Chapter 8: Lists and Arrays

Programming with Alice and Java
First Edition

by
John Lewis
and
Peter DePasquale

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Objectives

- Learn about collections and the Java Collections API.
- Define and use of classes that handle generic types.
- Use the ArrayList class to manage data.
- Define and use Java arrays for data organization.
- Understand bounds checking with arrays.
- Explore the creation and use of multidimensional arrays.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Java Collections

- A *collection* is an object that serves as a repository for other objects.
- A collection provides services to add, remove, and manage the elements it contains.
- The underlying data structure used to implement the collection is independent of the operations the collection provides.
- Java Collections API classes define a variety of specific collections.
- ArrayList class is used to manage arrays.
- LinkedList class is used to manage a list of objects.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Generic Types

- Java enables programmers to define a class based on a *generic type* (*parameterized type*).
- Example: define the Group class to store a generic type T:

```
class Group <T>
{
        // Code that manage objects of type T
}
```

- When a Group is needed, it is instantiated with a specific class used in place of T:

```
Group<Product> group1 = new Group<Product>;
```

or `Group<Friend> group2 = new Group<Friend>;`

- Using generics provides a safer implementation than using Object as the collection type.
- A generic type cannot be instantiated.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Methods of the ArrayList Class

- The ArrayList class is part of the java.util package of the Java standard class library.
- The list dynamically grows and shrinks as needed.
- An ArrayList object stores a list of references to the Object class.
- Example:

```
ArrayList<Family> reunion =
    new ArrayList<Family>( );
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Use of ArrayList

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Use of ArrayList (continued)

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Use of ArrayList (continued)

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

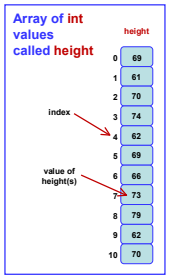
Arrays

- An array is a list of values, stored at a specific position in the array.
- This position is specified by an *index* or *subscript*.
- Array indexes begin at *zero*.
- Accessing a value in an array:

  height[8]

- What can be done within an array?

```
height[2] = 72;
height[count] = feet * 12;
average = height[0] + height[1] + height[2]) / 3;
System.out.println("Middle value: " + height[MAX/2]);
pick = height(rand.nextInt(11));
```



Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Array Declaration and Use

- To create an array, the reference to the array must be declared.
- The array can be instantiated using the new operator, which allocates memory to store values.
- int[ ] height = new int[11];
- All values stored in an array have the same type.
- An array can be set up to hold any primitive type or any object (class) type.
- Once an array object is instantiated to be a certain size, the number of values it can hold cannot be changed.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Bounds Checking

- Java performs automatic *bound checking* whenever an array element is referenced.
- It ensures that the index used to refer to an array element is in range.
- If the index is in valid range, the reference is carried out.
- If the index is not valid, an exception, ArrayIndexOutOfBoundsException, is thrown.
- Array indexes begin at zero and go up to one less than the size of the array – it is easy to create *off-by-one errors*.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Initializer Lists

- An *initializer list* can be used to instantiate an array and provide the initial values for the elements of the array.
- The new operator is not used when an initializer list is used.
- The size of the array is determined by the number of items in the initializer list.
- Examples:

```
int[ ] scores = {87, 98, 69, 87, 65, 76, 99, 83};
char[ ] vowels = {'A', 'E', 'I', 'O', 'U'};
```

- An initializer list can be used only when an array is first declared.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

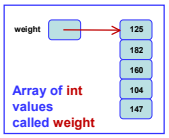
Arrays as Parameters

- An entire array can be passed as a parameter, making the formal parameter an alias of the original.
- A copy of the reference to the original array is passed.
- A method that receives an array as a parameter can permanently change an element of the array.
- The method cannot permanently change the reference to the array.
- An element of the array also can be passed to a method.
- If the element type is a primitive type, a copy of the value is passed.
- If the element is a reference to an object, a copy of the object reference is passed.

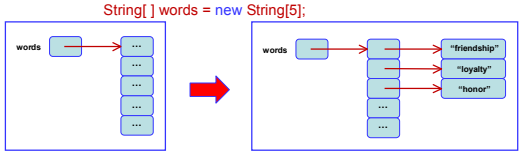
Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Arrays of Objects

- Arrays can store primitive types, but also can store elements that are references to objects.
- An array is an object itself and can be referenced by the address of the array.
- An array of objects is an array of object references.
- The new operator instantiates the array object and reserves space for five String references.



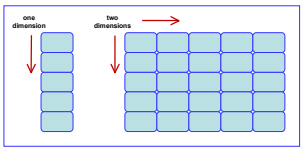
String[ ] words = new String[5];



Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Two-dimensional Arrays

- A *two-dimensional array* has values in two dimensions thought of as the rows and columns of a table.
- To refer to a value in a two-dimensional array two indexes are used, specifying row and column.
- Java represents a two-dimensional array as an array of arrays.
- Nested loops are helpful when processing a two-dimensional array.



Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Multidimensional Arrays

- Any array with more than one dimension is called a *multidimensional array*.
- A three-dimensional array can be visualized as a cube.
- It is difficult to visualize a multidimensional array.
- Each subsequent dimension is a subdivision of the previous one.
- Java does not directly support multidimensional arrays, but they are represented as an array of references to array objects.

Visualization of four-dimensional array

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Summary

- A list is just one of several types of collections available in the Java API.
- Generic classes ensure type compatibility among the objects stored by the collection.
- The capacity of an ArrayList object changes dynamically as needed.
- An array of size N is indexed from 0 to N-1.
- In Java, an array is an object and thus must be instantiated.
- Bounds checking ensures that an index used to refer to an array object is in range.
- An initializer list can be used to instantiate an array object instead of using the new operator.
- An entire array can be passed as a parameter, making the formal parameter an alias of the original.
- Instantiating an array of objects reserves room to store references only. The objects stored in each element must be instantiated separately.
- Using an array with more than two dimensions is rare in an object-oriented system.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley