



## Bölüm 2

# Java Uygulamaları; I/O ve Operatörler

Java™ How to Program, 10/e



# AMAÇLAR

Bu hafta;

- Basit Java uygulamaları yazacak,
- Giriş/Çıkış ifadelerini görecekt,
- Java'nın ilkel (primitive) tiplerini öğrenecek,
- Basit bellek kavramlarını anlayacak,
- Aritmetik operatörleri kullanacak,
- Karar alma mekanizmaları yazacak,
- İlişkisel ve eşitlik operatörlerini kullanacaksınız.



---

2.1 Giriş

2.2 Java'da bir metin yazdırma

2.3 Java Programını düzenleme

2.4 Tamsayıları Toplama:

2.4.1. *import* Tanımlamaları

2.4.2 Addition sınıfını tanımlama

2.4.3 Scanner nesnesi tanımlama ve  
yaratma

2.4.4 Tamsayı değişkenleri tanımlama

2.4.5 Kullanıcıdan girdi isteme

2.4.6 Java API Dokümantasyonu

2.5 Bellek Kavramları

2.6 Aritmetik

2.7 Karar Verme

---



## 2.1 Giriş

- Java Uygulamalarının Programlanması
- Programları derleme ve çalıştırma için JDK araçlarını kullanınız.
- [www.deitel.com/books/jhttp10/](http://www.deitel.com/books/jhttp10/) 'daki videoları izleyebilirsiniz .



## 2.2 Java'da ilk Program

- ▶ Java uygulaması (**Java Application**)
  - java komutu ile Java Virtual Machine (JVM) üzerinden çalıştırdığınız bilgisayar programı,
- ▶ Fig 2.1'de örnek bir program görebiliyoruz.



```
1 // Fig. 2.1: Welcome1.java
2 // Text-printing program.
3
4 public class Welcome1
5 {
6     // main method begins execution of Java application
7     public static void main(String[] args)
8     {
9         System.out.println("Welcome to Java Programming!");
10    } // end method main
11 } // end class Welcome1
```

Welcome to Java Programming!

**Fig. 2.1** | Text-printing program.

## 2.2 Java'da ilk Program (Dvm.)

### *Programlarınızda Yorum Satırı Kullanma*

#### ► Yorumlar

`// Fig. 2.1: welcome1.java`

- `//` satırın **yorum** olduğunu belirtir..
- Program dökümantasyonu sağlar ve okunurluğunu artırır.
- Derleyici yorumları yok sayar.
- Bir yorum `//` ile başlıyorsa bu yorumun bulunduğu satırda yorum biter.

#### ► (Geleneksel Yorum )Traditional comment ise birden fazla satıra yayılabilmektedir.

`/* This is a traditional comment. It  
can be split over multiple lines */`

- Bu tipteki yorumlar `/*` ile başlar ve `*/` ile biter. Delimiterler (Sınırlayıcılar) arasında kalan metin derleyici tarafından yok sayılır.

## 2.2 Java'da ilk Program (Dvm.)

- ▶ **Javadoc comments** (Javadoc yorumları)
  - `/**` ve `*/` sınırlayıcıları ile sınırlandırılmıştır.
  - Javadoc comment delimiterleri ile sınırlanmış tüm metin derleyici tarafından yok sayılır.
  - Program dökümantasyonunu direk olarak kendi programlarınıza gömmeye yarar.
  - **javadoc utility programı** Javadoc yorumlarını okur ve HTML formatında bir program dökümantasyonu üretir (Appendix G'ye bakabilirsiniz.).





## Genel Programlama Hatası

Delimiterlardan bir tanesinin unutulması sintaks hatasına neden olmaktadır. Bir sintaks hatası, derleyicinin Java dili kurallarına uymayan bir durumla karşılaştığında gerçekleşmektedir.

Bu kurallar doğal dilin doğal bir dilin gramer kurallarına benzemektedir. Sintaks hatalarına derleme sırasında fark edildiği için **derleme hatası** ya da **derleme zamanı hatası** da denilmektedir.



## Öneri

Bazı kurumlar her programın başında, programın neyi amaçladığını, programın geliştiricisinin kim olduğunu ve hangi tarih ve zamanda geliştirildiği ile ilgili yorum satırları bulunmasını istemektedir.

## Hata Önleme Önerisi

Yeni programlar yazdıkça yorumlarınızı kodlarınızla birlikte **güncel tutmaya çalışınız**. Programcılar programın yaptıklarını geliştirmek ve hatalarını düzeltmek için sıklıkla kodda düzenlemeler yapmaktadır.

Yorumları güncel tutmak kodun ne yaptığını kesin olarak yansıtılabilmesini sağlar. Bu kodlarınızın anlaşılabilirliğini artırır. Güncel olmayan yorumların bulunduğu bir programı değiştirmeye çalışmak programcılarda yanlış varsayımlar oluşturabilmektedir.



