3b

Arrays

OBJECTIVES

In this lecture you will learn:

- What arrays are.
- To use arrays to store data in and retrieve data from lists and tables of values.
- To declare an array, initialize an array and refer to individual elements of an array.
- To use the enhanced `for` statement to iterate through arrays.
- To pass arrays to methods.
- To declare and manipulate multidimensional arrays.
- To write methods that use variable-length argument lists.
- To read command-line arguments into a program.

Outline

3b.1 Introduction

Arrays

- Data structures
- Related data items of same type
- Remain same size once created
  - Fixed-length entries

3b.2 Arrays

Array

- **Group of variables**
  - **Have same type**
- **Reference type**

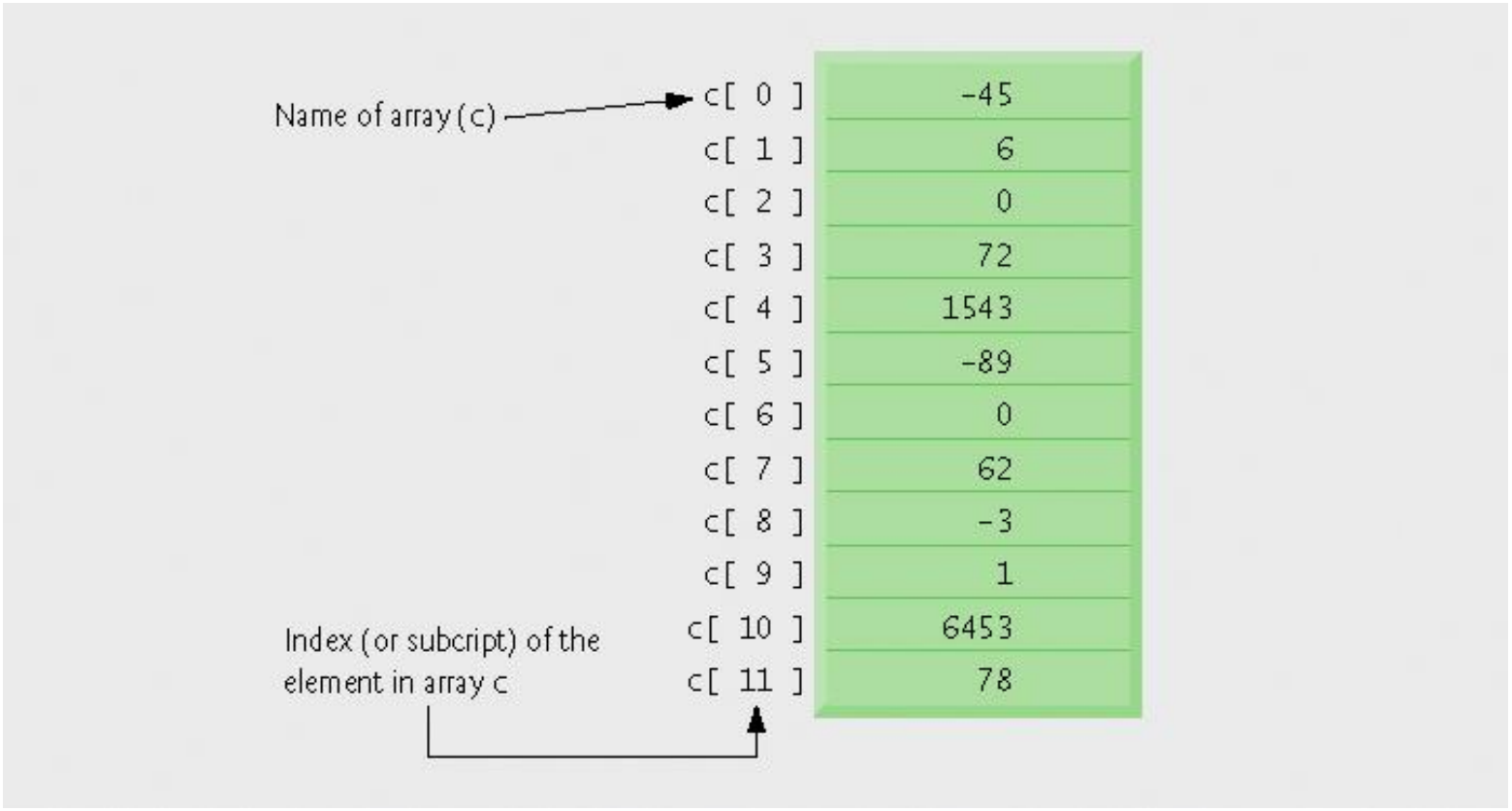

Fig. 3b.1 | A 12-element array.

