



# Arrays ve ArrayLists

Java™ How to Program, 10/e



# Amaçlar

- ▶ Arraylerin ne olduğu
- ▶ Veri depolama ve kullanmak için array kullanımı
- ▶ Array Tanımlama, ilklendirme ve arraylerin belirli alanlarına erişme
- ▶ Gelişmiş *for* yapısı ile arrayler üzerinde iterasyon kurma,
- ▶ Metotlara parametre olarak array gönderme,
- ▶ Çok boyutlu array tanımlama
- ▶ Değişik boyutlarda argüman listesi oluşturma,



# Amaçlar

- ▶ Arraylerin ne olduğu
- ▶ Komut satırından parametre alma
- ▶ Dinamik şekilde boyutu olabilen arraye benzer *ArrayList* yapısını kullanma



---

## 7.1 Introduction

## 7.2 Arrays

## 7.3 Declaring and Creating Arrays

## 7.4 Examples Using Arrays

7.4.1 Creating and Initializing an Array

7.4.2 Using an Array Initializer

7.4.3 Calculating the Values to Store in an Array

7.4.4 Summing the Elements of an Array

7.4.5 Using Bar Charts to Display Array Data Graphically

7.4.6 Using the Elements of an Array as Counters

7.4.7 Using Arrays to Analyze Survey Results

## 7.5 Exception Handling: Processing the Incorrect Response

7.5.1 The `try` Statement

7.5.2 Executing the `catch` Block

7.5.3 `toString` Method of the Exception Parameter

## 7.6 Case Study: Card Shuffling and Dealing Simulation

## 7.7 Enhanced `for` Statement

---



---

**7.8** Passing Arrays to Methods

**7.9** Pass-By-Value vs. Pass-By-Reference

**7.10** Case Study: Class **GradeBook** Using an Array to Store Grades

**7.11** Multidimensional Arrays

**7.12** Case Study: Class **GradeBook** Using a Two-Dimensional Array

**7.13** Variable-Length Argument Lists

**7.14** Using Command-Line Arguments

**7.15** Class **Arrays**

**7.16** Introduction to Collections and Class **ArrayList**

**7.17** (Optional) GUI and Graphics Case Study: Drawing Arcs

**7.18** Wrap-Up

---



# 7.1 Giriş

- ▶ Data structures (Veri Yapıları)
  - İlgili veri elemanlarının koleksiyonları
- ▶ Array objects (Dizi nesneler)
  - Aynı tipteki veri elemanlarının oluşturduğu veri
  - Aynı tipteki elemanları işlemek için kullanım kolaylığı sağlar.
  - Bir kere yaratıldıkten sonra aynı boyutta kalır.
- ▶ Arrayler üzerinde gelişmiş `for` yapısı ile elemanlar tek tek gezilebilmektedir.
- ▶ Değişken uzunluktaki argüman listeleri
  - Bu şekilde farklı sayıarda argümanı alan metodlar oluşturulabilmektedir.



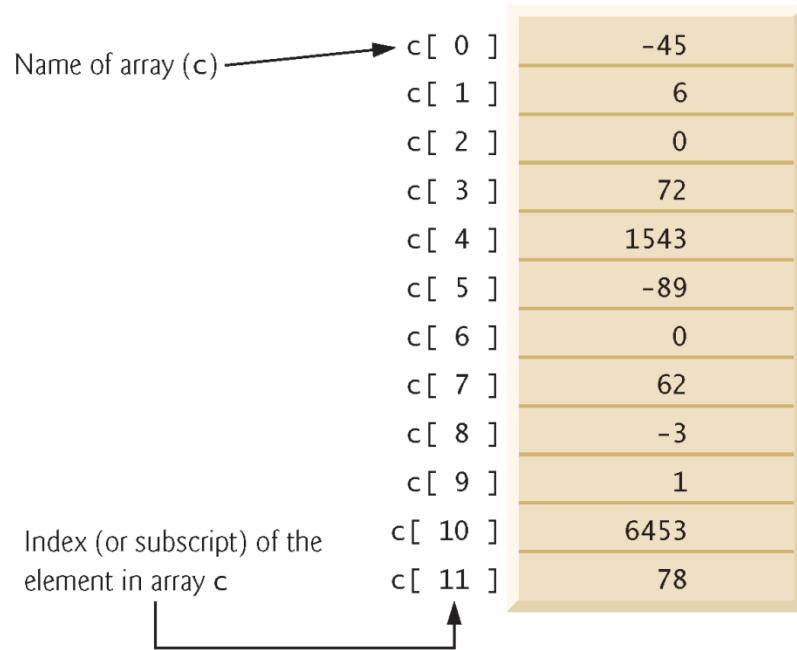
## 7.1 Giriş(Dvm.)

