NESNEYE YÖNELİK PROGRAMLAMA
23.11.2017

Yrd. Doç.Dr. Pelin GÖRGEL

İstanbul Üniversitesi
Bilgisayar Mühendisliği Bölümü

Abstract Methods and Classes

- An *abstract class* is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.
- An *abstract method* is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

```
abstract void moveTo(double deltaX,
                     double deltaY);
```

Abstract Methods and Classes

- If a class includes abstract methods, then the class itself *must* be declared abstract, as in:

```
public abstract class GraphicObject {
   // declare non static and non final fields
   // declare nonabstract methods
   abstract void draw();
}
```

Abstract Methods and Classes

- When an abstract class is subclassed,

  the subclass usually provides implementations for all of the abstract methods in its parent class.
- However, if it does not, then the subclass must also be declared abstract.

Soyut Sınıf Örneği-1

- abstract class GraphicObject {

  int x, y; …

  void moveTo(int newX, int newY) { … }

abstract void draw();

  abstract void resize();

}

Soyut Sınıf Örneği-1

```
class Circle extends GraphicObject {
    void draw() {
      … }
    void resize() {
      … }
 }
class Rectangle extends GraphicObject {
void draw() {
      … }
void resize() {
     … }
}
```

When an Abstract Class Implements an Interface

- To define a class that does not implement all of the interface's methods, provided that the class is declared to be abstract. For example,
- abstract class X implements Y {

  // implements all but one method of Y }

  class XX extends X {

  // implements the remaining method in Y }

Kod 1:Dizi Metodları

(4.6)

```
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. (Part 1 of 4.)

Kod 1


```
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 );
```

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

Kod 1


```
45          else
46             System.out.println( "5 not found in intArray" );
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       } // end main
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       } // end method displayArray
66    } // end class ArrayManipulations
```

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