Chapter 7, 8 Arrays

Uploaded By: Jibreel Bornat

1

Opening Problem

Read one hundred numbers, compute their average, and find out how many numbers are above the average.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

2

Introducing Arrays

Array is a data structure that represents a collection of the same types of data.

```
double[] myList = new double[10];
```

myList | reference | → myList[0] | 5.6

Array reference variable

myList[1]	4.5
myList[2]	3.3
myList[3]	13.2
myList[4]	4.0

Array element at index 5 → myList[5] | 34.33 ← Element value

myList[6]	34.0
myList[7]	45.45
myList[8]	99.993
myList[9]	11123

Uploaded By: Jibreel Bornat
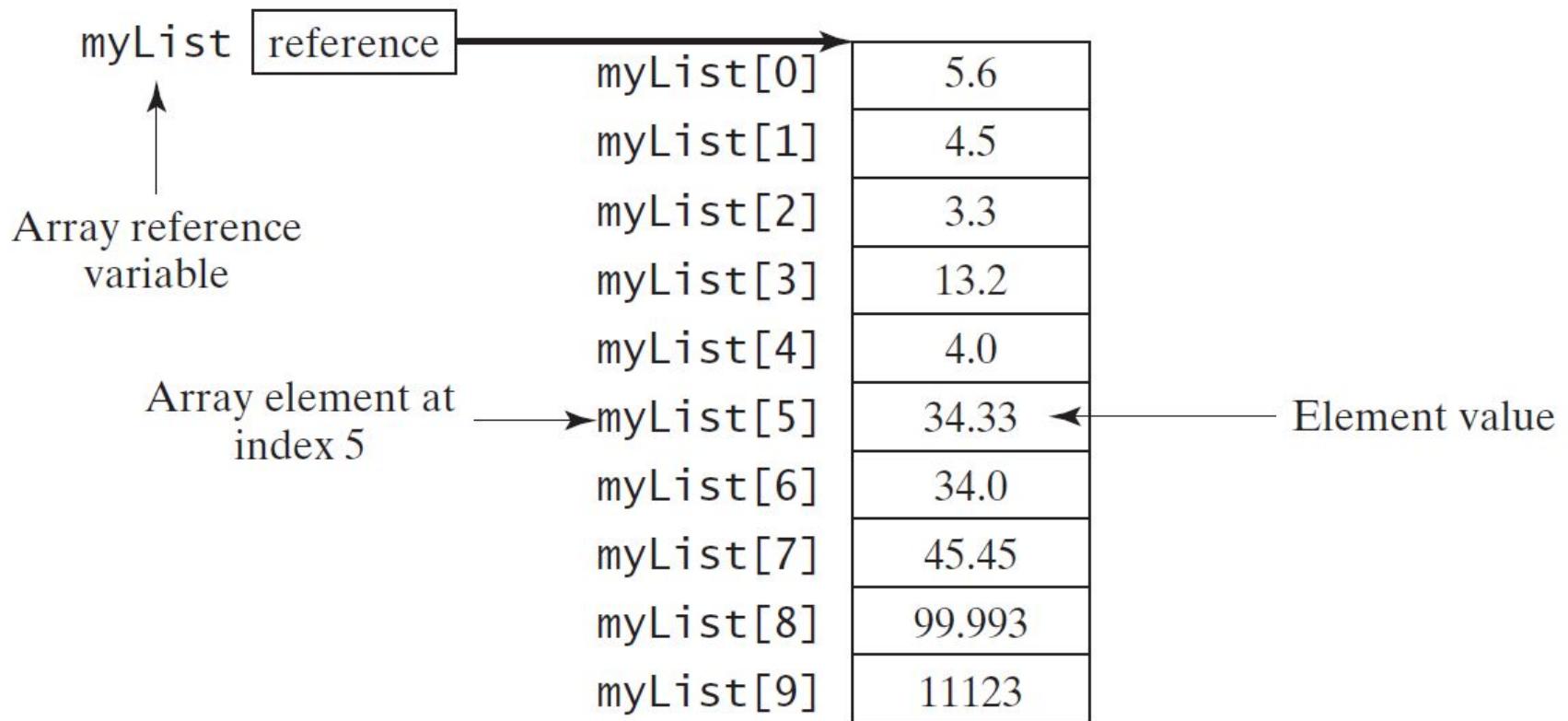
Declaring Array Variables

- `datatype[] arrayRefVar;`

`double[] myList;`
  - `datatype arrayRefVar[];` <u>// This style is allowed, but not preferred</u>

  `double myList[];`

- `arrayRefVar = new datatype[arraySize];`

  `myList = new double[10];`

   `myList[0]` references the first element in the array.