## 2.2 Java'da ilk Program (Dvm.)

### *Boş Satırların Kullanımı*

- ▶ Boş satır, boşluk ve tab karakterleri
- ▶ Programın okunabilirliğini artırır.
- ▶ Bunların hepsi **white space** olarak bilinmektedir
- ▶ **white space** derleyici tarafından yok sayılmaktadır.



## Öneri

Programın okunurluğunu arttırabilmek için **white space** leri kullanın.

## 2.2 Java'da ilk Program (Dvm.)

Bir sınıf tanımlaması yapmak

► (Class declaration)

```
public class welcome1
```

- Her Java programının en az bir sınıfının olması gerekmektedir. `class anahtar` kelimesi ve ardından bu class'ın isminin belirtilmesi gerekmektedir.
- **Anahtar kelimeler** Java için reserve edilmiştir. Bu kelimeler küçük harfle başlamaktadır.



## 2.2 Java'da ilk Program (Dvm.)

*public Class için dosya ismi (Filename for a public Class)*

- ▶ Bir `public` class'ın bulunduğu dosyanın adı `ClassName.java` olmalıdır.
- ▶ Yani class `Welcome1`, `Welcome1.java` isimli bir dosya içerisinde depolanmalıdır.



# Genel Programlama Hatası

Public class'ın .java uzantısı ile biten dosya adı class'ın adı ile aynı değilse derleme hatası oluşur.



## 2.2 Java'da ilk Program (Dvm.)

### *Class İsimleri ve Belirleyiciler (Identifiers)*

- ▶ Geleneksel olarak bir sınıfın ismi büyük harfle başlamakta ve birden fazla kelime içeriyorsa her kelimenin ilk harfları büyük harf ile başlamaktadır. (örn, `SampleClassName`).
- ▶ Bir sınıf adı harfler, sayılar `_` ve `$` içerebilmektedir. Bu ad sayı ile başlayamaz ve boşluk karakteri içeremez.
- ▶ Java **case sensitive** bir dildir— küçük harf- büyük harf duyarlılığı vardır. Yani `a1` ve `A1` farklı (ancak kurala uygun) identifierlardır.



## 2.2 Java'da ilk Program (Dvm.)

### *Class Body (Bir Sınıfın İçeriği)*

- ▶ **left brace**, { her sınıfın body'sini başlatmaktadır.
- ▶ Buna karşılık gelen **right brace** } 'de sınıf tanımlamasını bitirmekte, sınıf body'sini kapatmaktadır.

## Öneri

- { ve }'ler arasındaki kod satırlarını bir seviye içeriden başlatmak (indent) önerilmektedir (IDE).
- Bu formatla sınıf tanımının neresi olduğu vurgulanmaktadır ve okunurluğu arttırmaktadır. Genellikle 2, 3 veya 4 boşlukla bu girintileme gerçekleştirilmektedir.



# Genel Programlama Hatası

{ yazdığınızda hemen onu kapatan }'i de yazınız. Bu şekilde küme parantezlerini kapatmayı unutmazsınız.

Küme parantezleri doğru bir şekilde kullanılmazsa derleme hatası alırsınız.

## 2.2 Java'da ilk Program (Dvm.)

### *Metot Tanımlamak (Declaring a Method)*

```
public static void main( String[] args )
```

- ▶ Her Java uygulamasının başlangıç noktası bu metottur.
- ▶ main tanımlayıcısından sonraki parantez, bir **metot** olarak adlandırılan bir program yapı taşı olduğunu gösterir.
- ▶ Java sınıfları normal olarak bir veya daha fazla yöntem içerir.
- ▶ **main** gösterilen şekilde tanımlanmalıdır; aksi takdirde, JVM uygulaması çalıştırmaz.
- ▶ Metotlar görevleri yerine getirir ve görevlerini tamamladıklarında bilgi döndürebilir.
- ▶ **void** anahtar kelimesi , bu yöntemin herhangi bir bilgi döndürmediğini belirtir.

## Öneri

- Metodun kod satırlarını bir seviye içeriden başlatmak (indent) önerilmektedir (IDE).
- Bu formatla method tanımının neresi olduğu vurgulanmaktadır ve metodun anlaşılabilirliği arttırmaktadır.



## 2.2 Java'da ilk Program (Dvm.)

### ► Metot Gövdesi

- {} ile tanımlanmaktadır.

### ► Statement (Bildirim)

```
System.out.println("welcome to Java Programming!");
```

- Bilgisayara bir işlem yapmasını söyler
  - Çift tırnak işaretleri arasında bulunan karakterleri görüntüle!
- Tırnak işaretleri ve aralarındaki karakterler birlikte, bir karakter dizgesi veya bir dizge sabiti olarak da bilinen bir dizedir **string**, **string literal**.
- **Stringlerdeki** beyaz boşluk karakterleri derleyici tarafından dikkate alınmaz.
- **Stringler**, birden fazla kod satırına yayılamaz.

