESİZ METODLAR  
12         // todo metod static olduğu için nesne oluşturmadan direkt class ismi ile erişip metodunu çağırırdık  
13         ParametresizMetodlar.bilgileriGoster();  
14         // todo Nese oluşturma  
15         ParametresizMetodlar metodlar = new ParametresizMetodlar();  
16         // todo 1:  
17         metodlar.selamla();  
18         // todo 2:  
19         int sonuc = metodlar.sayiGetir();  
20         System.out.println("Gelen sayı: " + sonuc);  
21         // todo 3:  
22         String gelenMesaj=metodlar.mesajGetir();  
23         System.out.println(gelenMesaj);  
24     }  
25 }  
--
```

The code includes several TODO comments and placeholder text in Turkish, indicating steps for the developer to implement the functionality.

Parametreli Metod Örneği

The screenshot shows a Java code editor with two tabs: "Main.java" and "ParametreliMetod.java". The "ParametreliMetod.java" tab is active, displaying the following code:

```
1 package Metodlar;
2
3 public class ParametreliMetod { 3 usages
4     
5         public void selamla(String isim) { 1 usage
6             System.out.println("Merhaba " + isim);
7         }
8
9     }
10
11
12     public void kareAl(int sayi) { 1 usage
13         System.out.println("Karesi: " + (sayi * sayi));
14     }
15
16     public void topla(int a, int b) { 1 usage
17         System.out.println("Toplam: " + (a + b));
18     }
19
20     public int carp(int a, int b) { 1 usage
21         return a * b;
22     }
23
24
25 }
26
27 }
```

Red arrows point to the method names `selamla` and `topla`.

Parametreli Metodları çağrıma

