Chapter 7
Arrays and ArrayLists

Java™ How to Program, 10/e

7.10 Vaka Analizi: Class GradeBook Notları depolamak için Array kullanmak

- Vaka incelememizin ilk bölümününde, öğretim elemanlarının sınav notlarını kaydetmek için kullanabilecekleri ve notları, sınıf ortalamasını, en düşük notu, en yüksek notu ve not dağılımını içeren bir not raporu sergileyen bir GradeBook sınıfını inceleyeceğiz.

- `GradeBook` sınıfının bu versiyonu tek boyutlu bir dizide bir sınava ait notları saklar.
- In Section 7.12'de `GradeBook` sınıfının iki boyutlu array kullanan halini de inceleyeceğiz.

```
1  // Fig. 7.14: GradeBook.java
2  // GradeBook class using an array to store test grades.
3
4  public class GradeBook
5  {
6     private String courseName; // name of course this GradeBook represents
7     private int[] grades; // array of student grades
8
9     // constructor
10    public GradeBook(String courseName, int[] grades)
11    {
12       this.courseName = courseName;
13       this.grades = grades;
14    }
15
16    // method to set the course name
17    public void setCourseName(String courseName)
18    {
19       this.courseName = courseName;
20    }
21
```

Fig. 7.14 | GradeBook class using an array to store test grades. (Part 1 of 7.)

```
22      // method to retrieve the course name
23      public String getCourseName()
24      {
25         return courseName;
26      }
27
28      // perform various operations on the data
29      public void processGrades()
30      {
31         // output grades array
32         outputGrades();
33
34         // call method getAverage to calculate the average grade
35         System.out.printf("%nClass average is %.2f%n", getAverage());
36
37         // call methods getMinimum and getMaximum
38         System.out.printf("Lowest grade is %d%nHighest grade is %d%n%n",
39            getMinimum(), getMaximum());
40
41         // call outputBarChart to print grade distribution chart
42         outputBarChart();
43      }
44
```

Fig. 7.14 | GradeBook class using an array to store test grades. (Part 2 of 7.)

```
45      // find minimum grade
46      public int getMinimum()
47      {
48         int lowGrade = grades[0]; // assume grades[0] is smallest
49
50         // loop through grades array
51         for (int grade : grades)
52         {
53            // if grade lower than lowGrade, assign it to lowGrade
54            if (grade < lowGrade)
55               lowGrade = grade; // new lowest grade
56         }
57
58         return lowGrade;
59      }
60
```

Fig. 7.14 | GradeBook class using an array to store test grades. (Part 3 of 7.)

```
61      // find maximum grade
62      public int getMaximum()
63      {
64         int highGrade = grades[0]; // assume grades[0] is largest
65
66         // loop through grades array
67         for (int grade : grades)
68         {
69            // if grade greater than highGrade, assign it to highGrade
70            if (grade > highGrade)
71               highGrade = grade; // new highest grade
72         }
73
74         return highGrade;
75      }
76
```

Fig. 7.14 | GradeBook class using an array to store test grades. (Part 4 of 7.)

```
77      // determine average grade for test
78      public double getAverage()
79      {
80         int total = 0;
81
82         // sum grades for one student
83         for (int grade : grades)
84            total += grade;
85
86         // return average of grades
87         return (double) total / grades.length;
88      }
89
```

Fig. 7.14 | GradeBook class using an array to store test grades. (Part 5 of 7.)

```
90      // output bar chart displaying grade distribution
91      public void outputBarChart()
92      {
93         System.out.println("Grade distribution:");
94
95         // stores frequency of grades in each range of 10 grades
96         int[] frequency = new int[11];
97
98         // for each grade, increment the appropriate frequency
99         for (int grade : grades)
100           ++frequency[grade / 10];
101
102        // for each grade frequency, print bar in chart
103        for (int count = 0; count < frequency.length; count++)
104        {
105           // output bar label ("00-09: ", ..., "90-99: ", "100: ")
106           if (count == 10)
107              System.out.printf("%5d: ", 100);
108           else
109              System.out.printf("%02d-%02d: ",
110                 count * 10, count * 10 + 9);
111
```

Fig. 7.14 | GradeBook class using an array to store test grades. (Part 6 of 7.)

```
112            // print bar of asterisks
113            for (int stars = 0; stars < frequency[count]; stars++)
114               System.out.print("*");
115
116            System.out.println();
117         }
118      }
119
120      // output the contents of the grades array
121      public void outputGrades()
122      {
123         System.out.printf("The grades are:%n%n");
124
125         // output each student's grade
126         for (int student = 0; student < grades.length; student++)
127            System.out.printf("Student %2d: %3d%n",
128               student + 1, grades[student]);
129      }
130   } // end class GradeBook
```

Fig. 7.14 | GradeBook class using an array to store test grades. (Part 7 of 7.)