# BİLGİ

- Derleme bir sintaks hatası verdiğinde, hata derleyicinin söylediği satırda olmayabilmektedir.
- Öncelikle, derleyicinin söylediği satıra bakınız, hata bulamazsanız önceki kod satırlarını da inceleyiniz.





## 2.2 Java'da ilk Program (Dvm.)

- ▶ `System.out` nesnesi
  - Standard output object.
  - Bir Java uygulamasının çalıştırdığı komut penceresine (`command window`) bilgi görüntülemesine izin verir.
  
- ▶ `System.out.println` metodu
  - Komut penceresinde bir satır metin görüntüler (veya basar).
  - Parantez içindeki string yöntemin argümanıdır (`argument`). Komut penceresindeki bir sonraki satıra çıktı imlecini konumlandırır.
  - Çoğu bildirim, noktalı virgül ile sona erer.

## 2.2 Java'da ilk Program (Dvm.)

### *İlk Java Uygulamanızı Derleme (Compiling Your First Java Application)*

- ▶ Bir komut penceresi açın ve programın depolandığı dizine geçin.
- ▶ Çoğu işletim sistemi dizinleri değiştirmek için `cd` komutunu kullanır.
- ▶ Programı derlemek için şunu yazın:
  - ▶ `javac welcome1.java`
- ▶ Derleme hatası yoksa, yukarıdaki komut, uygulamayı temsil eden platform bağımsız Java bayt kodlarını içeren bir `.class` dosyası (sınıf dosyası olarak bilinir) oluşturur.
- ▶ Uygulamayı belirli bir platformda yürütmek için `java` komutunu kullandığımızda, bu bayt kodu JVM tarafından alttaki işletim sistemi tarafından anlaşılan talimatlara çevrilir.

## HATA

- *Welcome1* sınıfı *Welcome1.java* isminde bir dosyada bulunmalıdır. Aksi halde “class *Welcome1* is public, should be declared in a file named *Welcome1.java* ” hatası çıkmaktadır.



## 2.2 Java'da ilk Program (Dvm.)

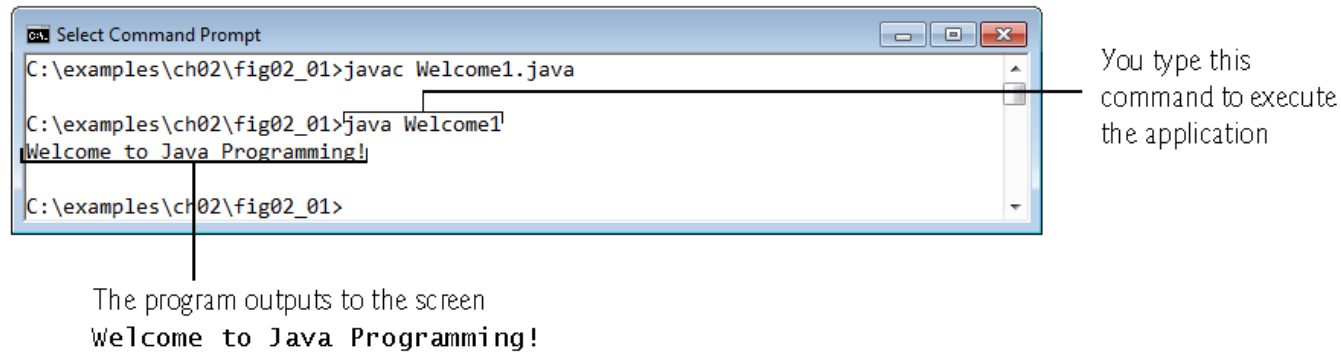
### *Executing the `welcome1` Application*

- ▶ Bu programı bir komut penceresinde yürütmek için dizini `welcome1.java` bulunan dizin olarak değiştirin.
- ▶ Ardından, `java Welcome1` yazın.
- ▶ Bu, `Welcome1.class` dosyasını yükleyen JVM'yi başlatır.
- ▶ Komut, `.class` dosya adı uzantısını atlar; aksi takdirde,
- ▶ JVM programı çalıştırmaz. JVM, sınıf `Welcome1`'in `main` metodunu çağırır.

# HATA



- Programı çalıştırırken “*Exception in thread “main” java.lang.NoClassDefFoundError*” hatası alırsanız *CLASSPATH* ortam değişkeni (environment Variable) düzgün bir şekilde ayarlanmamış demektir.
- Bazı sistemlerde bu değişkeni değiştirdikten sonra boot etmek gerekmektedir.



```
C:\examples\ch02\fig02_01>javac Welcome1.java
C:\examples\ch02\fig02_01>java Welcome1
Welcome to Java Programming!
C:\examples\ch02\fig02_01>
```

The program outputs to the screen  
Welcome to Java Programming!