- ▶ `java.util` paketi içerisindeki `Arrays` sınıfının `static` metotları ile array işlemleri
- ▶ `ArrayList` koleksiyonu
  - Array'e benziyor
  - **Dynamic resizing**
    - Daha fazla eleman için ekstra bellek kullanabiliyor ya da daha az eleman kullanması gerektiğinde belleği sisteme geri verebilir.
- ▶ Java SE 8
  - 17. Bölümden sonra, Java SE 8 Lambda ve Streamleri ile, bu bölümdeki örnekleri daha akıllıca ve performanslı şekilde oluşturabilirsiniz.



## 7.2 Arrays (Diziler)

- ▶ **Array**
  - Aynı tipte veri içeren bir grup değişken
  - Arrayler referans tipinde nesnelerdir.
  - Elemanları ilkel (primitive) tipte ya da referans tipinde olabilmektedir.
- **Array’deki belirli bir elemenin erişmek için**
  - Bu elemanın **index** değeri kullanılır.
  - **Array-access expression**— arrayin adı ve **square brackets**, [ ] içerisinde indis .
- ▶ **Array’ın ilk elemanın indisini sıfırdır.**
- ▶ Dizinin son elemanın indisini arraydeki eleman sayısından bir eksiktir.
- ▶ **Array isimleri için kullanılan notasyon diğer değişkenlerde kullanılan ile aynıdır.**



**Fig. 7.1** | A 12-element array.



## 7.2 Arrays (Dvm.)

- ▶ Array indexi negatif olmayan bir tamsayı olmalıdır.
- ▶ Index yerine bir matematiksel ifade de kullanılabilir.
- ▶ Her array nesnesi kendi uzunluğunu bilir ve bunu kendi `length` instance variable'ında tutar.
- ▶ `length` özelliği `final` olduğundan dolayı değişken değildir.



## BİLGİ

İndex **int** olmalıdır. **Byte**, **short** ya da **char** da kabul edilebilir ancak **long** olmamalıdır. **long** tipinde bir index değişkeni kullanırsanız derleme hatası oluşacaktır.

## 7.3 Arrayleri Tanımlama ve Yaratma

- ▶ Array nesneleri
  - `new` anahtar kelimesi kullanılarak yaratılır.
  - Eleman tipi ve kaç elemanlık bir array oluşturulacağı array yaratma ifadesinde kullanılır.
- ▶ 12 adet `int` tipinde elemanı olan bir array yaratma ifadesi
  - ▶ `int[] c = new int[12];`
  - ▶ İki satırda da şu şekilde ifade edilebilmektedir.  
`int[] c; // declare the array variable`  
`c = new int[12]; // creates the array`



## 7.3 Arrayleri Tanımlama ve Yaratma (Dvm.)

- ▶ Tanımlamada, tipten sonraki *square brackets* tanımlananın bir array olduğunu vurgulamaktadır.
- ▶ Dizi yaratıldığında array'in her elemanı varsayılan değerlerini almaktadır.
- ▶ Sayısal primitive tipteki elemanlar için 0, boolean elemanlar için **false** ve referans tipindeki elemanlar için **null** dır.



## BİLGİ

Array tanımlamada sol taraftaki parantezin içérisine arrayin boyutunu tanımlama derleme hatasıdır.

```
int [12] c; // HATA
```



## 7.3 Arrayleri Tanımlama ve Yaratma

- ▶ Başlangıçta eleman tipi ve köşeli parantez verildiyse ardındaki tüm tanımlamaları da array değişkenleri tanımlamaktadır.
  - Okunabilirlik açısından her satıda tek bir değişken tanımlanması önerilmektedir.
  - Yani int [] a,b,c;
  - Tanımlamasına izin verilmektedir bu durumda a,b ve c hepsi birer arraydir.
  - Ancak sadece a'nın array diğerlerinin int tipinde olmasını istersek
  - int a[], b, c;'ye izin verilmektedir.

## 7.3 Arrayleri Tanımlama ve Yaratma

- ▶ İlkel tipte tanımlanmış her eleman bu tiptedir.
- ▶ Referans tipinde tanımlanmış her eleman tanımlanmış tipte bir nesneye referans içermektedir.
  - Örn String tipinde bir arrayin her bir elemanı bir String nesnesine işaret etmektedir.

## 7.4 Array Kullanım Örnekleri

- ▶ Bu bölümde array tanımlama, array yaratma ve array elemanları üzerinde işlem yapmak için çeşitli örnekler göreceğiz.

## 7.4.1 Array Yaratma ve İlklenendirme

- ▶ Fig. 7.2'de anahtar kelimesi ile ilk değerleri 0 olan 10 elemanlı bir `int` arrayi oluşturulmuş.