7.10 Vaka Analizi: Class GradeBook Notları depolamak için Array kullanmak (Dvm.)

- Fig. 7.15'te `GradeBook` (Fig. 7.14) sınıfının bir nesnesi, dersin `String` ders adı ve `int` array `grades-Array` kullanılarak yaratılmaktadır (12,13. satır).

```
1  // Fig. 7.15: GradeBookTest.java
2  // GradeBookTest creates a GradeBook object using an array of grades,
3  // then invokes method processGrades to analyze them.
4  public class GradeBookTest
5  {
6     // main method begins program execution
7     public static void main(String[] args)
8     {
9        // array of student grades
10       int[] gradesArray = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
11
12       GradeBook myGradeBook = new GradeBook(
13          "CS101 Introduction to Java Programming", gradesArray);
14       System.out.printf("Welcome to the grade book for%n%s%n%n",
15          myGradeBook.getCourseName());
16       myGradeBook.processGrades();
17    }
18 } // end class GradeBookTest
```

Fig. 7.15 | GradeBookTest creates a GradeBook object using an array of grades, then invokes method processGrades to analyze them. (Part 1 of 3.)

```
Welcome to the grade book for
CS101 Introduction to Java Programming

The grades are:

Student  1:  87
Student  2:  68
Student  3:  94
Student  4: 100
Student  5:  83
Student  6:  78
Student  7:  85
Student  8:  91
Student  9:  76
Student 10:  87

Class average is 84.90
Lowest grade is 68
Highest grade is 100
```

Fig. 7.15 | `GradeBookTest` creates a `GradeBook` object using an array of grades, then invokes method `processGrades` to analyze them. (Part 2 of 3.)

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

Fig. 7.15 | `GradeBookTest` creates a `GradeBook` object using an array of grades, then invokes method `processGrades` to analyze them. (Part 3 of 3.)

7.10 Vaka Analizi: Class GradeBook Notları depolamak için Array kullanmak (Dvm.)

Java SE 8

- Bölüm 17'de Java SE 8 Lambda ve Streamleri kulllanarak array üzerinde daha hızlı bir şekilde işlemlerin gerçekleştirildiğini inceleyebilirsiniz

7.11 Çok Boyutlu Arrayler

- İki boyutlu diziler genellikle, satır ve sütunlarda düzenlenmiş veriler içeren değer tablolarını temsil etmek için kullanılır.
- Her tablo elemanını iki endeksle tanımlanır
  - İlk indeks satır ikinci indeks sütun olacak sekilde
- Çok boyutlu diziler ikiden fazla boyuta sahip olabilir.
-
- Genelde, m satırlı ve n sütunlu bir diziye m-by-n dizisi denir.

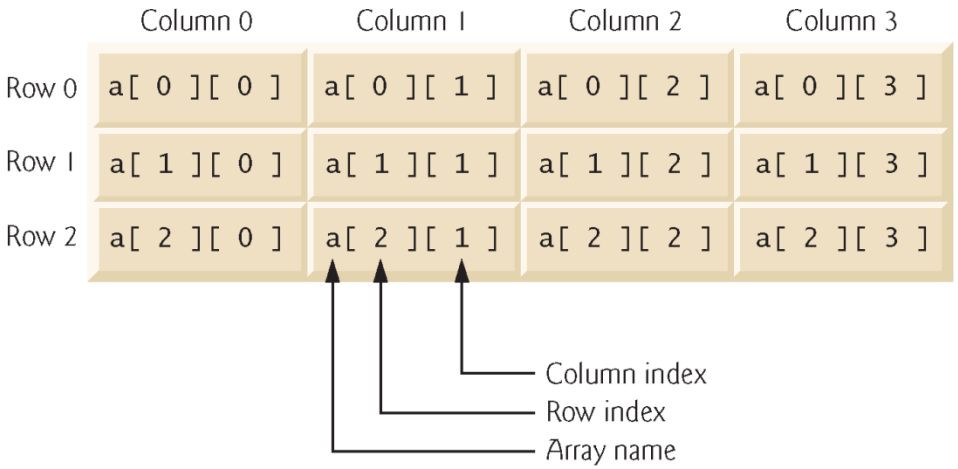

Fig. 7.16 | Two-dimensional array with three rows and four columns.

7.11 Çok Boyutlu Arrayler (Dvm.)