You type this command to execute the application

**Fig. 2.2** | Executing `Welcome1` from the **Command Prompt**.

## 2.3 İlk Java Programınızı Değiştirme

- ▶ Şekil 2.3'te gösterilen Welcome2 sınıfı, Şekil 2.1'de gösterilenle aynı çıktıyı üretmek için iki ifade kullanır.
- ▶ New and key features in each code listing are highlighted
- ▶ `System.out`'un `print` metodu string yazdırır.
- ▶ `println`'den farklı olarak, `print` metodu, komut penceresinde bir sonraki satırın başında çıktı imlecini konumlandırmaz.
  - Programın bir sonraki karakteri, yazdırılan son karakterin hemen ardından belirir..



```
1 // Fig. 2.3: Welcome2.java
2 // Printing a line of text with multiple statements.
3
4 public class Welcome2
5 {
6     // main method begins execution of Java application
7     public static void main(String[] args)
8     {
9         System.out.print("Welcome to ");
10        System.out.println("Java Programming!");
11    } // end method main
12 } // end class Welcome2
```

Welcome to Java Programming!

**Fig. 2.3** | Printing a line of text with multiple statements.





## 2.3 İlk Java Programınızı Değiştirme (Dvm.)

- ▶ **Newline** karakterleri `System.out`'ın `print` and `println` methodlarına, çıktı imlecini komut penceresindeki bir sonraki satırın başlangıcına ne zaman yerleştirileceğini vurgular.
- ▶ **Newline** karakterleri `are whitespace` karakterleri olarak bilinmektedir.
- ▶ **Backslash** (`\`) karakteri **escape character**'ı olarak bilinir.
  - Ardında özel bir karakter olduğunu vurgular.
- ▶ **Backslash** bir **escape karakteri** oluşturmak için bir sonraki karakterle birleştirilir; `\n` yeni satır karakteri temsil eder.
- ▶ Tüm escape karakterlerine aşağıdan ulaşabilirsiniz.  
<http://docs.oracle.com/javase/specs/jls/se7/html/jls-3.html#jls-3.10.6>.



```
1 // Fig. 2.4: Welcome3.java
2 // Printing multiple lines of text with a single statement.
3
4 public class Welcome3
5 {
6     // main method begins execution of Java application
7     public static void main(String[] args)
8     {
9         System.out.println("Welcome\nto\nJava\nProgramming!");
10    } // end method main
11 } // end class Welcome3
```

```
Welcome
to
Java
Programming!
```

**Fig. 2.4** | Printing multiple lines of text with a single statement.

| Escape sequence | Description   |
|-----------------|---|
| <code>\n</code> | Newline. Position the screen cursor at the beginning of the <i>next</i> line.   |
| <code>\t</code> | Horizontal tab. Move the screen cursor to the next tab stop.  |
| <code>\r</code> | Carriage return. Position the screen cursor at the beginning of the <i>current</i> line—do <i>not</i> advance to the next line. Any characters output after the carriage return <i>overwrite</i> the characters previously output on that line. |
| <code>\\</code> | Backslash. Used to print a backslash character.   |
| <code>\"</code> | Double quote. Used to print a double-quote character. For example,<br><code>System.out.println("\"in quotes\")</code> ;<br>displays "in quotes".  |

**Fig. 2.5** | Some common escape sequences.

## 2.4 printf ile Metin Göstermek

- ▶ `System.out.printf` metodu
  - `f` “formatted” anlamına gelmektedir
  - displays *formatted* data
- ▶ Birden fazla method argümanı, **comma-separated** liste ile verilir.
- ▶ Bir metodu çağırmak **invoking** a method olarak da bilinir.
- ▶ Java uzun statementların birden fazla satırda bulunmasına izin verir.
  - Stringlerde bu durum geçerli değildir.
- ▶ `printf`’ methodunun ilk argümanı **format string**’idir.
  - İçerisinde standart metin ya da **format specifierlar** bulunabilir.
  - standart metin çıktısı `print` or `println` ile aynı olmaktadır.
  - Her biçim belirteci, bir değer için bir yer tutucudur ve ıktılacak veri türünü belirtir.
- Biçim belirteçleri yüzde işaretiyle (%) başlar ve ardından veri türünü temsil eden bir karakter gelir.
- ▶ Biçim belirteç `%s`, bir dize için bir yer tutucudur.



```
1 // Fig. 2.6: Welcome4.java
2 // Displaying multiple lines with method System.out.printf.
3
4 public class Welcome4
5 {
6     // main method begins execution of Java application
7     public static void main(String[] args)
8     {
9         System.out.printf("%s\n%s\n",
10             "Welcome to", "Java Programming!");
11     } // end method main
12 } // end class Welcome4
```

```
Welcome to
Java Programming!
```

**Fig. 2.6** | Displaying multiple lines with method `System.out.printf`.



## HATA

Bir tanımlayıcı veya string ortasında bir statementı(deyimi) bölmek sintaks hatasıdır.

