7

ArrayList

Objectives

- midterm results
- review midterm solutions
- dynamically resizing arrays: ArrayList class

Midterm results

- great performance
- mean = 80 (min = 17.5, max = 98)
- test solutions on the website

Outline

7.1 Introduction
7.2 Arrays
7.3 Declaring and Creating Arrays
7.4 Examples Using Arrays
7.5 Case Study: Card Shuffling and Dealing Simulation
7.6 Enhanced `for` Statement
7.7 Passing Arrays to Methods
7.8 Case Study: Class `GradeBook` Using an Array to Store Grades
7.9 Multidimensional Arrays
7.10 Case Study: Class `GradeBook` Using a Two-Dimensional Array
7.11 Variable-Length Argument Lists
7.12 Using Command-Line Arguments
7.13 (Optional) GUI and Graphics Case Study: Drawing Arcs
7.14 (Optional) Software Engineering Case Study: Collaboration Among Objects
7.15 Wrap-Up

Variable-Length Argument Lists

- Variable-length argument lists
  - Unspecified number of arguments
  - Use ellipsis (...) in method's parameter list
    - Can occur only once in parameter list
    - Must be placed at the end of parameter list
  - Array whose elements are all of the same type

Outline

```
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; // initialize total
10
11       // calculate total using the enha
12       for ( double d : numbers )
13          total += d;
14
15       return total / numbe
16    } // end method average
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
```

Method `average` receives a variable length sequence of `double`s

Calculate the total of the `double`s in the array

Access `numbers.length` to obtain the size of the `numbers` array

`VarargsTest`

`.java`

(1 of 2)

Line 7

Lines 12-13

Line 15

Outline

```
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 d
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     } // end main
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
```

Invoke method average with two arguments

Invoke method average with three arguments

Invoke method average with four arguments

VarargsTest .java

2)

Line 29

Line 31

Line 33

Program output

Common Programming Error 7.6

Placing an ellipsis in the middle of a method parameter list is a syntax error. An ellipsis may be placed only at the end of the parameter list.

Collections Introduction

- Java collections framework
  - Contain prepackaged data structures, interfaces, algorithms
  - Use generics
  - Use existing data structures
    - Example of code reuse
  - Provides reusable componentry

Collections Overview

- **Collection**
  - **Data structure (object) that can hold references to other objects**
- **Collections framework**
  - **Interfaces declare operations for various collection types**
  - **Provide high-performance, high-quality implementations of common data structures**
  - **Enable software reuse**
  - **Enhanced with generics capabilities in J2SE 5.0**
    - **Compile-time type checking**

Lists

- List
  - Ordered `Collection` that can contain duplicate elements
  - Sometimes called a *sequence*
  - Implemented via interface `List`
    - `ArrayList`
    - `LinkedList`
    - `Vector`

Class `ArrayList`

- ArrayList is a dynamically sized array
  - Generic array of elements
  - Specify item type using `<...>` , for example:

    `ArrayList<String>, ArrayList<Object>`

- Instance methods
  - `get`
  - `size`
  - `add`
  - `contains`
  - and others ...