- Çok boyutlu diziler, dizi başlatıcılarıyla (array initializers) tanımlanıp oluşturulabilir.
- İki satır ve iki sütun içeren iki boyutlu bir dizi b, aşağıdaki gibi nested dizi başlatıcılarıyla bildirilebilir ve başlatılabilir: `int[][] b = {{1, 2}, {3, 4}};`
  - İlk değerler, parantez içinde satır ile gruplandırılmıştır.
  - Nested dizi başlatıcılarının sayısı (dış parantez içindeki parantez kümeleriyle temsil edilir) satırların sayısını belirler.
  - Bir satır için ilgili dizi başlatıcı içindeki eleman sayısı, o satırdaki sütun sayısını belirler.
  - *Satırların farklı uzunlukları olabilir.*

7.11 Çok Boyutlu Arrayler (Dvm.)

- İki boyutlu bir dizideki satırların uzunluklarının aynı olması gerekmez:
- ```
  int[][] b = {{1, 2}, {3, 4, 5}};
  ```
  - `b` 'nin her elemanı, tek boyutlu bir `int` değişken dizisine referanstır.
  - Satır 0 için `int` dizisi iki öğeye sahip (1 ve 2) tek boyutlu bir dizidir.
  - Satır 1 için `int` dizisi üç öğeyle (3, 4 ve 5) tek boyutlu bir dizidir.
  ```

7.11 Çok Boyutlu Arrayler (Dvm.)

- Her satırda aynı sayıda sütuna sahip çok boyutlu bir dizi, dizi oluşturma ifadesiyle oluşturulabilir.

```
int[][] b = new int[3][4];
```

  - 3 satır ve 4 sütun.
- Dizi nesnesi oluşturulduğunda çok boyutlu bir dizinin öğeleri ilklenir.
- Her satırın farklı sayıda sütun içerdiği çok boyutlu bir dizi aşağıdaki gibi oluşturulabilir:
-

```
int[][] b = new int[2][];   // create 2 rows
b[0] = new int[5]; // create 5 columns for row 0
b[1] = new int[3]; // create 3 columns for row 1
```

  - İki satırlı iki boyutlu bir dizi oluşturur.
  - Satır 0, beş sütuna sahiptir ve 1. satırda üç sütun vardır.

7.11 Çok Boyutlu Arrayler (Dvm.)

- Şekil 7.17, dizi başlatıcıları ile iki boyutlu dizileri oluşturmayı ve dizileri gezmek için için iç içe girmiş `for` kullanmayı göstermektedir.

```
1  // Fig. 7.17: InitArray.java
2  // Initializing two-dimensional arrays.
3
4  public class InitArray
5  {
6     // create and output two-dimensional arrays
7     public static void main(String[] args)
8     {
9        int[][] array1 = {{1, 2, 3}, {4, 5, 6}};
10       int[][] array2 = {{1, 2}, {3}, {4, 5, 6}};
11
12       System.out.println("Values in array1 by row are");
13       outputArray(array1); // displays array1 by row
14
15       System.out.printf("%nValues in array2 by row are%n");
16       outputArray(array2); // displays array2 by row
17    }
18
```

Fig. 7.17 | Initializing two-dimensional arrays. (Part 1 of 2.)

```
19      // output rows and columns of a two-dimensional array
20      public static void outputArray(int[][] array)
21      {
22         // loop through array's rows
23         for (int row = 0; row < array.length; row++)
24         {
25            // loop through columns of current row
26            for (int column = 0; column < array[row].length; column++)
27               System.out.printf("%d  ", array[row][column]);
28
29            System.out.println();
30         }
31      }
32   } // end class InitArray
```

```
Values in array1 by row are
1  2  3
4  5  6

Values in array2 by row are
1  2
3
4  5  6
```

Fig. 7.17 | Initializing two-dimensional arrays. (Part 2 of 2.)

7.12 7.10 Vaka Analizi: Class GradeBook Notları depolamak için İki Boyutulu Array kullanmak

- Çoğu yarıyılda, bir derste öğrenciler çeşitli sınavlardan geçerler.
- Şekil 7.18, öğrencilerin çoklu sınavlardan aldığı notları saklamak için iki boyutlu bir dizi notu kullanan bir sınıf `GradeBook` sınıfı sürümünü içerir.
  - Her satır, tüm ders için bir öğrencinin notlarını temsil eder.
  - Bu örnekte, üç sınavda on öğrencinin notlarını içeren on üçlük bir dizi kullanıyoruz.