## 2.5 Tamsayı Toplama Uygulaması

### ► Integers

- Tam sayılar, -22, 7, 0 ve 1024 gibi
- Programlar, bilgisayarın belleğindeki sayıları ve diğer verileri hatırlar ve bunlara değişken (**variables**) denilen program öğeleri aracılığıyla erişir.
- Şekil 2.7'deki program bu kavramları göstermektedir.



---

```
1 // Fig. 2.7: Addition.java
2 // Addition program that inputs two numbers then displays their sum.
3 import java.util.Scanner; // program uses class Scanner
4
5 public class Addition
6 {
7     // main method begins execution of Java application
8     public static void main(String[] args)
9     {
10         // create a Scanner to obtain input from the command window
11         Scanner input = new Scanner(System.in);
12
13         int number1; // first number to add
14         int number2; // second number to add
15         int sum; // sum of number1 and number2
16
17         System.out.print("Enter first integer: "); // prompt
18         number1 = input.nextInt(); // read first number from user
19
20         System.out.print("Enter second integer: "); // prompt
21         number2 = input.nextInt(); // read second number from user
22
```

---

**Fig. 2.7** | Addition program that inputs two numbers then displays their sum. (Part 1 of 2.)





```
23      sum = number1 + number2; // add numbers, then store total in sum
24
25      System.out.printf("Sum is %d\n", sum); // display sum
26  } // end method main
27 } // end class Addition
```

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

**Fig. 2.7** | Addition program that inputs two numbers then displays their sum. (Part 2 of 2.)



## 2.5.1 `import` Tanımlamaları

- ▶ Derleyiciye, programda kullanılan bir sınıfın bulunmasına yardımcı olur.
- ▶ Tekerleği tekrar yaratmaktan ziyade yeniden kullanabileceğiniz önceden tanımlı sınıflar seti bulunmaktadır.
- ▶ Sınıflar, ilgili sınıfların paketlere dayalı adlandırılmış gruplara (package) ayrılır ve bunlar topluca Java sınıfı kitaplığı, **Java class library**, veya Java Uygulama Programlama Arayüzü (Java API) **Java Application Programming Interface (Java API)** olarak adlandırılır.
- ▶ Bir Java programında kullanılan önceden tanımlı sınıfları bildirmek için ***import*** bildirimlerini kullanırsınız



## HATA

Tüm ***import*** bildirimleri, dosyadaki birinci sınıf bildiriminden önce olmalıdır.

Bir sınıf bildiriminde veya sonrasında bir ***import*** bildirimi yerleştirmek bir sözdizimi hatasıdır.

## HATA

Alınması gereken bir sınıf için bir ***import*** bildirimi eklemeyi unutursanız, "cannot find symbol" gibi bir ileti içeren bir derleme hatasına neden olur.

Bu durumda, doğru ***import*** bildirimleri sağladığınızdan ve bunların adlarının doğru olduğundan emin olun; uygun büyük harf kullanımı da dahil olmak üzere.

Her yeni Java sürümünde, API'ler genellikle hataları düzelten, performansı geliştiren veya görevleri yerine getirmek için daha iyi araçlar sunan yeni özellikler içerir. Karşılık gelen eski sürümlere artık gerek yoktur ve kullanılmamalıdır. Bu tür API'lerin kullanımdan çıkarıldığı söylenir ve daha sonraki Java sürümlerinden kaldırılabilir.

Online API belgelerine göz atarken çoğu zaman kullanımdan kaldırılmış API'lerle karşılaşabilirsiniz. Kapatılan API'leri kullanan kod derlediğinizde derleyici sizi uyaracaktır. Kodunuzu javac'la derlerseniz, derleyici size kullanımdan kaldırılmış özellikleri kullandığınızı söyleyecektir. Her biri için online belgeler (<http://docs.oracle.com/javase/7/docs/api/>), kullanımdan kaldırılanın yerini alan yeni sınıfları gösterir ve genellikle bu yeni sınıfa bağlanır.



## 2.5 Tamsayı Toplama Uygulaması (Dvm.)

### ▶ Scanner

- Bir programın, programda kullanmak üzere verileri okumasını sağlar.
- Veriler klavyedeki kullanıcı veya disk üzerindeki bir dosya gibi birçok kaynaktan gelebilir.
- Bir **Scanner nesnesi** kullanmadan önce, tarayıcıyı oluşturmalı ve verilerin kaynağını belirtmelisiniz.
- ▶ Bir tanımlamadaki eşittir işareti (=), değişkenin eşitlik işaretinin sağındaki ifadenin sonucuyla başlatılmasını (yani programda kullanılmaya hazır duruma getirilmesi gerektiğini) gösterir.
- ▶ **new** anahtar kelimesi bir nesne oluşturur.
- ▶ **Standard input nesnesi**, **System.in**, uygulamaların kullanıcı tarafından girilen verilerin baytlarını okumasını sağlar. **Scanner** nesnesi, bu baytları bir programda kullanılabilen türlere çevirir.