3b.2 Arrays (Cont.)

Index

- Also called subscript
- Position number in square brackets
- Must be positive integer or integer expression
- First element has index zero

```
a = 5;
b = 6;
c[ a + b ] += 2;
```

- Adds 2 to c[11]

Common Programming Error 3b.1

Using a value of type `long` as an array index results in a compilation error. An index must be an `int` value or a value of a type that can be promoted to int—namely, `byte`, `short` or `char`, but not `long`.

3b.2 Arrays (Cont.)

Examine array `c`

- **`c` is the array *name***
- **`c.length` accesses array `c`'s *length***
- **`c` has 12 *elements* (`c[0]`, `c[1]`, … `c[11]`)**
  - **The *value* of `c[0]` is `-45`**

3b.3 Declaring and Creating Arrays

Declaring and Creating arrays

- Arrays are objects that occupy memory
- Created dynamically with keyword new

```
int c[] = new int[ 12 ];
```

  - Equivalent to

```
int c[];  // declare array variable
c = new int[ 12 ]; // create array
```

- We can create arrays of objects too

```
String b[] = new String[ 100 ];
```

Common Programming Error 3b.2

In an array declaration, specifying the number of elements in the square brackets of the declaration (`e.g., int c[ 12 ];`) is a syntax error.

Good Programming Practice 3b.1

For readability, declare only one variable per declaration. Keep each declaration on a separate line, and include a comment describing the variable being declared.

Common Programming Error 3b.3

Declaring multiple array variables in a single declaration can lead to subtle errors. Consider the declaration `int[] a, b, c;`. If `a`, `b` and `c` should be declared as array variables, then this declaration is correct—placing square brackets directly following the type indicates that all the identifiers in the declaration are array variables. However, if only `a` is intended to be an array variable, and `b` and `c` are intended to be individual int variables, then this declaration is incorrect—the declaration `int a[], b, c;` would achieve the desired result.

3b.4 Examples Using Arrays

Declaring arrays

Creating arrays

Initializing arrays

Manipulating array elements

3b.4 Examples Using Arrays

Creating and initializing an array

- **Declare array**
- **Create array**
- **Initialize array elements**

Outline

```
1  // Fig. 7.2: InitArray.java
2  // Creating an array.
3
4  public class InitArray
5  {
6     public static void main( Stri
7     {
8        int array[]; // declare array named array
9
10       array = new int[ 10 ]; // create the sp
11
12       System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings
13
14       // output each array element's value
15       for ( int counter = 0; counter < array.length; counter++ )
16          System.out.printf( "%5d%8d\n", counter, array[ counter ] );
17    } // end main
18 } // end class InitArray
```

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

Declare `array` as an array of `int`s

Create 10 `int`s for `array`; each `int` is initialized to 0 by default

`array.length` returns length of `array`

Each `int` is initialized to 0 by default

`array[counter]` returns `int` associated with index in `array`

InitArray.java

Line 8
Declare array as an array of **int**s

Line 10
Create 10 **int**s for array; each **int** is initialized to 0 by default

Line 15
array.length returns length of array

Line 16
array[counter] returns **int** associated with index in array

Program output

3b.4 Examples Using Arrays (Cont.)

Using an array initializer

- Use *initializer list*
  - Items enclosed in braces ({})
  - Items in list separated by commas

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

    - Creates a five-element array
    - Index values of 0, 1, 2, 3, 4
- Do not need keyword new

```java
// Fig. 7.3: InitArray.java
// Initializing the elements of an array with an array initializer.

public class InitArray
{
   public static void main( String args[] )
   {
      // initializer list specifies the value for each element
      int array[] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };

      System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings

      // output each array element's value
      for ( int counter = 0; counter < array.length; counter++ )
         System.out.printf( "%5d%8d\n", counter, array[ counter ] );
   } // end main
} // end class InitArray
```

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

Declare `array` as an array of `int`s

Compiler uses initializer list to allocate array

Outline

itArray.java

Line 9
Declare array as an array of ints

Line 9
Compiler uses initializer list to allocate array

Program output

3b.4 Examples Using Arrays (Cont.)