```
1  // Fig. 7.18: GradeBook.java
2  // GradeBook class using a two-dimensional array to store grades.
3
4  public class GradeBook
5  {
6     private String courseName; // name of course this grade book represents
7     private int[][] grades; // two-dimensional array of student grades
8
9     // two-argument constructor initializes courseName and grades array
10    public GradeBook(String courseName, int[][] grades)
11    {
12       this.courseName = courseName;
13       this.grades = grades;
14    }
15
16    // method to set the course name
17    public void setCourseName(String courseName)
18    {
19       this.courseName = courseName;
20    }
21
```

Fig. 7.18 | GradeBook class using a two-dimensional array to store grades. (Part 1 of 8.)

```
22      // method to retrieve the course name
23      public String getCourseName()
24      {
25         return courseName;
26      }
27
28      // perform various operations on the data
29      public void processGrades()
30      {
31         // output grades array
32         outputGrades();
33
34         // call methods getMinimum and getMaximum
35         System.out.printf("%n%s %d%n%s %d%n%n",
36            "Lowest grade in the grade book is", getMinimum(),
37            "Highest grade in the grade book is", getMaximum());
38
39         // output grade distribution chart of all grades on all tests
40         outputBarChart();
41      }
42
```

Fig. 7.18 | GradeBook class using a two-dimensional array to store grades. (Part 2 of 8.)

```
43      // find minimum grade
44      public int getMinimum()
45      {
46         // assume first element of grades array is smallest
47         int lowGrade = grades[0][0];
48
49         // loop through rows of grades array
50         for (int[] studentGrades : grades)
51         {
52            // loop through columns of current row
53            for (int grade : studentGrades)
54            {
55               // if grade less than lowGrade, assign it to lowGrade
56               if (grade < lowGrade)
57                  lowGrade = grade;
58            }
59         }
60
61         return lowGrade;
62      }
63
```

Fig. 7.18 | GradeBook class using a two-dimensional array to store grades. (Part 3 of 8.)

```
64      // find maximum grade
65      public int getMaximum()
66      {
67         // assume first element of grades array is largest
68         int highGrade = grades[0][0];
69
70         // loop through rows of grades array
71         for (int[] studentGrades : grades)
72         {
73            // loop through columns of current row
74            for (int grade : studentGrades)
75            {
76               // if grade greater than highGrade, assign it to highGrade
77               if (grade > highGrade)
78                  highGrade = grade;
79            }
80         }
81
82         return highGrade;
83      }
84
```

Fig. 7.18 | GradeBook class using a two-dimensional array to store grades. (Part 4 of 8.)

```
85      // determine average grade for particular set of grades
86      public double getAverage(int[] setOfGrades)
87      {
88         int total = 0;
89
90         // sum grades for one student
91         for (int grade : setOfGrades)
92            total += grade;
93
94         // return average of grades
95         return (double) total / setOfGrades.length;
96      }
97
98      // output bar chart displaying overall grade distribution
99      public void outputBarChart()
100     {
101        System.out.println("Overall grade distribution:");
102
103        // stores frequency of grades in each range of 10 grades
104        int[] frequency = new int[11];
105
```

Fig. 7.18 | GradeBook class using a two-dimensional array to store grades. (Part 5 of 8.)

```
106         // for each grade in GradeBook, increment the appropriate frequency
107         for (int[] studentGrades : grades)
108         {
109            for (int grade : studentGrades)
110               ++frequency[grade / 10];
111         }
112
113         // for each grade frequency, print bar in chart
114         for (int count = 0; count < frequency.length; count++)
115         {
116            // output bar label ("00-09: ", ..., "90-99: ", "100: ")
117            if (count == 10)
118               System.out.printf("%5d: ", 100);
119            else
120               System.out.printf("%02d-%02d: ",
121                  count * 10, count * 10 + 9);
122
123            // print bar of asterisks
124            for (int stars = 0; stars < frequency[count]; stars++)
125               System.out.print("*");
126
127            System.out.println();
128         }
129      }
```

Fig. 7.18 | GradeBook class using a two-dimensional array to store grades. (Part 6 of 8.)

```
130
131    // output the contents of the grades array
132    public void outputGrades()
133    {
134       System.out.printf("The grades are:%n%n");
135       System.out.print("            "); // align column heads
136
137       // create a column heading for each of the tests
138       for (int test = 0; test < grades[0].length; test++)
139          System.out.printf("Test %d  ", test + 1);
140
141       System.out.println("Average"); // student average column heading
142
143       // create rows/columns of text representing array grades
144       for (int student = 0; student < grades.length; student++)
145       {
146          System.out.printf("Student %2d", student + 1);
147
148          for (int test : grades[student]) // output student's grades
149             System.out.printf("%8d", test);
150
```

Fig. 7.18 | GradeBook class using a two-dimensional array to store grades. (Part 7 of 8.)