## 2.5.4 Tamsayı Depolayan Değişken Tanımlama

- ▶ Değişken tanımlama bildirimleri

```
int number1; // first number to add  
int number2; // second number to add  
int sum; // sum of number1 and number2
```

number1, number2 sum değişkenleri `int` tipinde değişken tutuyor

- `int` aralığı  $-2,147,483,648$  to  $+2,147,483,647$ .
- ▶ Aynı türden birçok değişken, bir bildirimde değişken adları virgüllerle ayrılmış olarak bildirilebilir.

## Öneri

Tanımlanan her değişkeni kendi bildiriminde tanımlayın. Bu format, her değişkenin yanında açıklayıcı bir yorumun eklenmesini sağlar.





## Öneri

Anlamlı değişken isimleri seçmek, bir programın kendiliğinden belgelendirilmesine yardımcı olur (diğer bir deyişle, programı ilişkili dokümanları okumaktan ziyade kendisini okuyarak veya fazla sayıda yorum oluşturarak ve görüntüleyerek anlayabilirsiniz)



## Öneri

Genel olarak, değişken adı tanımlayıcıları küçük harfle başlar ve ilk kelimedenden sonra gelen her kelime büyük harfle başlar.

Örneğin, değişken adı tanımlayıcı `firstNumber`, ikinci kelimesi `Number` ' büyük harf `N` ile başlatır.

Bu adlandırma notasyonu `camelCase` olarak bilinir; çünkü büyük harfler bir devenin hörgüçleri gibi öne çıkmaktadır.



## 2.5.5 java.lang

### ► Class System

- `java.lang` paketinin bir parçasıdır.
- Class System import edilmez.
- Varsayılan olarak, paket `java.lang` paketi bir Java programında import edilir; Bu nedenle `java.lang`'daki sınıflar Java API'sinde import beyanı gerektirmeyen yalnızca sınıflardır.

## 2.5.6 Kullanıcıdan girdi olarak bir int elde etme

### ► Scanner method `nextInt`

```
number1 = input.nextInt(); // read first number from user
```

- Klavyedeki kullanıcıdan bir tamsayı alır.
- Program, kullanıcının numarayı yazmasını ve numarayı programa göndermek için Enter tuşuna basmasını bekler.
- `nextInt` metodunun çıktısı **atama (assignment operatorü)**, `=` ile `number1` değişkenine atanır.
- “`number1`, `input.nextInt()`. değerini alır”
- Operator = is **binary operator olarak adlandırılır**—iki tane *two operandı (işleneni) olduğu için*.
- Atama operatörünün sağındaki her şey `=`, atama yapılmadan önce hesaplanır.

## 2.5 Kullanıcıdan girdi olarak bir int elde etme (Dvm.)



### ► Aritmetik

```
sum = number1 + number2; // add numbers then store total  
in sum
```

- number1 ve number2 değişkenlerinin toplamını hesaplayan atama ifadesi, atama işleci = kullanılarak, sum değişkenine sonucunu atar.
- “sum number1 + number2. değerini alır”
- Hesaplamaları içeren bildirimlerin bölümleri **expressions** olarak isimlendirilir.
- Bir **expressions** , ifadenin ilişkili bir değere sahip herhangi bir bölümüdür.



## 2.5.9 Hesaplamanın Sonucunu Görüntüleme

- ▶ Integer formatındaki çıktı

```
System.out.printf( "Sum is %d%n", sum );
```

- Biçim belirteci % d, bir int değeri için yer tutucudur
- d harfi “decimal integer” anlamına gelmektedir.



## 2.6 Bellek Kavramları (Memory Concepts)

### ► Değişkenler

- Her değişkenin **adı**, **türü**, **boyutu** (bayt cinsinden) ve bir **değeri** vardır.
- Yeni bir değer bir değişkene yerleştirildiğinde, yeni değer önceki değerın yerine geçer (varsa).
- Önceki değer kaybolur, bu nedenle bu işlemin yıkıcı olduğu söylenir.