---

```
1 // Fig. 7.2: InitArray.java
2 // Initializing the elements of an array to default values of zero.
3
4 public class InitArray
5 {
6     public static void main(String[] args)
7     {
8         // declare variable array and initialize it with an array object
9         int[] array = new int[10]; // create the array object
10
11     System.out.printf("%s%8s%n", "Index", "Value"); // column headings
12
13     // output each array element's value
14     for (int counter = 0; counter < array.length; counter++)
15         System.out.printf("%5d%8d%n", counter, array[counter]);
16     }
17 } // end class InitArray
```

---

**Fig. 7.2** | Initializing the elements of an array to default values of zero. (Part I of 2.)



Index	Value
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

**Fig. 7.2** | Initializing the elements of an array to default values of zero. (Part 2 of 2.)

## 7.4.2 Array İlkendirici Kullanma (Using an Array Initializer )

### ▶ Array initializer

- Küme parantezleri içerisinde ilk değer listesi verilerek array tanımlanabilmektedir.
- Arrayin boyu (length'i) atanan listedeki eleman sayısı ile belirlenmektedir.

```
int[] n = {10, 20, 30, 40, 50};
```

- 5 elemanlı index değerleri 0 ile 4 arasında olan bir array oluşturmaktadır.
- Derleyici arrayin boyutunu kendi saymaktadır.
- Uygun new operasyonunu kendisi halletmektedir.



---

```
1 // Fig. 7.3: InitArray.java
2 // Initializing the elements of an array with an array initializer.
3
4 public class InitArray
5 {
6     public static void main(String[] args)
7     {
8         // initializer list specifies the initial value for each element
9         int[] array = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
10
11     System.out.printf("%s%8s%n", "Index", "Value"); // column headings
12
13     // output each array element's value
14     for (int counter = 0; counter < array.length; counter++)
15         System.out.printf("%5d%8d%n", counter, array[counter]);
16
17 } // end class InitArray
```

---

**Fig. 7.3** | Initializing the elements of an array with an array initializer. (Part I of 2.)



Index	Value
0	32
1	27
2	64
3	18
4	95
5	14
6	90
7	70
8	60
9	37

**Fig. 7.3** | Initializing the elements of an array with an array initializer. (Part 2 of 2.)

## 7.4.3 Arraydeki eleman değerlerini otomatik oluşturmak

- ▶ Fig. 7.4'de gösterilen uygulamada 10 elemanlı bir array yaratılmakta her bir elemanına 2'den 20'ye (2, 4, 6, ..., 20) değerleri verilmektedir.



---

```
1 // Fig. 7.4: InitArray.java
2 // Calculating the values to be placed into the elements of an array.
3
4 public class InitArray
5 {
6     public static void main(String[] args)
7     {
8         final int ARRAY_LENGTH = 10; // declare constant
9         int[] array = new int[ARRAY_LENGTH]; // create array
10
11        // calculate value for each array element
12        for (int counter = 0; counter < array.length; counter++)
13            array[counter] = 2 + 2 * counter;
14
15        System.out.printf("%s%8s%n", "Index", "Value"); // column headings
16
17        // output each array element's value
18        for (int counter = 0; counter < array.length; counter++)
19            System.out.printf("%5d%8d%n", counter, array[counter]);
20    }
21 } // end class InitArray
```

---

**Fig. 7.4** | Calculating the values to be placed into the elements of an array. (Part I of 2.)



Index	Value
0	2
1	4
2	6
3	8
4	10
5	12
6	14
7	16
8	18
9	20

**Fig. 7.4** | Calculating the values to be placed into the elements of an array. (Part 2 of 2.)

## 7.4 Array Kullanım Örnekleri (Dvm.)

- ▶ `final` değişkenleri kullanılmadan önce değişkenler ilk değerlerini almalıdır.
- ▶ `final` bir değişkeni ilk değerini aldıktan sonra değiştirmeye çalışmak derleme hatasına neden olur.
- ▶ `final` bir değişkene ilk değerini vermeden değiştirmeye çalışmak derleme hatasına neden olmaktadır.
  - “variable *variableName* might not have been initialized”



## BİLGİ

Sabit değerler programı literallerle yazmaktan daha okunur hale getirmektedir. 10 yerine `ARRAY_LENGTH` kullanılması programın çok daha rahat okunmasına neden olur.

Sabit isimlerindeki notasyon yukarıdaki gibi dir

## 7.4.4 Array Elemanlarını toplama

- ▶ Figure 7.5 10 elemandan oluşan bir arrayin toplanmasını göstermektedir.



```
1 // Fig. 7.5: SumArray.java
2 // Computing the sum of the elements of an array.
3
4 public class SumArray
5 {
6     public static void main(String[] args)
7     {
8         int[] array = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
9         int total = 0;
10
11        // add each element's value to total
12        for (int counter = 0; counter < array.length; counter++)
13            total += array[counter];
14
15        System.out.printf("Total of array elements: %d%n", total);
16    }
17 } // end class SumArray
```

```
Total of array elements: 849
```

**Fig. 7.5** | Computing the sum of the elements of an array.



## 7.4.5 Array verisini grafiksel olarak gösterebilmek için Bar Chart Kullanmak

- ▶ Bir çok program verilerini kullanıcılarına grafiksel olarak göstermektedir.
- ▶ Sayısal veriler bar chart ile gösterilebilmektedir.
  - Daha yüksek bar daha yüksek büyülüğe sahip veri demektir.
  - Veriyi bar chart olarak göstermenin basit bir yolu bar değerlerini \* ile göstermektir.
  - **%02d** format belirleyicisi **int** değerinin iki hane ile gösterilmesini sağlamaktadır.
  - 0 flagi ise değeri iki haneden küçük olan verinin ilk hanesine 0 konulması içindir.



---

```
1 // Fig. 7.6: BarChart.java
2 // Bar chart printing program.
3
4 public class BarChart
5 {
6     public static void main(String[] args)
7     {
8         int[] array = { 0, 0, 0, 0, 0, 0, 1, 2, 4, 2, 1 };
9
10        System.out.println("Grade distribution:");
11
12        // for each array element, output a bar of the chart
13        for (int counter = 0; counter < array.length; counter++)
14        {
15            // output bar label ("00-09: ", ..., "90-99: ", "100: ")
16            if (counter == 10)
17                System.out.printf("%5d: ", 100);
18            else
19                System.out.printf("%02d-%02d: ",
20                                  counter * 10, counter * 10 + 9);
21
22            // print bar of asterisks
23            for (int stars = 0; stars < array[counter]; stars++)
24                System.out.print("*");
```

---

**Fig. 7.6** | Bar chart printing program. (Part I of 2.)



---

```
25
26         System.out.println();
27     }
28 }
29 } // end class BarChart
```

Grade distribution:

00-09:  
10-19:  
20-29:  
30-39:  
40-49:  
50-59:  
60-69: \*70-79: \*\*  
80-89: \*\*\*  
90-99: \*\*  
100: \*

**Fig. 7.6** | Bar chart printing program. (Part 2 of 2.)



## 7.4.6 Bir Arrayin Elemanlarını Sayaç Olarak Kullanmak

- ▶ Bazı programlar veriyi özetlemek –bir araştırma sonuçları gibi- için sayıç değişkenleri kullanabilirler.
- ▶ Fig. 6.7'de altı yüzlü bir zarın 6,000,000 her bir yüzünün kaç defa geldiğini sayan bir program bulunmaktadır.
- ▶ Fig. 7.7'de ise bu programın array versiyonu bulunuyor.
  
- ▶ **frequency** array'i 6 tane sayacı tutacak kadar büyük olmalıdır.
  - **frequency[0]** 'ı yok saydığımızdan 7 elemanlı bir array oluşturduk .



---

```
1 // Fig. 7.7: RollDie.java
2 // Die-rolling program using arrays instead of switch.
3 import java.security.SecureRandom;
4
5 public class RollDie
6 {
7     public static void main(String[] args)
8     {
9         SecureRandom randomNumbers = new SecureRandom();
10        int[] frequency = new int[7]; // array of frequency counters
11
12        // roll die 6,000,000 times; use die value as frequency index
13        for (int roll = 1; roll <= 6000000; roll++)
14            ++frequency[1 + randomNumbers.nextInt(6)];
15
16        System.out.printf("%s%10s%n", "Face", "Frequency");
17
18        // output each array element's value
19        for (int face = 1; face < frequency.length; face++)
20            System.out.printf("%4d%10d%n", face, frequency[face]);
21    }
22 } // end class RollDie
```

---

**Fig. 7.7** | Die-rolling program using arrays instead of switch. (Part 1 of 2.)



Face	Frequency
1	999690
2	999512
3	1000575
4	999815
5	999781
6	1000627

**Fig. 7.7** | Die-rolling program using arrays instead of switch. (Part 2 of 2.)

## 7.4.7 Araştırma Sonuçlarını Analiz Etmek için Array Kullanmak

- ▶ Figure 7.8 bir araştırmada toplanan verilerin sonuçlarını ölçmek için arrayleri kullanmaktadır.
  - *20 öğrenci kafeteryadaki yemekleri beğenme oranlarına göre 1 ile 5 arasında puanlaşacaklar. 1 çok kötü, 5 çok lezzetli olacak şekilde. 20 yanıtı bir arrayde tutarak cevapların oranlarını hesaplayalım.*
- ▶ **responses** arrayi 20-elemanlı bir **int** arrayidir.
- ▶ 6-elemanlı **frequency** arrayi ise her puanın kaç kere söylendiğini tutmaktadır. Her eleman yaratıldığında varsayılan olarak 0 değerini almaktadır.
  - **frequency[0]** ’I yok saydık.



