JAVA PROGRAMLAMA

Metodlar ve Paketler

Sunum İçeriği

📋 İÇİNDEKİLER

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 metodAdi() {
    // yapılacak işlemler
}
```

📝 Metod Bileşenleri:

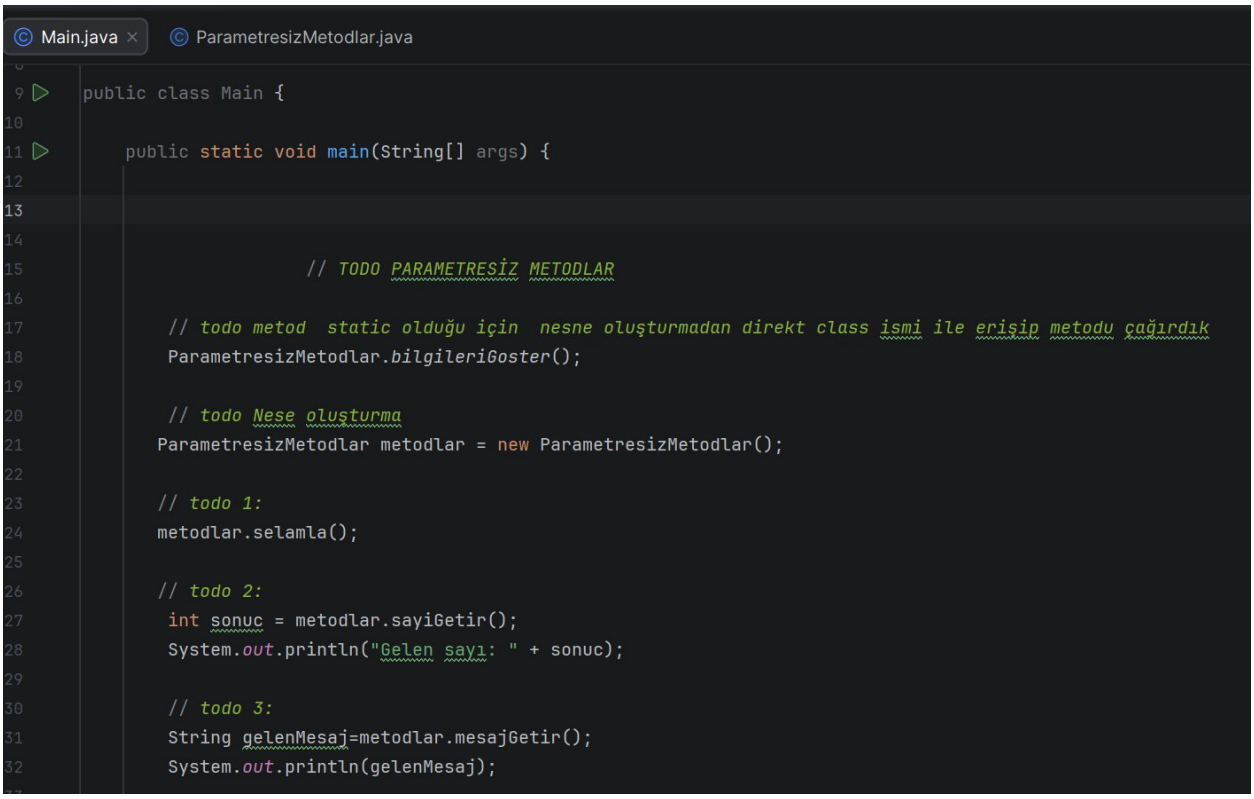
Bileşen	Açıklama
Erişim Belirleyici	Metodun görünürlük seviyesini belirler (public, private, protected, default)