---

number1

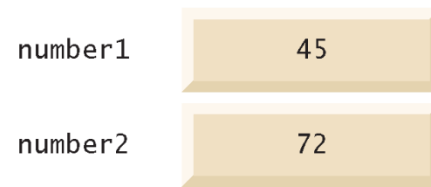


45

---

**Fig. 2.8** | Memory location showing the name and value of variable `number1`.





---

**Fig. 2.9** | Memory locations after storing values for `number1` and `number2`.

---

|         |     |
|---------|-----|
| number1 | 45  |
| number2 | 72  |
| sum     | 117 |

---

**Fig. 2.10** | Memory locations after storing the sum of `number1` and `number2`.

## 2.7 Aritmetik

- ▶ Aritmetik operatörler Şekil 2.11'de özetlenmiştir.
- ▶ Yıldız (\*) çarpımı gösterir
- ▶ Yüzde işareti (%), kalan operatördür
- ▶ Aritmetik işleçler ikili (binary) işleçlerdir çünkü her biri iki işlenen üzerinde çalışırlar.
- ▶ Tamsayı bölme, tam sayı bölüm verir (truncate).

| Java operation | Operator | Algebraic expression                                 | Java expression    |
|----------------|----------|--|--------------------|
| Addition       | +        | $f + 7$  | <code>f + 7</code> |
| Subtraction    | -        | $p - c$  | <code>p - c</code> |
| Multiplication | *        | $bm$   | <code>b * m</code> |
| Division       | /        | $x / y$ <b>or</b> $\frac{x}{y}$ <b>or</b> $x \div y$ | <code>x / y</code> |
| Remainder      | %        | $r \bmod s$  | <code>r % s</code> |

**Fig. 2.11** | Arithmetic operators.



## 2.7 Arithmetic (Cont.)

- ▶ Parantezler, ifadelerdeki terimleri cebirsel ifadelerde olduğu gibi gruplamak için kullanılır.
- ▶ Bir deyim iç içe parantez içeriyorsa, en içteki parantez dizisindeki ifade değerlendirilir.

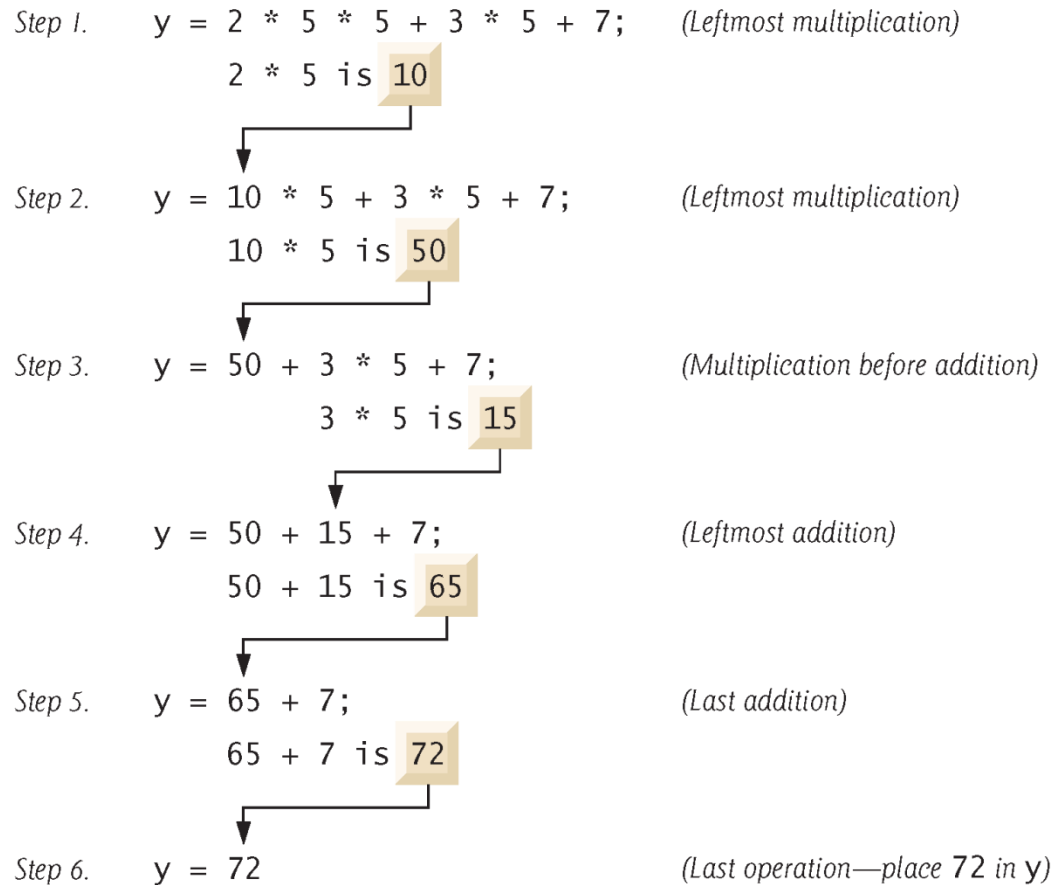
## 2.7 Arithmetic (Cont.)