```
151             // call method getAverage to calculate student's average grade;
152             // pass row of grades as the argument to getAverage
153             double average = getAverage(grades[student]);
154             System.out.printf("%9.2f%n", average);
155          }
156       }
157    } // end class GradeBook
```

Fig. 7.18 | GradeBook class using a two-dimensional array to store grades. (Part 8 of 8.)

```
1  // Fig. 7.19: GradeBookTest.java
2  // GradeBookTest creates GradeBook object using a two-dimensional array
3  // of grades, then invokes method processGrades to analyze them.
4  public class GradeBookTest
5  {
6     // main method begins program execution
7     public static void main(String[] args)
8     {
9        // two-dimensional array of student grades
10       int[][] gradesArray = {{87, 96, 70},
11                              {68, 87, 90},
12                              {94, 100, 90},
13                              {100, 81, 82},
14                              {83, 65, 85},
15                              {78, 87, 65},
16                              {85, 75, 83},
17                              {91, 94, 100},
18                              {76, 72, 84},
19                              {87, 93, 73}};
20
```

Fig. 7.19 | `GradeBookTest` creates `GradeBook` object using a two-dimensional array of grades, then invokes method `processGrades` to analyze them. (Part 1 of 4.)

```
21          GradeBook myGradeBook = new GradeBook(
22             "CS101 Introduction to Java Programming", gradesArray);
23          System.out.printf("Welcome to the grade book for%n%s%n%n",
24             myGradeBook.getCourseName());
25          myGradeBook.processGrades();
26       }
27    } // end class GradeBookTest
```

Fig. 7.19 | `GradeBookTest` creates `GradeBook` object using a two-dimensional array of grades, then invokes method `processGrades` to analyze them. (Part 2 of 4.)

```
Welcome to the grade book for
CS101 Introduction to Java Programming

The grades are:

            Test 1  Test 2  Test 3  Average
Student  1      87      96      70    84.33
Student  2      68      87      90    81.67
Student  3      94     100      90    94.67
Student  4     100      81      82    87.67
Student  5      83      65      85    77.67
Student  6      78      87      65    76.67
Student  7      85      75      83    81.00
Student  8      91      94     100    95.00
Student  9      76      72      84    77.33
Student 10      87      93      73    84.33

Lowest grade in the grade book is 65
Highest grade in the grade book is 100
```

Fig. 7.19 | `GradeBookTest` creates `GradeBook` object using a two-dimensional array of grades, then invokes method `processGrades` to analyze them. (Part 3 of 4.)

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

Fig. 7.19 | `GradeBookTest` creates `GradeBook` object using a two-dimensional array of grades, then invokes method `processGrades` to analyze them. (Part 4 of 4.)

7.13 Değişken Boyutlu Argüman Listesi

- Değişken Boyutlu Argüman Listesi
- Önceden belirlenmemiş sayıda argüman alan metotlar oluşturmak için kullanılabilir.
- Bir (…) ile takip edilen parametre tipi, metodun söz konusu tipte değişken sayıda argüman aldığını gösterir.
  - Üç nokta bir parametre listesinin sadece sonunda ve sadece bir kez oluşabilir.

```
1  // Fig. 7.20: VarargsTest.java
2  // Using variable-length argument lists.
3
4  public class VarargsTest
5  {
6     // calculate average
7     public static double average(double... numbers)
8     {
9        double total = 0.0;
10
11       // calculate total using the enhanced for statement
12       for (double d : numbers)
13          total += d;
14
15       return total / numbers.length;
16    }
17
18    public static void main(String[] args)
19    {
20       double d1 = 10.0;
21       double d2 = 20.0;
22       double d3 = 30.0;
23       double d4 = 40.0;
24
```

Fig. 7.20 | Using variable-length argument lists. (Part 1 of 2.)

```
25          System.out.printf("d1 = %.1f%nd2 = %.1f%nd3 = %.1f%nd4 = %.1f%n%n",
26             d1, d2, d3, d4);
27
28          System.out.printf("Average of d1 and d2 is %.1f%n",
29             average(d1, d2)  );
30          System.out.printf("Average of d1, d2 and d3 is %.1f%n",
31             average(d1, d2, d3)  );
32          System.out.printf("Average of d1, d2, d3 and d4 is %.1f%n",
33             average(d1, d2, d3, d4)  );
34       }
35    } // end class VarargsTest
```

```
d1 = 10.0
d2 = 20.0
d3 = 30.0
d4 = 40.0

Average of d1 and d2 is 15.0
Average of d1, d2 and d3 is 20.0
Average of d1, d2, d3 and d4 is 25.0
```

Fig. 7.20 | Using variable-length argument lists. (Part 2 of 2.)

7.14 Komut Satırı Argümanlarını Kullanmak

- Java komutunda sınıf adından sonra görünen komut satırı argümanları, main metodunda String dizisi args tarafından alınır.
- Komut satırı argümanlarının sayısı dizinin lenth özelliğine erişerek elde edilir.
- Komut satırı argümanları, virgül değil, beyaz boşlukla ayrılır.

```
1  // Fig. 7.21: InitArray.java
2  // Initializing an array using command-line arguments.
3
4  public class InitArray
5  {
6     public static void main(String[] args)
7     {
8        // check number of command-line arguments
9        if (args.length != 3)
10          System.out.printf(
11             "Error: Please re-enter the entire command, including%n" +
12             "an array size, initial value and increment.%n");
13       else
14       {
15          // get array size from first command-line argument
16          int arrayLength = Integer.parseInt(args[0]);
17          int[] array = new int[arrayLength];
18
19          // get initial value and increment from command-line arguments
20          int initialValue = Integer.parseInt(args[1]);
21          int increment = Integer.parseInt(args[2]);
22
```

Fig. 7.21 | Initializing an array using command-line arguments. (Part 1 of 3.)

```
23            // calculate value for each array element
24            for (int counter = 0; counter < array.length; counter++)
25               array[counter] = initialValue + increment * counter;
26
27            System.out.printf("%s%8s%n", "Index", "Value");
28
29            // display array index and value
30            for (int counter = 0; counter < array.length; counter++)
31               System.out.printf("%5d%8d%n", counter, array[counter]);
32         }
33      }
34   } // end class InitArray
```

```
java InitArray
Error: Please re-enter the entire command, including
an array size, initial value and increment.
```

Fig. 7.21 | Initializing an array using command-line arguments. (Part 2 of 3.)

```
java InitArray 5 0 4
Index   Value
    0       0
    1       4
    2       8
    3      12
    4      16
```

```
java InitArray 8 1 2
Index   Value
    0       1
    1       3
    2       5
    3       7
    4       9
    5      11
    6      13
    7      15
```

Fig. 7.21 | Initializing an array using command-line arguments. (Part 3 of 3.)

7.15 Class Arrays

- `Arrays` sınıfı
  - Genel dizi manipülasyonları için statik yöntemler sağlar.
- Metotlar:
  - `sort` Bir diziyi sıralamak için (varsayılan olarak artan düzen)
  - `binarySearch` sıralanmış bir diziyi aramak için
  - `equals` dizileri karşılaştırmak için
  - `fill` değerleri bir diziye yerleştirmek için.
  - İlkel-tipi diziler için ve nesnelerin dizileri için metotlar overload edilmiştir
  - `System` sınıfının `static` `arraycopy` metodu
  - Bir dizinin içeriğini diğerine kopyalar.

```
1  // Fig. 7.22: ArrayManipulations.java
2  // Arrays class methods and System.arraycopy.
3  import java.util.Arrays;
4
5  public class ArrayManipulations
6  {
7     public static void main(String[] args)
8     {
9        // sort doubleArray into ascending order
10       double[] doubleArray = { 8.4, 9.3, 0.2, 7.9, 3.4 };
11       Arrays.sort(doubleArray);
12       System.out.printf("%ndoubleArray: ");
13
14       for (double value : doubleArray)
15          System.out.printf("%.1f ", value);
16
17       // fill 10-element array with 7s
18       int[] filledIntArray = new int[10];
19       Arrays.fill(filledIntArray, 7);
20       displayArray(filledIntArray, "filledIntArray");
21
```

Fig. 7.22 | `Arrays` class methods and `System.arraycopy`. (Part 1 of 4.)

```
22          // copy array intArray into array intArrayCopy
23          int[] intArray = { 1, 2, 3, 4, 5, 6 };
24          int[] intArrayCopy = new int[intArray.length];
25          System.arraycopy(intArray, 0, intArrayCopy, 0, intArray.length);
26          displayArray(intArray, "intArray");
27          displayArray(intArrayCopy, "intArrayCopy");
28
29          // compare intArray and intArrayCopy for equality
30          boolean b = Arrays.equals(intArray, intArrayCopy);
31          System.out.printf("%n%nintArray %s intArrayCopy%n",
32             (b ? "==" : "!="));
33
34          // compare intArray and filledIntArray for equality
35          b = Arrays.equals(intArray, filledIntArray);
36          System.out.printf("intArray %s filledIntArray%n",
37             (b ? "==" : "!="));
38
39          // search intArray for the value 5
40          int location = Arrays.binarySearch(intArray, 5);
41
42          if (location >= 0)
43             System.out.printf(
44                "Found 5 at element %d in intArray%n", location);
45          else
46             System.out.println("5 not found in intArray");
```

Fig. 7.22 | `Arrays` class methods and `System.arraycopy`. (Part 2 of 4.)