```
9  public class Main {  
10     public static void main(String[] args) {  
11         // TODO PARAMETRELİ METODLAR  
12         ParametreliMetod metodParametreli = new ParametreliMetod();  
13         metodParametreli.selamla(isim: "Emirsah");  
14         metodParametreli.kareAl(sayı: 5);  
15         metodParametreli.topla(a: 3, b: 5);  
16         int CarpimSonucu= metodParametreli.carp(a: 4, b: 5);  
17         System.out.println("Carpim Sonucu: " + CarpimSonucu);  
18         // todo metod static olduğu için nesne oluşturmadan direkt class ismi ile erişip metodu çağırıldık  
19         Menu.calistir();  
20     }  
21 }
```

Return Değerleri

Return ifadesi, metodun işini bitirip bir değer döndürmesini sağlar. Geri dönüş tipi **void** olmayan her metod **return** ifadesi kullanmalıdır.

! ÖNEMLİ NOTLAR

- void tipi metod return kullanmaz (veya boş return; kullanır)
 - Return ifadesinden sonraki kodlar çalışmaz
 - Metodun geri dönüş tipi ile return edilen değerin tipi aynı olmalı

3. METHOD OVERLOADING (METOD AŞIRI YÜKLEME)

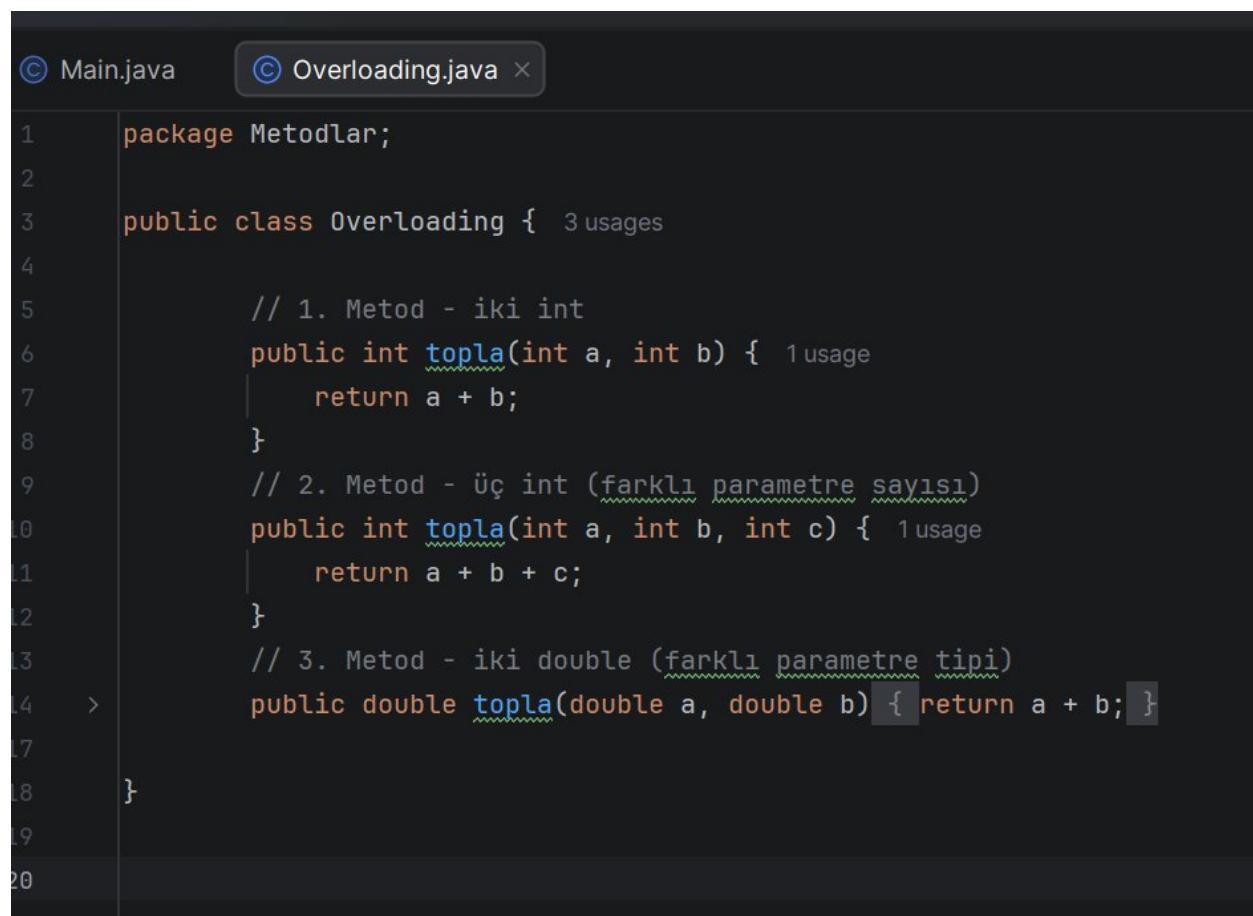
⌚ Overloading Nedir?

Method Overloading, **aynı isimde birden fazla metod** tanımlamaktır. Bu metodlar, **farklı parametre sayısı veya farklı parametre tiplerine sahip olmalıdır**.

✓ Overloading Kuralları:

- 1. Metod adları aynı olmalı
- 2. Parametre sayısı VEYA parametre tipleri farklı olmalı

Örnek



The screenshot shows a Java code editor with two tabs: 'Main.java' and 'Overloading.java'. The 'Overloading.java' tab is active, displaying the following code:

```
1 package Metodlar;
2
3 public class Overloading { 3 usages
4
5     // 1. Metod - iki int
6     public int topla(int a, int b) { 1 usage
7         return a + b;
8     }
9
10    // 2. Metod - Üç int (farklı parametre sayısı)
11    public int topla(int a, int b, int c) { 1 usage
12        return a + b + c;
13    }
14
15    // 3. Metod - iki double (farklı parametre tipi)
16    public double topla(double a, double b) { return a + b; }
17
18 }
19
20 }
```

The code defines a class named 'Overloading' with three methods, all named 'topla'. The first method takes two integer parameters and returns their sum. The second method takes three integer parameters and returns their sum. The third method takes two double parameters and returns their sum. The code editor highlights the word 'topla' in orange across all three definitions.