---

```
1 // Fig. 7.8: StudentPoll.java
2 // Poll analysis program.
3
4 public class StudentPoll
5 {
6     public static void main(String[] args)
7     {
8         // student response array (more typically, input at runtime)
9         int[] responses = { 1, 2, 5, 4, 3, 5, 2, 1, 3, 3, 1, 4, 3, 3, 3,
10            2, 3, 3, 2, 14 };
11         int[] frequency = new int[6]; // array of frequency counters
12 }
```

---

**Fig. 7.8** | Poll analysis program. (Part I of 3.)



```
13 // for each answer, select responses element and use that value
14 // as frequency index to determine element to increment
15 for (int answer = 0; answer < responses.length; answer++)
16 {
17     try
18     {
19         ++frequency[responses[answer]];
20     }
21     catch (ArrayIndexOutOfBoundsException e)
22     {
23         System.out.println(e); // invokes toString method
24         System.out.printf("    responses[%d] = %d%n%n",
25                           answer, responses[answer]);
26     }
27 }
28
29 System.out.printf("%s%10s%n", "Rating", "Frequency");
30
31 // output each array element's value
32 for (int rating = 1; rating < frequency.length; rating++)
33     System.out.printf("%6d%10d%n", rating, frequency[rating]);
34 }
35 } // end class StudentPoll
```

**Fig. 7.8** | Poll analysis program. (Part 2 of 3.)

```
java.lang.ArrayIndexOutOfBoundsException: 14  
    responses[19] = 14
```

Rating Frequency

1	3
2	4
3	8
4	2
5	2

**Fig. 7.8** | Poll analysis program. (Part 3 of 3.)

## 7.4.7 Araştırma Sonuçlarını Analiz Etmek için Array Kullanmak (CoDvm.)

- ▶ **responses** arrayindeki herhangi bir elemanın geçersiz bir değeri varsa, örn :14 ki bu durumda **frequency[14]** 'e bir eklenmeye çalışılmaktadır.
  - Java buna izin vermemektedir.
  - JVM array indislerini kontrol ederek sınırların aşılımadığını garantilemektedir.
  - Eğer program yanlış bir indeks kullanırsa, Java exception denilen bir çalışma zamanı hatası üretir.

## 7.5 Exception Handling: Hatalı Durumları İşlemek

- ▶ Bir exception olması, program çalışırken bir hata olduğunu vurgulamaktadır.
- ▶ “exception” problemin nadiren olduğunu vurgulamaktadır. Rule “Kural” programın düzgün çalışmasını tanımlıyorsa, problem kurala harici (“exception”) bir durum oluşturmaktadır.
- ▶ Exception handling , hata toleranslı (fault-tolerant programs) oluşturmanıza yardımcı olur.



## 7.5 Exception Handling: Hatalı Durumları İşlemek (DVM.)

- ▶ JVM bir yanlış array indeksi ya da metoda beklenmeyen bir argüman gönderilmesi gibi bir hata olduğunu farkettiğinde bir exception fırlatır (**exception throws**).



## 7.5.1 try İfadesi

- ▶ Exception'ı düzeltmek için, exception fırlatılabilecek kod blogunu **try** ifadesinin içerisine yerleştirmek gereklidir.
- ▶ **try** blogu exception fırlatılabilecek (hata oluşabilecek) kodu kapsamalıdır.
- ▶ **catch** blogu hatayı, exceptionın düzeltilmesi ile uğraşan kodu içermektedir.
- ▶ Try blogu içerisinde oluşabilecek farklı exception tiplerine özel ayrı catch bloklarını oluşturabilirsiniz.

## 7.5.2 catch Block

- ▶ Program, responses dizisindeki geçersiz değer 14 ile karşılaştığında, dizinin sınırlarının dışında olan **frequency[14]** 'e 1 ekleme girişiminde bulunur - frek **frequency** dizisi yalnızca altı elemana sahiptir (0–5 arası indisler).
- ▶ Dizi sınır denetimi çalışma zamanında gerçekleştirildiğinden, JVM bir exception oluşturur - özellikle 19 satırı bu sorun programa bildirmek için `ArrayIndexOutOfBoundsException` öğesini atar.
- ▶ Bu noktada, **try** bloğu sona erer ve **catch** bloğu çalışmaya başlar - try blogunda herhangi bir yerel değişken bildirdiyseniz, artık kapsam dışıdır.



## 7.5.2 **catch** Bloğunu çalıştırma (Dvm.)

- ▶ **catch** bloğu, (IndexOutOfRangeException) türünde bir exception parametresi (e) bildirir.
- ▶ **catch** bloğu içinde, yakalanan exception nesnesiyle etkileşimde bulunmak için parametrenin adını kullanabilirsiniz.

## 7.5.3 Exception Parametresinin toString Metodu

- ▶ Exception nesnesinin `toString` metodu, exception nesnesinde saklanan hata iletisini döndürür.
- ▶ Program kontrolü, `catch` bloğunun kapanış sağ küme parantezine ulaştığında exceptionın giderildiği kabul edilir.

## 7.6 Case Study: Kağıt Karma and Simülasyon Oluşturma

- ▶ Şimdije kadar gördüğümüz örneklerde ilkel tiplerin elemanlarını içeren diziler kullanmıştır.
- ▶ Bir dizinin elemanları ilkel türler veya referans türleri olabilir.
- ▶ Bu örnekte referans tipindeki elemanları olan bir array oluşturacağız. Bu örnekte iskambil kağıtlarını karmaya çalışacağız.

## 7.6 Case Study: Kağıt Karma and Simülasyon Oluşturma (Dvm)

- ▶ **Card Sınıfı** (Fig. 7.9) iki **String** instance variable'ına sahiptir—**face** ve **suit**—belirli bir kağıdın tipi ve hangi eleman olduğunu tutmaktadır.
- ▶ **toString** metodu kağıdın türünü ve tipini içeren bir **String** oluşturarak göndermektedir.
  - Bir **Card** nesnesinin string gösterimini oluşturmak için çağrılabilmektedir .
  - Nesnenin **String** gösterimi beklenliğinde üstü kapalı olarak çağrılabılır.



---

```
1 // Fig. 7.9: Card.java
2 // Card class represents a playing card.
3
4 public class Card
5 {
6     private final String face; // face of card ("Ace", "Deuce", ...)
7     private final String suit; // suit of card ("Hearts", "Diamonds", ...)
8
9     // two-argument constructor initializes card's face and suit
10    public Card(String cardFace, String cardSuit)
11    {
12        this.face = cardFace; // initialize face of card
13        this.suit = cardSuit; // initialize suit of card
14    }
15
16    // return String representation of Card
17    public String toString()
18    {
19        return face + " of " + suit;
20    }
21 } // end class Card
```

---

**Fig. 7.9** | Card class represents a playing card.

## 7.6 Case Study: Kağıt Karma and Simülasyon Oluşturma (Dvm)



- ▶ DeckofCards Sınıfı (Şekil 7.10) örnek değişken olarak `deck` adında bir Card dizisi tanımlamaktadır.
- ▶ `deck`'in elemanları varsayılan olarak `are null`'dır.
- ▶ Constructor `deck` array'ini Card nesneleri ile doldurur.



## 7.6 Case Study: Kağıt Karma and Simülasyon Oluşturma (Dvm)

- ▶ **shuffle** metodu **deck** içerisindeki **Card** nesnelerini karıştırmaktadır.
  - 52 **Card** içerisinde iterasyon ile gezilmektedir (array indis 0'dan 51'e kadardır).
  - Her **Card** rastgele seçilen başka bir kağıt ile swap edilmektedir.
- ▶ **dealCard** metodu array içerisindeki bir kağıt üzerinde çalışır.
  - **currentCard** üzerinde çalışılacak bir sonraki kağıdın indisini tutmaktadır.
  - Kağıt kalmazsa **null** döndürmektedir.



```
1 // Fig. 7.10: DeckOfCards.java
2 // DeckOfCards class represents a deck of playing cards.
3 import java.security.SecureRandom;
4
5 public class DeckOfCards
{
6
7     private Card[] deck; // array of Card objects
8     private int currentCard; // index of next Card to be dealt (0-51)
9     private static final int NUMBER_OF_CARDS = 52; // constant # of Cards
10    // random number generator
11    private static final SecureRandom randomNumbers = new SecureRandom();
12
13    // constructor fills deck of Cards
14    public DeckOfCards()
15    {
16        String[] faces = { "Ace", "Deuce", "Three", "Four", "Five", "Six",
17                          "Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King" };
18        String[] suits = { "Hearts", "Diamonds", "Clubs", "Spades" };
19
20        deck = new Card[NUMBER_OF_CARDS]; // create array of Card objects
21        currentCard = 0; // first Card dealt will be deck[0]
22    }
```

**Fig. 7.10** | DeckOfCards class represents a deck of playing cards. (Part 1 of 3.)



```
23     // populate deck with Card objects
24     for (int count = 0; count < deck.length; count++)
25         deck[count] =
26             new Card(faces[count % 13], suits[count / 13]);
27     }
28
29     // shuffle deck of Cards with one-pass algorithm
30     public void shuffle()
31     {
32         // next call to method dealCard should start at deck[0] again
33         currentCard = 0;
34
35         // for each Card, pick another random Card (0-51) and swap them
36         for (int first = 0; first < deck.length; first++)
37         {
38             // select a random number between 0 and 51
39             int second = randomNumbers.nextInt(NUMBER_OF_CARDS);
40
41             // swap current Card with randomly selected Card
42             Card temp = deck[first];
43             deck[first] = deck[second];
44             deck[second] = temp;
45         }
46     }
```