Calculating a value to store in each array element

- **Initialize elements of 10-element array to even integers**

Outline

```
1  // Fig. 7.4: InitArray.java
2  // Calculating values to be placed into 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 ar
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
20    } // end main
21 } // end class InitArray
```

Declare constant variable ARRAY_LENGTH using the `final` modifier

Declare and create `array` that contains 10 `int`s

Use `array` index to assign array value

InitArray.java

8
are constant
variable

Line 9
Declare and
create array that
contains 10 `int`s

Line 13
Use array index
to assign array

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

Program output

Good Programming Practice 3b.2

Constant variables also are called named constants or read-only variables. Such variables often make programs more readable than programs that use literal values (e.g., 10)—a named constant such as ARRAY_LENGTH clearly indicates its purpose, whereas a literal value could have different meanings based on the context in which it is used.

Common Programming Error 3b.4

Assigning a value to a constant after the variable has been initialized is a compilation error.

Common Programming Error 3b.5

Attempting to use a constant before it is initialized is a compilation error.

3b.4 Examples Using Arrays (Cont.)

Summing the elements of an array

- **Array elements can represent a series of values**
  - **We can sum these values**

```
1  // Fig. 7.5: SumArray.java
2  // Computing the sum of the elements of
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    } // end main
17 } // end class SumArray
```

Declare `array` with initializer list

Sum all `array` values

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

Outline

SumArray.java

Line 8
Declare array with initializer list

Lines 12-13
Sum all array values

Program output

3b.4 Examples Using Arrays (Cont.)

Using bar charts to display array data graphically

- **Present data in graphical manner**
  - **E.g., bar chart**
- **Examine the distribution of grades**

```
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( "*" );
25
26          System.out.println(); // start a new line of output
27       } // end outer for
28    } // end main
29 } // end class BarChart
```

Declare `array` with initializer list

Use the `0` flag to display one-digit grade with a leading 0

For each `array` element, print associated number of asterisks

Outline

BarChart.java

(1 of 2)

Line 8
Declare array with initializer list

Line 19
Use the `0` flag to display one-digit grade with a leading 0

associated
number of
asterisks

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

Outline

BarChart.java

(2 of 2)

Program output

3b.4 Examples Using Arrays (Cont.)

Using the elements of an array as counters

- **Use a series of counter variables to summarize data**

Outline

```
1  // Fig. 7.7: RollDie.java
2  // Roll a six-sided die 6000 times.
3  import java.util.Random;
4
5  public class RollDie
6  {
7     public static void main( String args[] )
8     {
9        Random randomNumbers = new Random(); // random number generator
10       int frequency[] = new int[ 7 ]; // array of frequency counters
11
12       // roll die 6000 times; use die value as frequency ind
13       for ( int roll = 1; roll <= 6000; roll++ )
14          ++frequency[ 1 + randomNumbers.nextInt( 6 ) ];
15
16       System.out.printf( "%s%10s\n"
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    } // end main
22 } // end class RollDie
```

```
Face Frequency
   1       988
   2       963
   3      1018
   4      1041
   5       978
   6      1012
```

Declare `frequency` as array of 7 `int`s

Generate 6000 random integers in range 1-6

Increment `frequency` values at index associated with random number

Die.java

Line 10
Declare frequency as of 7 ints

13-14
Generate 6000 random integers in range 1-6

Line 14
Increment frequency values at index associated with random number

Program output

3b.4 Examples Using Arrays (Cont.)

Using arrays to analyze survey results

- **40 students rate the quality of food**
  - **1–10 Rating scale: 1 means awful, 10 means excellent**
- **Place 40 responses in array of integers**
- **Summarize results**

```
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        // array of survey responses
9        int responses[] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
10          10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5
11          4, 8, 6, 8, 10 };
12       int frequency[] = new int[ 11 ]; // array of frequency counters
13
14       // for each answer, select responses element and use that value
15       // as frequency index to determine element to increment
16       for ( int answer = 0; answer < responses.length; answer++ )
17          ++frequency[ responses[ answer ] ];
18
19       System.out.printf( "%s%10s", "Rating", "Frequency" );
20
21       // output each array element's value
22       for ( int rating = 1; rating < frequency.length; rating++ )
23          System.out.printf( "%d%10d", rating, frequency[ rating ] );
24    } // end main
25 } // end class StudentPoll
```