```
>Main.java x  Overloading.java
9     public class Main {
1         public static void main(String[] args) {
2
3
4
5
6
7
8
9
10    //*****
11
12    // TODO OVERLOADING METODLAR
13
14    /*
15     TANIM : Aynı isimde birden fazla metod tanımlamaktır. Farklı parametre sayısı veya farklı
16     parametre tiplerine sahip olmalıdır.
17     */
18
19
20
21    Overloading overloading = new Overloading();
22
23    int overloadingToplamMetod1= overloading.topla( a: 3, b: 2);
24    int overloadingToplamMetod2= overloading.topla( a: 3, b: 2, c: 1);
25
26    double overloadingToplamMetod3= overloading.topla( a: 10.2, b: 10.2);
27
28    System.out.println("overloadingToplam Metod 1 :" + overloadingToplamMetod1);
29    System.out.println("overloadingToplam Metod 2 :" + overloadingToplamMetod2);
30    System.out.println("overloadingToplam Metod 3 :" + overloadingToplamMetod3);
31
32
33 }
```

4. PAKET (PACKAGE) KAVRAMI

Paket Nedir?

Paketler, **ilgili sınıfları ve arayüzleri bir araya gruplandırmak** için kullanılan Java organizasyon yapısıdır. Dosya sistemindeki klasörlere benzer.

1. Paket (Package): Sınıfın Adresi

package ifadesi, yazdığınız Java dosyasının hangi klasör (kategori) altında olduğunu belirtir. Bilgisayarındaki bir dosya yolu gibidir.

Nerede Kullanılır? Java dosyasının en üst satırında (ilk satırda) kullanılır.

Görevi: Sınıfı bir grup içine dahil eder.

⌚ Paketlerin Faydaları:

- ✓ Organizasyon: İlgili sınıfları gruplar, projeyi düzenli tutar
- ✓ İsim Çakışmasını Önler: Farklı paketlerde aynı isimli sınıflar olabilir
- ✓ Erişim Kontrolü: Paket düzeyinde erişim sağlar
- ✓ Yeniden Kullanılabilirlik: Paketler kolayca diğer projelere aktarılabilir

🏷️ Paket İsimlendirme Kuralları:

- Tamamen küçük harf kullanılır
- Nokta (.) ile ayrılır

♣ Yaygın Java Paketleri

Paket Adı	Açıklama
java.lang	Temel sınıflar (String, Math, System) - Otomatik import
java.util	Koleksiyonlar, tarih/saat (ArrayList, Scanner, Date)
java.io	Dosya okuma/yazma işlemleri
java.awt	GUI bileşenleri (pencere, buton vb.)
java.net	Ağ programlama (URL, Socket)

5. IMPORT İFADESİ

Import Nedir?

Import ifadesi, **başka paketlerdeki sınıfları kullanmak** için kullanılır. Import kullanmadan da sınıfları kullanabilirsiniz, ancak her seferinde tam paket yolunu yazmanız gereklidir.

Import Türleri

Tek Sınıf Import:

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

public class Program {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        ArrayList<String> liste = new ArrayList<>();

    }
}
```

Tüm Paket Import (Wildcard):

```
import java.util.*;

// util paketindeki TÜM sınıfları import eder

public class Program {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        ArrayList<String> liste = new ArrayList<>();

        Date tarih = new Date();
    }
}
```

3 Static Import nedir ?

Class ismini yazmadan,
o classın static üyelerini direkt kullanmamızı sağlar.

```
import static java.lang.Math.*;  
  
public class Hesaplama {  
    public static void main(String[] args) {  
  
        // Math.PI yerine direkt PI yazabilirsiniz  
  
        double alan = PI * pow(5, 2);  
        // Math yazmaya gerek yok!  
        System.out.println(sqrt(16));  
        // Math.sqrt yerine sqrt    } }
```

Not : Aynı pakette bulunan sınıflar import edilmez.

6. ERİŞİM BELİRLEYİCİLER (ACCESS MODIFIERS)

💡 Erişim Belirleyiciler Nedir?

Erişim belirleyiciler, **sınıfların, metodların ve değişkenlerin görünürlük seviyesini** kontrol eder. Java'da 4 ana erişim belirleyici vardır.