**Fig. 7.10** | DeckOfCards class represents a deck of playing cards. (Part 2 of 3.)



---

```
47
48     // deal one Card
49     public Card dealCard()
50     {
51         // determine whether Cards remain to be dealt
52         if (currentCard < deck.length)
53             return deck[currentCard++]; // return current Card in array
54         else
55             return null; // return null to indicate that all Cards were dealt
56     }
57 } // end class DeckOfCards
```

**Fig. 7.10** | DeckOfCards class represents a deck of playing cards. (Part 3 of 3.)

## 7.6 Case Study: Kağıt Karma and Simülasyon Oluşturma (Dvm)

- ▶ Figure 7.11'de DeckofCards sınıfı verilmiştir.



---

```
1 // Fig. 7.11: DeckOfCardsTest.java
2 // Card shuffling and dealing.
3
4 public class DeckOfCardsTest
5 {
6     // execute application
7     public static void main(String[] args)
8     {
9         DeckOfCards myDeckOfCards = new DeckOfCards();
10        myDeckOfCards.shuffle(); // place Cards in random order
11
12        // print all 52 Cards in the order in which they are dealt
13        for (int i = 1; i <= 52; i++)
14        {
15            // deal and display a Card
16            System.out.printf("%-19s", myDeckOfCards.dealCard());
17
18            if (i % 4 == 0) // output a newline after every fourth card
19                System.out.println();
20        }
21    }
22 } // end class DeckOfCardsTest
```

---

**Fig. 7.11** | Card shuffling and dealing. (Part I of 2.)



Six of Spades	Eight of Spades	Six of Clubs	Nine of Hearts
Queen of Hearts	Seven of Clubs	Nine of Spades	King of Hearts
Three of Diamonds	Deuce of Clubs	Ace of Hearts	Ten of Spades
Four of Spades	Ace of Clubs	Seven of Diamonds	Four of Hearts
Three of Clubs	Deuce of Hearts	Five of Spades	Jack of Diamonds
King of Clubs	Ten of Hearts	Three of Hearts	Six of Diamonds
Queen of Clubs	Eight of Diamonds	Deuce of Diamonds	Ten of Diamonds
Three of Spades	King of Diamonds	Nine of Clubs	Six of Hearts
Ace of Spades	Four of Diamonds	Seven of Hearts	Eight of Clubs
Deuce of Spades	Eight of Hearts	Five of Hearts	Queen of Spades
Jack of Hearts	Seven of Spades	Four of Clubs	Nine of Diamonds
Ace of Diamonds	Queen of Diamonds	Five of Clubs	King of Spades
Five of Diamonds	Ten of Clubs	Jack of Spades	Jack of Clubs

**Fig. 7.11** | Card shuffling and dealing. (Part 2 of 2.)

## 7.6 Case Study: Kağıt Karma and Simülasyon Oluşturma (Dvm)

### *NullPointerException hatalarını Önleme*

- ▶ Fig. 7.10'de, 52 Card referansına sahip bir deck arrayi bulunmaktadır—her eleman referans tipinde ve varsayılan değerleri null'dır.
- ▶ Bir sınıfın reference-tipi değişkeni olan alanları da varsayılan olarak null'dır.
- ▶ null reference'a sahip bir nesne üzerinden metod çağrırmaya çalışırsanız NullPointerException oluşmaktadır.
- ▶ .

## 7.7 Gelişmiş for Bildirimi

### ▶ Gelişmiş for bildirimi

- Bir sayaç kullanmadan bir dizinin öğeleri arasında geçiş yapar.
- Arrayin sınırlarının dışına çıkmayı önler.
- JAVA API'sinin diğer koleksiyonları ile de çalışır (sadece arrayler için değil).

### ▶ Sintaks:

```
for (parameter : arrayName)  
    statement
```

Parametre türü dizinin öğe tipiyle tutarlı olmalıdır.

Gelişmiş **for** bildirimi dizi üzerinde gezilmesini kolaylaştırır.



```
1 // Fig. 7.12: EnhancedForTest.java
2 // Using the enhanced for statement to total integers in an array.
3
4 public class EnhancedForTest
5 {
6     public static void main(String[] args)
7     {
8         int[] array = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
9         int total = 0;
10
11         // add each element's value to total
12         for (int number : array)
13             total += number;
14
15         System.out.printf("Total of array elements: %d%n", total);
16     }
17 } // end class EnhancedForTest
```

```
Total of array elements: 849
```

**Fig. 7.12** | Using the enhanced for statement to total integers in an array.

## 7.7 Gelişmiş for Bildirimi (Dvm.)

- ▶ Geliştirilmiş for deyimi yalnızca dizi elemanları üzerinde çalışmak için kullanılabilir
  - Öğeleri değiştirmek için kullanılamaz.
  - Öğeleri değiştirmek için geleneksel karşı kontrollü ifadeyi kullanın.
- Elemanın indeksine erişmeniz gerekmiyorsa, sayıç kontrollü for döngüsü yerine kullanılabilir.