- `datatype[] arrayRefVar = new`
     `datatype[arraySize];`

  `double[] myList = new double[10];`

- `myList.length returns 10`

Uploaded By: Jibreel Bornat

4

More on Arrays

When an array is created, its elements are assigned the default value of

<u>0</u> for the numeric primitive data types,

<u>'\u0000'</u> for <u>char</u> types, and

<u>false</u> for <u>boolean</u> types.

The array elements are accessed through the index. The array indices are *0-based*, i.e., it starts from 0 to arrayRefVar.length-1.

```
myList[2] = myList[0] + myList[1];
```

Uploaded By: Jibreel Bornat

Declaring, creating, initializing Using the Shorthand Notation

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

This shorthand notation is equivalent to the following statements:

```
double[] myList = new double[4];

myList[0] = 1.9;

myList[1] = 2.9;

myList[2] = 3.4;

myList[3] = 3.5;
```

Uploaded By: Jibreel Bornat

CAUTION

Using the shorthand notation, you have to declare, create, and initialize the array all in one statement. Splitting it would cause a syntax error. For example, the following is wrong:

double[] myList;

myList = {1.9, 2.9, 3.4, 3.5};

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Trace Program with Arrays

Declare array variable values, create an array, and assign its reference to values

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0

Uploaded By: Jibreel Bornat

8

Trace Program with Arrays

BIRZEIT UNIVERSITY

> After this line, values[0] is 11 (1 + 10)

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5, i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

0	11
1	1
2	3
3	6
4	10

Initializing arrays

- With input values

```
java.util.Scanner input = new java.util.Scanner(System.in);
System.out.print("Enter " + myList.length + " values: ");
for (int i = 0; i < myList.length; i++)
  myList[i] = input.nextDouble();
```

- With Random numbers

```
for (int i = 0; i < myList.length; i++) {
  myList[i] = Math.random() * 100;
}
```

- Printing:

```
for (int i = 0; i < myList.length; i++) {
  System.out.print(myList[i] + " ");
}
```

Uploaded By: Jibreel Bornat 10

Array Processing

- ## Summing

```
for (int i = 0; i < myList.length; i++) {
  total += myList[i];
}
```

- ## Largest Element

```
double max = myList[0];
for (int i = 1; i < myList.length; i++) {
  if (myList[i] > max) max = myList[i];
}
```

Uploaded By: Jibreel Bornat 11

Shuffling & Shifting

```java
for (int i = 0; i < myList.length - 1; i++) {
  // Generate an index j randomly
  int j = (int)(Math.random()
    * myList.length);

  // Swap myList[i] with myList[j]
  double temp = myList[i];
  myList[i] = myList[j];
  myList[j] = temp;
}
```

myList

i → [0] .
[1]
[i] .

A random index [j]

swap
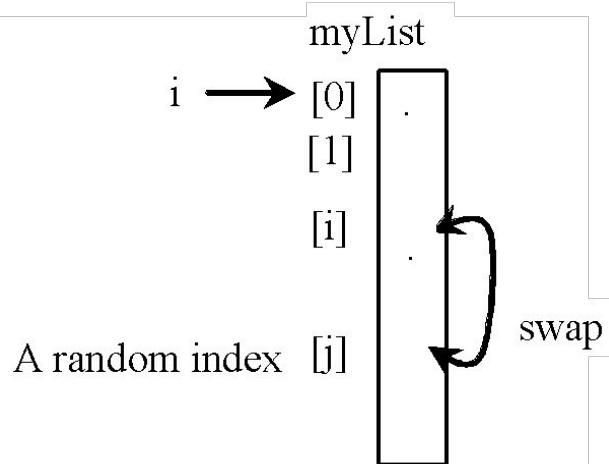
```java
double temp = myList[0]; // Retain the first element

// Shift elements left
for (int i = 1; i < myList.length; i++) {
  myList[i - 1] = myList[i];
}

// Move the first element to fill in the last position
myList[myList.length - 1] = temp;
```

myList

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

12
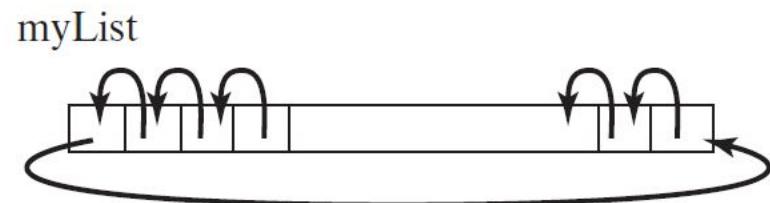
Enhanced <u>for</u> Loop (for-each loop)

JDK 1.5 introduced a new for loop that enables you to traverse the complete array sequentially without using an index variable. For example, the following code displays all elements in the array myList:

```
for (double value: myList)
   System.out.println(value);
```

In general, the syntax is

```
for (elementType value: arrayRefVar) {
   // Process the value
}
```

You still have to use an index variable if you wish to traverse the array in a different order or change the elements in the array.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Analyze Numbers

Read one hundred numbers, compute their average, and find out how many numbers are above the average.

AnalyzeNumbers   Run

Uploaded By: Jibreel Bornat

Copying Arrays

Often, in a program, you need to duplicate an array or a part of an array. In such cases you could attempt to use the assignment statement (=), as follows:

list2 = list1;

Before the assignment
`list2 = list1;`

list1 ⟶ Contents of list1

list2 ⟶ Contents of list2

After the assignment
`list2 = list1;`

list1 ⟶ Contents of list1

list2 ⟶ Contents of list1

Contents of list2
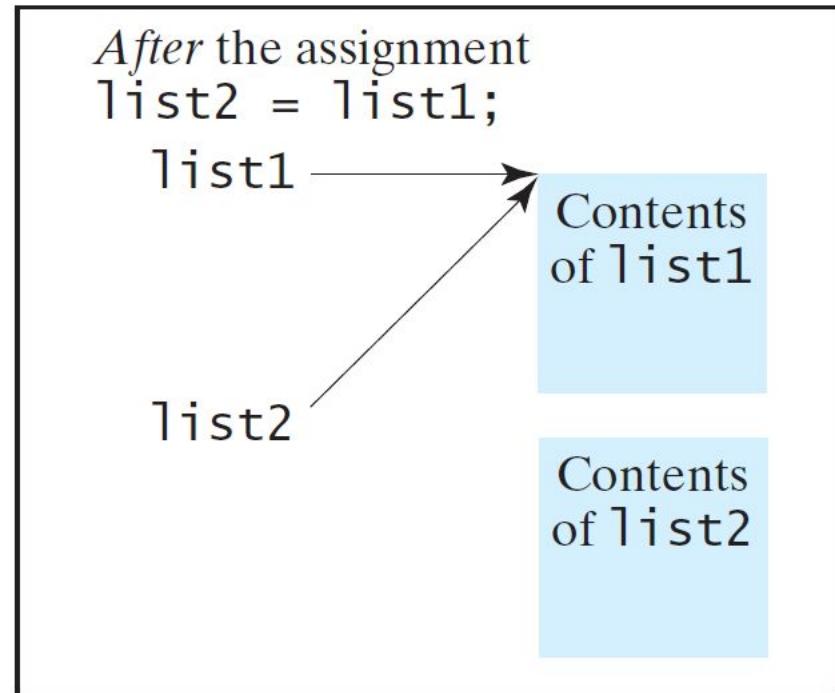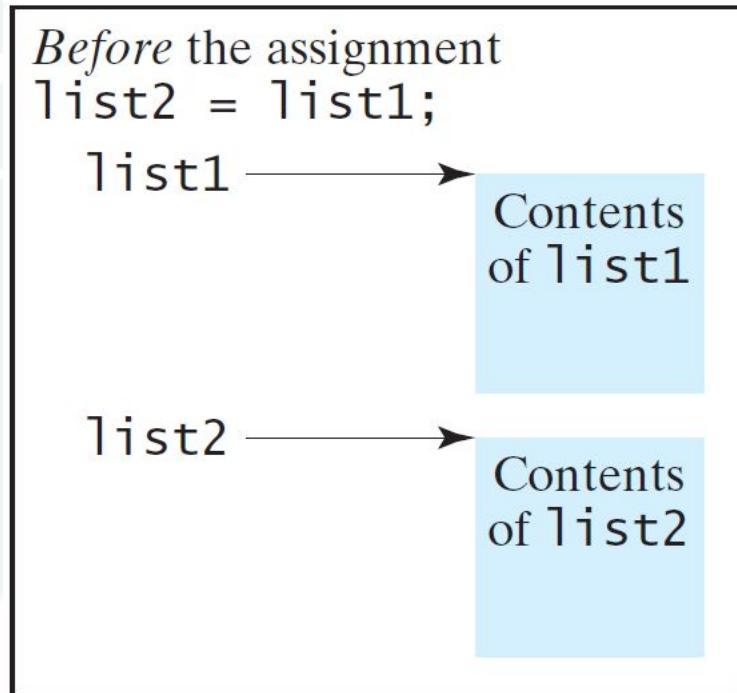
Copying Arrays