III Erişim Seviyeleri Tablosu

Belirleyici	Aynı Sınıf	Aynı Paket	Alt Sınıf	Tüm Dünya
public	✓	✓	✓	✓
protected	✓	✓	✓	✗
default (yazılmaz)	✓	✓	✗	✗
private	✓	✗	✗	✗

1. **Public = Herkese Açık** Her yerden erişilebilir. En geniş erişim seviyesidir.

```
java

package com.sirket.proje;

public class Musteri {

    // public -> her yerden erişilebilir
    public String ad;

    // public -> her yerden çağrılabılır
    public void bilgiGoster() {
        System.out.println("Müşteri: " + ad);
    }
}
```

2. private - Özel

Sadece tanımlandığı sınıf içinden erişilebilir. En kısıtlı erişim seviyesidir.

```
public class BankaHesabi {

    // private -> sadece bu sınıfın içinden erişilebilir
    private double bakiye;

    // private -> sadece bu sınıfın içinden çağrılabılır
    private void gizliIslem() {
        // Hassas işlemler burada yapılır
    }

    public void paraCek(double miktar) {

        // private değişkene burada erişebiliriz
        if (miktar <= bakiye) {
            bakiye -= miktar;
        }
    }
}
```

3. protected - Korumalı

Aynı paket içinden ve alt sınıflardan erişilebilir.

```
java

public class Hayvan {

    // protected -> aynı paket + alt sınıflar erişebilir
    protected String isim;

    // protected -> aynı paket + alt sınıflar çağrılabılır
    protected void sesCikar() {
        System.out.println("Hayvan ses çıkarıyor");
    }
}

class Kedi extends Hayvan {

    public void miyavla() {
        isim = "Minnoş";    // protected değişkene erişim
        sesCikar();         // protected metoda erişim
    }
}
```

4. default (package-private) - Varsayılan

Erişim belirleyici yazılmadığında otomatik olarak uygulanır. Sadece aynı paket içinden erişilebilir.

```
package com.sirket.proje;

// default (package-private) erişim -> sadece aynı paketten erişilebilir
class YardimciSınıf {

    // default -> sadece bu paketten erişilebilir
    String mesaj;

    // default -> sadece bu paketten çağrılabılır
    void yaz() {
        System.out.println(mesaj);
    }
}
```

💡 EN İYİ UYGULAMALAR VE İPUÇLARI

⌚ Metodlar İçin:

- ✓ Metod isimlerini açıklayıcı ve anlamlı yapın
- ✓ Bir metod tek bir iş yapmalı
- ✓ Metod isimleri fiil ile başlamalı (hesapla, bul, getir, yaz)
- ✓ Parametre sayısını makul tutun (ideal 0-3 arası)

📦 Paketler İçin:

- ✓ İlgili sınıfları aynı pakete koyun
- ✓ Paket isimlerini tamamen küçük harf yapın
- ✓ Şirket domain adınızı ters çevirerek başlayın (com.sirketadi)
- ✓ Anlamlı paket hiyerarşisi oluşturun

Erişim Belirleyiciler İçin:

- ✓ Değişkenleri mümkün olduğunca private yapın
- ✓ Getter ve Setter metodları kullanın
- ✓ API sınıflarını public, yardımcı sınıfları default yapın
- ✓ En kısıtlayıcı erişim seviyesini kullanın (Encapsulation)

ÖZET

Bu sunumda öğrendiklerimiz:

- 1 Metodların nasıl tanımlanıp çağırılacağı
- 2 Parametreler ve return değerlerinin kullanımı
- 3 Method Overloading ile aynı isimde farklı metodlar oluşturma
- 4 Paketler ile kod organizasyonu
- 5 Import ifadesi ile diğer paketlerdeki sınıfları kullanma
- 6 Erişim belirleyiciler ile kod güvenliği sağlama

Bu konuları iyi anlamak, Java'da profesyonel ve sürdürülebilir kod yazmak için kritik öneme sahiptir!

Başarılılar Dilerim! 