Geri Dönüş Tipi	Metodun döndüreceği veri tipi (int, String, void vb.)
Metod Adı	Metodun benzersiz ismi (camelCase kullanılır)
Parametreler	Metoda gönderilen değerler (opsiyonel)

Parametresiz Metod Örneği

Main.java | ParametresizMetodlar.java

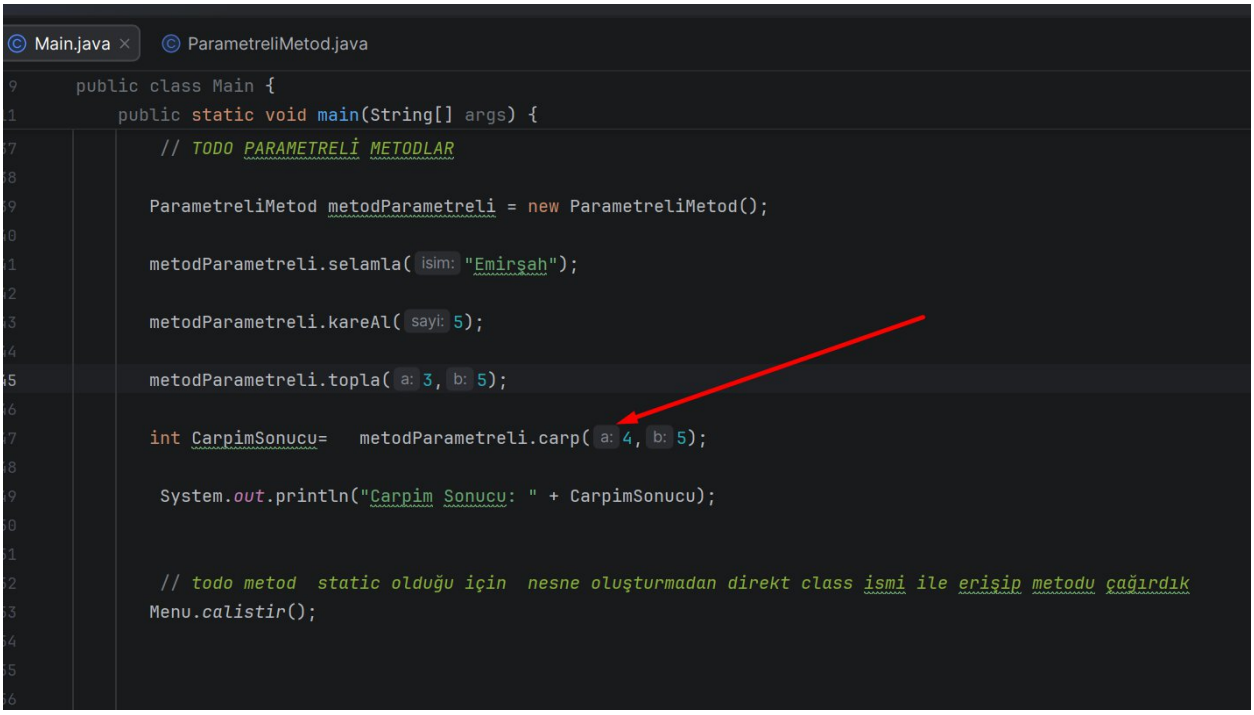
```java
package Metodlar;