Using a loop:

```
int[] sourceArray = {2, 3, 1, 5, 10};
int[] targetArray = new
  int[sourceArray.length];


for (int i = 0; i < sourceArrays.length; i++)
    targetArray[i] = sourceArray[i];
```

Uploaded By: Jibreel Bornat

The `arraycopy` Utility

arraycopy(sourceArray, src_pos, targetArray, tar_pos, length);

Example:

System.arraycopy(sourceArray, 0, targetArray, 0, sourceArray.length);

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Passing Arrays to Methods

```java
public static void printArray(int[] array) {
  for (int i = 0; i < array.length; i++) {
    System.out.print(array[i] + " ");
  }
}
```

Invoke the method

```java
int[] list = {3, 1, 2, 6, 4, 2};
printArray(list);
```

Invoke the method

```java
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

Anonymous array

Uploaded By: Jibreel Bornat

BIRZEIT UNIVERSITY

Anonymous Array

The statement

     printArray(new int[]{3, 1, 2, 6, 4, 2});

creates an array using the following syntax:

     new dataType[]{literal0, literal1, ..., literalk};

There is no explicit reference variable for the array. Such array is called an *anonymous array*.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Pass by Reference vs. By Value

Java uses *pass by value* to pass arguments to a method. There are important differences between passing a value of variables of primitive data types and passing arrays.

● For a parameter of a primitive type value, the actual value is passed. Changing the value of the local parameter inside the method does not affect the value of the variable outside the method.

● For a parameter of an array type, the value of the parameter contains a reference to an array; this reference is passed to the method. Any changes to the array that occur inside the method body will affect the original array that was passed as the argument.

STUDENTS-HUB.com   Uploaded By: Jibreel Bornat

Simple Example

```java
public class Test {
  public static void main(String[] args) {
    int x = 1; // x represents an int value
    int[] y = new int[10]; // y represents an array of int values