- ▶ Operatör önceliği kuralları
- ▶ Çarpma, bölme ve mod alma işlemleri önce uygulanır. Bir ifade birkaç işlem içerirse, bunlar soldan sağa uygulanır. Çarpma, bölme ve kalan operatörler aynı öncelik düzeyine sahiptir.
- ▶ Toplama ve çıkarma işlemleri daha sonra uygulanır. Bir ifade içerisinde böyle işlemler varsa, operatörler soldan sağa uygulanır.
- ▶ Toplama ve çıkarma operatörleri aynı öncelik düzeyine sahiptir. Operatörlerin soldan sağa sürüldüğünü söylersek, onların birleştiriciliğine atıfta bulunuyoruz. Bazı operatörler sağdan sola ilişki kurarlar. Komple öncelikler tablosu Ek A'da bulunmaktadır.



| Operator(s) | Operation(s)                            | Order of evaluation (precedence)  |
|-------------|---|---|
| *<br>/<br>% | Multiplication<br>Division<br>Remainder | Evaluated first. If there are several operators of this type, they're evaluated from <i>left to right</i> . |
| +<br>-      | Addition<br>Subtraction                 | Evaluated next. If there are several operators of this type, they're evaluated from <i>left to right</i> .  |
| =           | Assignment                              | Evaluated last.   |

**Fig. 2.12** | Precedence of arithmetic operators.



**Fig. 2.13** | Order in which a second-degree polynomial is evaluated.



## 2.8 Karar Verme: Eşitlik and İlişkisel Operatörleri

- ▶ **Condition**
  - Bir ifade `true` ya da `false` olabilir.
- ▶ **if selection statement**
  - Bir programın bir koşulun değerine dayalı olarak bir karar vermesine izin verir.
- ▶ **Equality operators** (`==` and `!=`)
- ▶ **Relational operators** (`>`, `<`, `>=` and `<=`)
- ▶ Her iki eşitlik operatörü de, ilişkisel operatörlerinkinden daha düşük, önceliği aynı seviyededir.
- ▶ Eşitlik operatörleri soldan sağa işlenirler.
- ▶ İlişkisel operatörlerin hepsi aynı önceliğe sahiptir ve aynı zamanda soldan sağa doğru işlenirler.

| Algebraic operator          | Java equality or relational operator | Sample Java condition | Meaning of Java condition       |
|-----------------------------|--------------------------------------|-----------------------|---------------------------------|
| <i>Equality operators</i>   |                                      |                       |                                 |
| =                           | ==                                   | x == y                | x is equal to y                 |
| ≠                           | !=                                   | x != y                | x is not equal to y             |
| <i>Relational operators</i> |                                      |                       |                                 |
| >                           | >                                    | x > y                 | x is greater than y             |
| <                           | <                                    | x < y                 | x is less than y                |
| ≥                           | >=                                   | x >= y                | x is greater than or equal to y |
| ≤                           | <=                                   | x <= y                | x is less than or equal to y    |

**Fig. 2.14** | Equality and relational operators.



---

```
1 // Fig. 2.15: Comparison.java
2 // Compare integers using if statements, relational operators
3 // and equality operators.
4 import java.util.Scanner; // program uses class Scanner
5
6 public class Comparison
7 {
8     // main method begins execution of Java application
9     public static void main(String[] args)
10    {
11        // create Scanner to obtain input from command line
12        Scanner input = new Scanner(System.in);
13
14        int number1; // first number to compare
15        int number2; // second number to compare
16
17        System.out.print("Enter first integer: "); // prompt
18        number1 = input.nextInt(); // read first number from user
19
20        System.out.print("Enter second integer: "); // prompt
21        number2 = input.nextInt(); // read second number from user
22    }
```

---

**Fig. 2.15** | Compare integers using if statements, relational operators and equality operators. (Part I of 3.)



```
23     if (number1 == number2)
24         System.out.printf("%d == %d%n", number1, number2);
25
26     if (number1 != number2)
27         System.out.printf("%d != %d%n", number1, number2);
28
29     if (number1 < number2)
30         System.out.printf("%d < %d%n", number1, number2);
31
32     if (number1 > number2)
33         System.out.printf("%d > %d%n", number1, number2);
34
35     if (number1 <= number2)
36         System.out.printf("%d <= %d%n", number1, number2);
37
38     if (number1 >= number2)
39         System.out.printf("%d >= %d%n", number1, number2);
40 } // end method main
41 } // end class Comparison
```

**Fig. 2.15** | Compare integers using if statements, relational operators and equality operators. (Part 2 of 3.)



```
Enter first integer: 777
Enter second integer: 777
777 == 777
777 <= 777
777 >= 777
```

```
Enter first integer: 1000
Enter second integer: 2000
1000 != 2000
1000 < 2000
1000 <= 2000
```

```
Enter first integer: 2000
Enter second integer: 1000
2000 != 1000
2000 > 1000
2000 >= 1000
```

**Fig. 2.15** | Compare integers using if statements, relational operators and equality operators. (Part 3 of 3.)