## 7.8 Arrayleri Metotlara Gönderme

- ▶ Bir array argümanını bir metoda göndermek için, arrayin adını köşeli parantez olmadan belirtin.
  - Her array nesnesi kendi uzunluğunu “bildiğinden”, dizi uzunluğunu ek bir argüman olarak geçmemize gerek yoktur.
  - Bir dizi parametresi almak için, metodun parametre listesi bir dizi parametresi belirtmelidir.
- Bir metodun aldığı bir argüman, bir array ya da referans türünde tek bir array elamanı olduğunda, çağrılan yöntem, referansın bir kopyasını alır.
- ▶ Bir metoda gönderilen bir argüman, tek bir dizi ögesinin ilkel bir türü olduğunda, çağrılan yöntem, ögenin değerinin bir kopyasını alır.
  - Bu ilkel değerlere scalar denilmektedir.



```
1 // Fig. 7.13: PassArray.java
2 // Passing arrays and individual array elements to methods.
3
4 public class PassArray
5 {
6     // main creates array and calls modifyArray and modifyElement
7     public static void main(String[] args)
8     {
9         int[] array = { 1, 2, 3, 4, 5 };
10
11     System.out.printf(
12         "Effects of passing reference to entire array:%n" +
13         "The values of the original array are:%n");
14
15     // output original array elements
16     for (int value : array)
17         System.out.printf("    %d", value);
18
19     modifyArray(array); // pass array reference
20     System.out.printf("%n%nThe values of the modified array are:%n");
21 }
```

**Fig. 7.13** | Passing arrays and individual array elements to methods. (Part I of 3.)



```
22     // output modified array elements
23     for (int value : array)
24         System.out.printf("    %d", value);
25
26     System.out.printf(
27         "%n%nEffects of passing array element value:%n" +
28         "array[3] before modifyElement: %d%n", array[3]);
29
30     modifyElement(array[3]); // attempt to modify array[3]
31     System.out.printf(
32         "array[3] after modifyElement: %d%n", array[3]);
33 }
34
35 // multiply each element of an array by 2
36 public static void modifyArray(int[] array2)
37 {
38     for (int counter = 0; counter < array2.length; counter++)
39         array2[counter] *= 2;
40 }
```

**Fig. 7.13** | Passing arrays and individual array elements to methods. (Part 2 of 3.)



```
41
42 // multiply argument by 2
43 public static void modifyElement(int element)
44 {
45     element *= 2;
46     System.out.printf(
47         "Value of element in modifyElement: %d%n", element);
48 }
49 } // end class PassArray
```

Effects of passing reference to entire array:

The values of the original array are:

1 2 3 4 5

The values of the modified array are:

2 4 6 8 10

Effects of passing array element value:

array[3] before modifyElement: 8

Value of element in modifyElement: 16

array[3] after modifyElement: 8

**Fig. 7.13** | Passing arrays and individual array elements to methods. (Part 3 of 3.)

## 7.9 Pass-By-Value ve. Pass-By-Reference

- ▶ Pass-by-value (bazen **call-by-value**)
  - Argümanın değerinin bir kopyası metoda gönderilmektedir.
  - Çağırılan metot kopya üzerinde özel olarak çalışmaktadır.
  - Çağrılan yöntemin parametre üzerinde gerçekleştirdiği değişiklikler, çağrıran yerdeki orijinal değişkenin değerini etkilemez.
- ▶ Pass-by-reference (bazen **call-by-reference**)
  - Çağrılan metot, çağrıranın argümanın değerine doğrudan erişebilir ve gerekirse bu verileri değiştirebilir.
  - Performansı arttırmak verilerin kopyalanmasını azaltmak için uygundur.

## 7.9 Pass-By-Value ve Pass-By-Reference (Dvm.)

- ▶ Java'daki tüm argümanlar pass-by-value ile geçirilir..
- ▶ Bir yöntem çağrısı, bir yönteme iki tür değeri iletebilir.
  - İlkel tiplerin kopyası
  - Nesnelerin referans tipleri
- ▶ Bir yöntem, bir referans türü parametresini başka bir nesneyi ifade edecek şekilde değiştirirse, yalnızca parametre yeni nesneye işaret eder orijinal argümanın gösterdiği referans değişmez.
- ▶ Bir nesnenin referansı pass by value'e göre aktarılısa da, bir metot, nesnenin referansının kopyasını kullanarak public metodlarını çağırarak başvurulan nesneyi değiştirebilir.