public class ParametresizMetodlar {  4 usages


                                                    // TODO 1) Void Metod

public void selamla() {  1 usage


    System.out.println("Arkadaşlar sunuma Hoş Geldiniz!");

    // todo void = geri değer döndürmez demektir.
 }

                                                      // TODO 2) Strign Metod
  public String mesajGetir() {   // parametre yok  1 usage
        return "Merhaba Dünya!";
    }

                                                     // TODO 3) int metod
   public int sayiGetir() {  1 usage

        return 10;
    }

                                                     // TODO static ile tanımlama
   public static void bilgileriGoster() {  1 usage

        System.out.println("Benim Adım Emirşah");

    }

```

Parametresiz Metodları çağırma



Parametreli Metod Örneği

Parametreli Metodları çağırma



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.

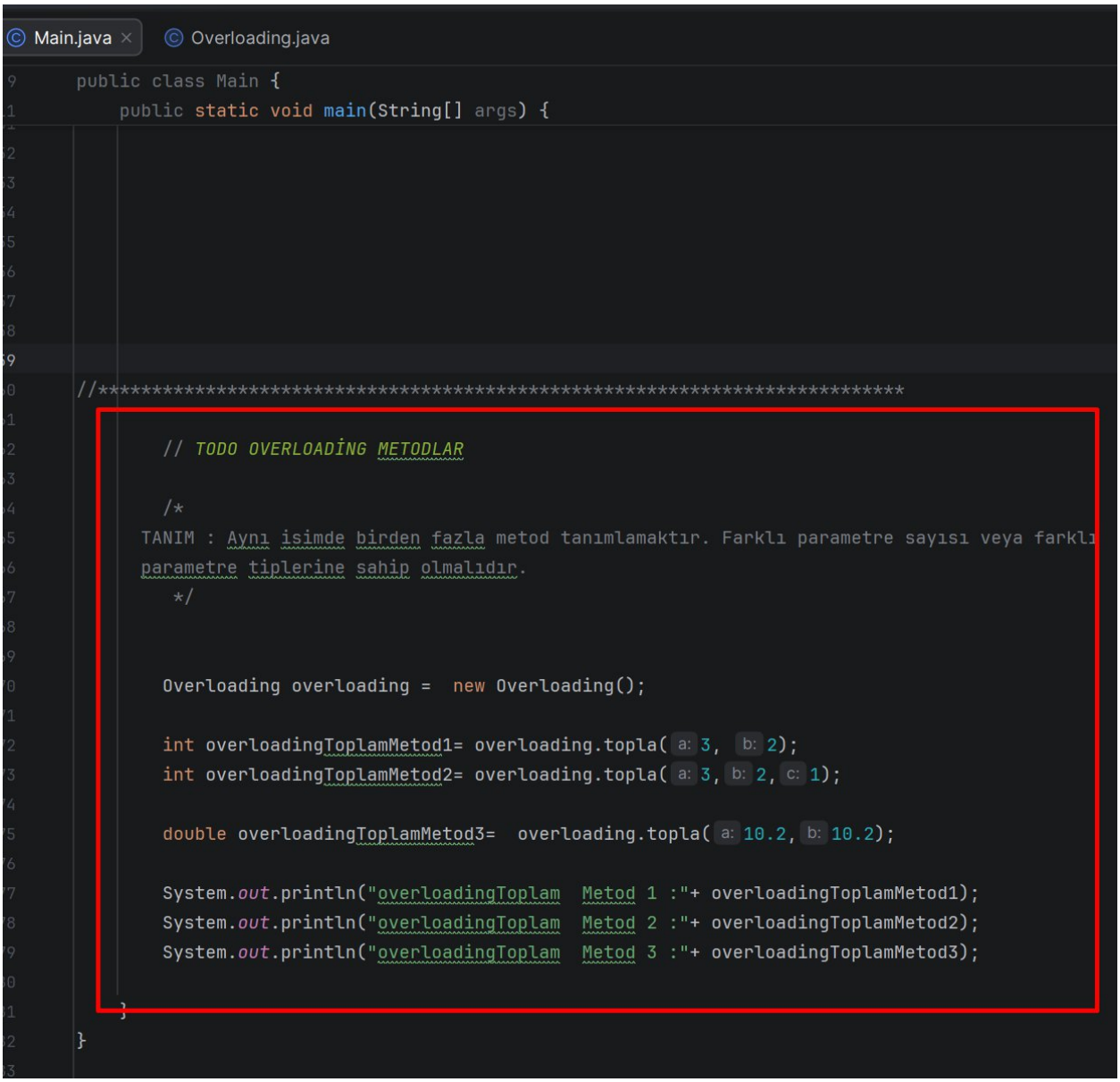
✔ Overloading Kuralları:

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

Örnek

Main.java | Overloading.java

```java
package Metodlar;

public class Overloading {  3 usages

        // 1. Metod - iki int
        public int topla(int a, int b) {  1 usage
            return a + b;
        }
        // 2. Metod - Üç int (farklı parametre sayısı)
        public int topla(int a, int b, int c) {  1 usage
            return a + b + c;
        }
        // 3. Metod - iki double (farklı parametre tipi)
        public double topla(double a, double b) { return a + b; }


}
```



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, yazdığın 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. İmport kullanmadan da sınıfları kullanabilirsiniz, ancak her seferinde tam paket yolunu yazmanız gerekir.

Import Türleri

1 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<>();

    } }
```

2 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.

📊 Erişim Seviyeleri Tablosu

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

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

```java
package com.sirket.proje;

public class Musteri {

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

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

2. private - Özel

Sadece tanımlandığı sınıf içinden erişilebilir. En kısıtlı erişim seviyesidir.

```java
public class BankaHesabi {

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

    // private -> sadece bu sınıfın içinden çağrılabilir
    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 çağırabilir
    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 YardimciSinif {

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

    // default -> sadece bu paketten çağrılabilir
    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!

Başarılar Dilerim! 🚀