```
47
48          // search intArray for the value 8763
49          location = Arrays.binarySearch(intArray, 8763);
50
51          if (location >= 0)
52             System.out.printf(
53                "Found 8763 at element %d in intArray%n", location);
54          else
55             System.out.println("8763 not found in intArray");
56       }
57
58       // output values in each array
59       public static void displayArray(int[] array, String description)
60       {
61          System.out.printf("%n%s: ", description);
62
63          for (int value : array)
64             System.out.printf("%d ", value);
65       }
66    } // end class ArrayManipulations
```

Fig. 7.22 | `Arrays` class methods and `System.arraycopy`. (Part 3 of 4.)

```
doubleArray: 0.2 3.4 7.9 8.4 9.3
filledIntArray: 7 7 7 7 7 7 7 7 7 7
intArray: 1 2 3 4 5 6
intArrayCopy: 1 2 3 4 5 6

intArray == intArrayCopy
intArray != filledIntArray
Found 5 at element 4 in intArray
8763 not found in intArray
```

Fig. 7.22 | `Arrays` class methods and `System.arraycopy`. (Part 4 of 4.)

Bilgi

Array içeriklerinin eşitlikleri karşılaştırılırken Arrays.equals arraylerin elemanlarının değerlerini kontrol eder. Array1.equals(array2) ise arraylerin içeriklerinden ziyade ikisinin de bellekte aynı nesneyi gösterip göstermediğine bakar.

7.16 ArrayList

- Diziler, kapasitesinin dışındaki ek elemanları barındırmak için yürütme süresinde boyutlarını otomatik olarak değiştirmez.
- ArrayList<T>(paket java.util), boyutlarını daha fazla öğeye uyacak şekilde dinamik olarak değiştirebilir.
  - T depolanacak verinin tipi için yer tutucudur.
- Bu şekilde herhangi bir sınıf ile kullanlıabilecek yer turuculara generic classes.

Method	Description
`add`	Adds an element to the *end* of the `ArrayList`.
`clear`	Removes all the elements from the `ArrayList`.
`contains`	Returns `true` if the `ArrayList` contains the specified element; otherwise, returns `false`.
`get`	Returns the element at the specified index.
`indexOf`	Returns the index of the first occurrence of the specified element in the `ArrayList`.
`remove`	Overloaded. Removes the first occurrence of the specified value or the element at the specified index.
`size`	Returns the number of elements stored in the `ArrayList`.
`trimToSize`	Trims the capacity of the `ArrayList` to the current number of elements.

Fig. 7.23 | Some methods and properties of class `ArrayList<T>`.

7.16 ArrayList (Dvm.)

- Şekil 7.24, bazı önemli ArrayList metotlarının kullanınımını göstermektedir.
- Bir ArrayList'in kapasitesi, büyümeden kaç öğe tutabileceğini gösterir.
- ArrayList büyüdüğünde, daha büyük bir iç dizi oluşturmalı ve her öğeyi yeni diziye kopyalamalıdır.
- Bu zaman alıcı bir işlemdir. Bir öğe eklendiğinde ArrayList'in büyümesi performans açısından verimsizolurdu.
- Bir ArrayList yalnızca bir eleman eklendiğinde ve elemanların sayısı kapasiteye eşit olduğunda büyür - yani, yeni eleman için yer yoksa.

7.16 Introduction to Collections and Class `ArrayList` (Cont.)

- Method `add` adds elements to the `ArrayList`.
  - One-argument version appends its argument to the end of the `ArrayList`.
  - Two-argument version inserts a new element at the specified position.
  - Collection indices start at zero.
- Method `size` returns the number of elements in the `ArrayList`.
- Method `get` obtains the element at a specified index.
- Method `remove` deletes an element with a specific value.
  - An overloaded version of the method removes the element at the specified index.
- Method `contains` determines if an item is in the `ArrayList`.