Declare `responses` as array to store 40 responses

Declare `frequency` as array of 11 `int` and ignore the first element

For each response, increment `frequency` values at index associated with that response

Outline

StudentPoll.java

(1 of 2)

onses as array to store 40 responses

Line 12 Declare frequency as array of 11 int e the ement

-17 response, increment frequency values at index associated with that response

```
Rating Frequency
     1         2
     2         2
     3         2
     4         2
     5         5
     6        11
     7         5
     8         7
     9         1
    10         3
```

Outline

StudentPoll.java

(2 of 2)

Program output

Error-Prevention Tip 3b.1

An exception indicates that an error has occurred in a program. A programmer often can write code to recover from an exception and continue program execution, rather than abnormally terminating the program. When a program attempts to access an element outside the array bounds, an `ArrayIndexOutOfBoundsException` occurs. Exception handling is discussed in Chapter 13.

Error-Prevention Tip 3b.2

When writing code to loop through an array, ensure that the array index is always greater than or equal to 0 and less than the length of the array. The loop-continuation condition should prevent the accessing of elements outside this range.

3b.5 Case Study: Card Shuffling and Dealing Simulation

Program simulates card shuffling and dealing

- **Use random number generation**
- **Use an array of reference type elements to represent cards**
- **Three classes**
  - **Card**
    - **Represents a playing card**
  - **DeckOfCards**
    - **Represents a deck of 52 playing cards**
  - **DeckOfCardsTest**
    - **Demonstrates card shuffling and dealing**

Outline

Card.java

Lines 17-20

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

Return the string representation of a card

```
1  // Fig. 7.10: DeckOfCards.java
2  // DeckOfCards class represents a deck of playing cards.
3  import java.util.Random;
4
5  public class DeckOfCards
6  {
7     private Card deck[]; // array of Card objects
8     private int currentCard; // index of next Card to be dea
9     private final int NUMBER_OF_CARDS = 52; // constant number of Cards
10    private Random randomNumbers; // random number generator
11
12    // constructor fills deck of Cards
13    public DeckOfCards()
14    {
15       String faces[] = { "Ace", "Deuce
16          "Seven", "Eight", "Nine", "Te
17       String suits[] = { "Hearts", "Di
18
19       deck = new Card[ NUMBER_OF_CARDS ]; // create array of Card objects
20       currentCard = 0; // set currentCard so first Card dealt is deck[ 0 ]
21       randomNumbers = new Random(); // create random nu
22
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    } // end DeckOfCards constructor
```

Declare `deck` as array to store `Card` objects

Constant `NUMBER_OF_CARDS` indicates the number of Cards in the deck

Declare and initialize `faces` with `String`s that represent the face of card

Declare and initialize `suits` with `String`s that represent the suit of card

Fill the `deck` array with `Card`s

Outline

(1 of 2)

Line 7

Line 9

Lines 15-16

Line 17

Lines 24-26

Outline

DeckOfCards.java

(2 of 2)

```
28
29    // shuffle deck of Cards with one-pass algorithm
30    public void shuffle()
31    {
32       // after shuffling, dealing should start at deck[ 0 ] again
33       currentCard = 0; // reinitialize currentCard
34
35       // for each Card, pick another random Card 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       } // end for
46    } // end method shuffle
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    } // end method dealCard
57 } // end class DeckOfCards
```

Swap current `Card` with randomly selected `Card`

-44

Line 52

Determine whether `deck` is empty

Outline

DeckOfCardsTest.java

(1 of 2)

```
1  // Fig. 7.11: DeckOfCardsTest.java
2  // Card shuffling and dealing application.
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 = 0; i < 13; i++ )
14       {
15          // deal and print 4 Cards
16          System.out.printf( "%-20s%-20s%-20s%-20s\n",
17             myDeckOfCards.dealCard(), myDeckOfCards.dealCard(),
18             myDeckOfCards.dealCard(), myDeckOfCards.dealCard() );
19       } // end for
20    } // end main
21 } // end class DeckOfCardsTest
```

```
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
```

Outline

`DeckOfCardsTest.java`

(2 of 2)