```
1  // Fig. 7.24: ArrayListCollection.java
2  // Generic ArrayList<T> collection demonstration.
3  import java.util.ArrayList;
4
5  public class ArrayListCollection
6  {
7     public static void main(String[] args)
8     {
9        // create a new ArrayList of Strings with an initial capacity of 10
10       ArrayList<String> items = new ArrayList<String>();
11
12       items.add("red"); // append an item to the list
13       items.add(0, "yellow"); // insert "yellow" at index 0
14
15       // header
16       System.out.print(
17          "Display list contents with counter-controlled loop:");
18
19       // display the colors in the list
20       for (int i = 0; i < items.size(); i++)
21          System.out.printf(" %s", items.get(i));
22
```

Fig. 7.24 | Generic `ArrayList<T>` collection demonstration. (Part 1 of 3.)

```
23          // display colors using enhanced for in the display method
24          display(items,
25             "%nDisplay list contents with enhanced for statement:");
26
27          items.add("green"); // add "green" to the end of the list
28          items.add("yellow"); // add "yellow" to the end of the list
29          display(items, "List with two new elements:");
30
31          items.remove("yellow"); // remove the first "yellow"
32          display(items, "Remove first instance of yellow:");
33
34          items.remove(1); // remove item at index 1
35          display(items, "Remove second list element (green):");
36
37          // check if a value is in the List
38          System.out.printf("\"red\" is %sin the list%n",
39             items.contains("red") ? "": "not ");
40
41          // display number of elements in the List
42          System.out.printf("Size: %s%n", items.size());
43       }
44
```

Fig. 7.24 | Generic `ArrayList<T>` collection demonstration. (Part 2 of 3.)

```
45      // display the ArrayList's elements on the console
46      public static void display(ArrayList<String> items, String header)
47      {
48         System.out.printf(header); // display header
49
50         // display each element in items
51         for (String item : items)
52            System.out.printf(" %s", item);
53
54         System.out.println();
55      }
56   } // end class ArrayListCollection
```

```
Display list contents with counter-controlled loop: yellow red
Display list contents with enhanced for statement: yellow red
List with two new elements: yellow red green yellow
Remove first instance of yellow: red green yellow
Remove second list element (green): red yellow
"red" is in the list
Size: 2
```

Fig. 7.24 | Generic `ArrayList<T>` collection demonstration. (Part 3 of 3.)

```
1  // Fig. 7.25: DrawRainbow.java
2  // Drawing a rainbow using arcs and an array of colors.
3  import java.awt.Color;
4  import java.awt.Graphics;
5  import javax.swing.JPanel;
6
7  public class DrawRainbow extends JPanel
8  {
9     // define indigo and violet
10    private final static Color VIOLET = new Color(128, 0, 128);
11    private final static Color INDIGO = new Color(75, 0, 130);
12
13    // colors to use in the rainbow, starting from the innermost
14    // The two white entries result in an empty arc in the center
15    private Color[] colors =
16       { Color.WHITE, Color.WHITE, VIOLET, INDIGO, Color.BLUE,
17         Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
18
19    // constructor
20    public DrawRainbow()
21    {
22       setBackground(Color.WHITE); // set the background to white
23    }
24
```

Fig. 7.25 | Drawing a rainbow using arcs and an array of colors. (Part 1 of 2.)

```
25      // draws a rainbow using concentric arcs
26      public void paintComponent(Graphics g)
27      {
28         super.paintComponent(g);
29
30         int radius = 20; // radius of an arc
31
32         // draw the rainbow near the bottom-center
33         int centerX = getWidth() / 2;
34         int centerY = getHeight() - 10;
35
36         // draws filled arcs starting with the outermost
37         for (int counter = colors.length; counter > 0; counter--)
38         {
39            // set the color for the current arc
40            g.setColor(colors[counter - 1]);
41
42            // fill the arc from 0 to 180 degrees
43            g.fillArc(centerX - counter * radius,
44               centerY - counter * radius,
45               counter * radius * 2, counter * radius * 2, 0, 180);
46         }
47      }
48   } // end class DrawRainbow
```

Fig. 7.25 | Drawing a rainbow using arcs and an array of colors. (Part 2 of 2.)

```
1  // Fig. 7.26: DrawRainbowTest.java
2  // Test application to display a rainbow.
3  import javax.swing.JFrame;
4
5  public class DrawRainbowTest
6  {
7     public static void main(String[] args)
8     {
9        DrawRainbow panel = new DrawRainbow();
10       JFrame application = new JFrame();
11
12       application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13       application.add(panel);
14       application.setSize(400, 250);
15       application.setVisible(true);
16    }
17 } // end class DrawRainbowTest
```

Fig. 7.26 | Test application to display a rainbow. (Part 1 of 2.)