    m(x, y); // Invoke m with arguments x and y

    System.out.println("x is " + x);
    System.out.println("y[0] is " + y[0]);
  }

  public static void m(int number, int[] numbers) {
    number = 1001; // Assign a new value to number
    numbers[0] = 5555; // Assign a new value to numbers[0]
  }
}
```

Call Stack
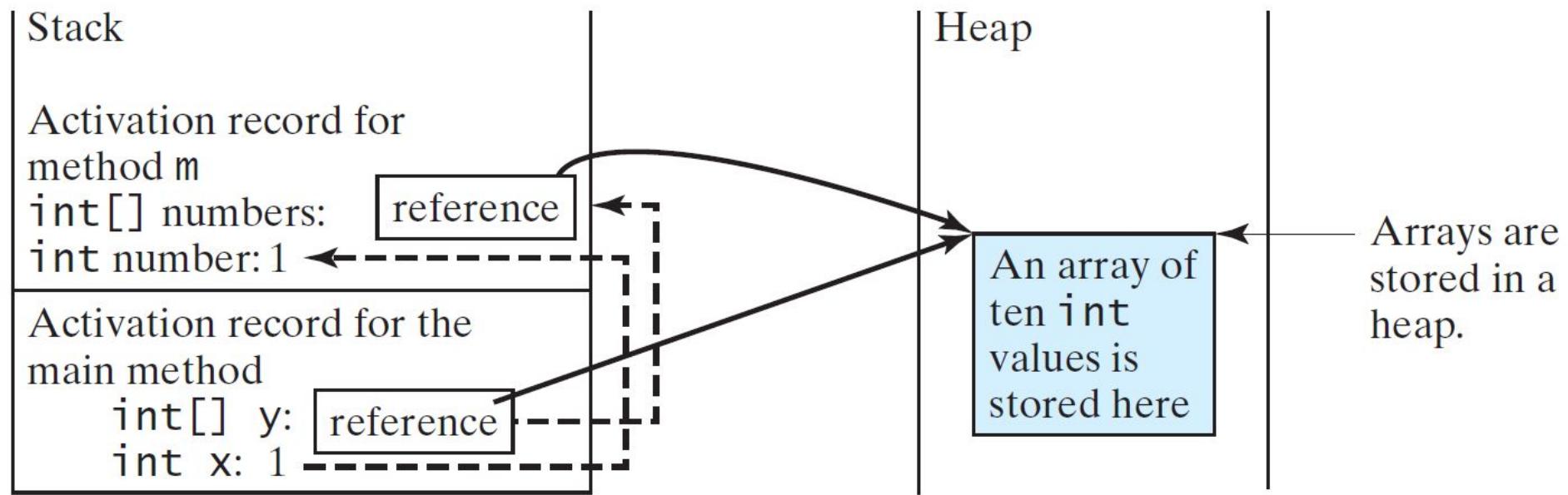
When invoking m(x, y), the values of x and y are passed to number and numbers. Since y contains the reference value to the array, numbers now contains the same reference value to the same array.

Uploaded By: Jibreel Bornat

22

Call Stack

When invoking m(x, y), the values of x and y are passed to number and numbers. Since y contains the reference value to the array, numbers now contains the same reference value to the same array.

Uploaded By: Jibreel Bornat

23
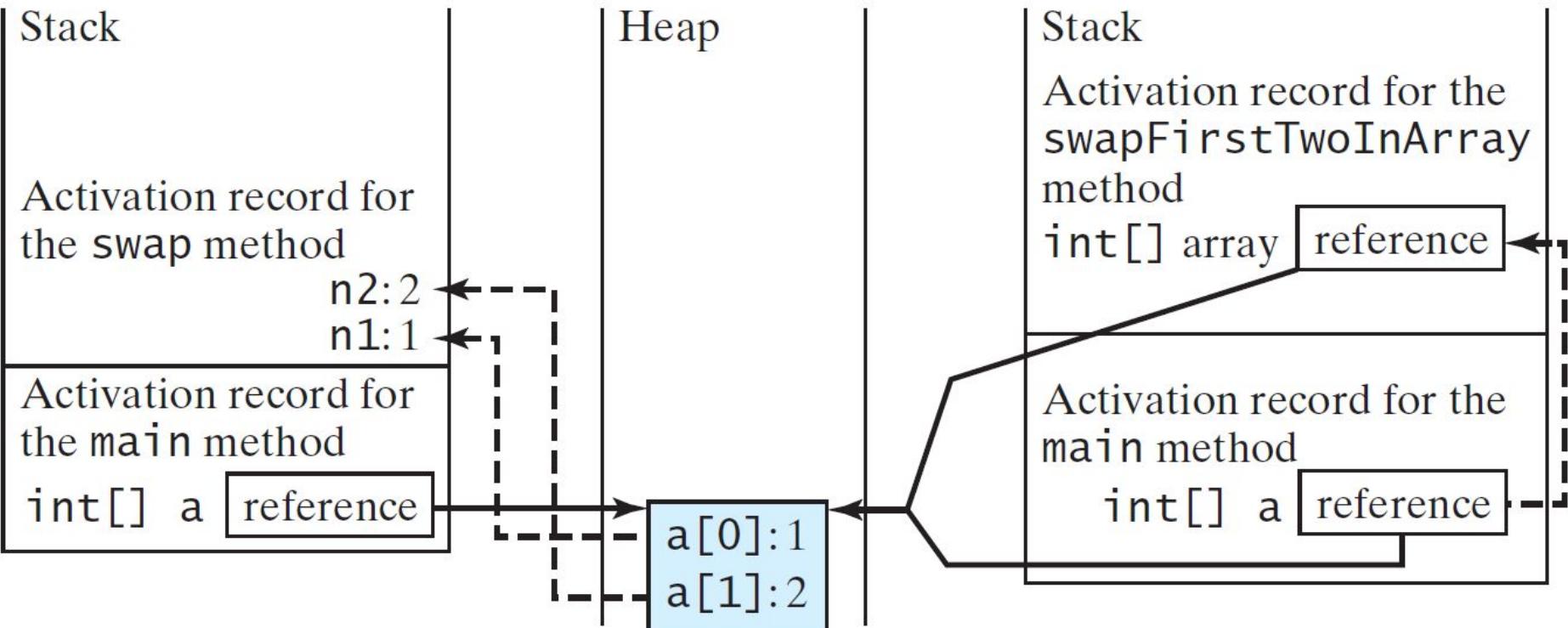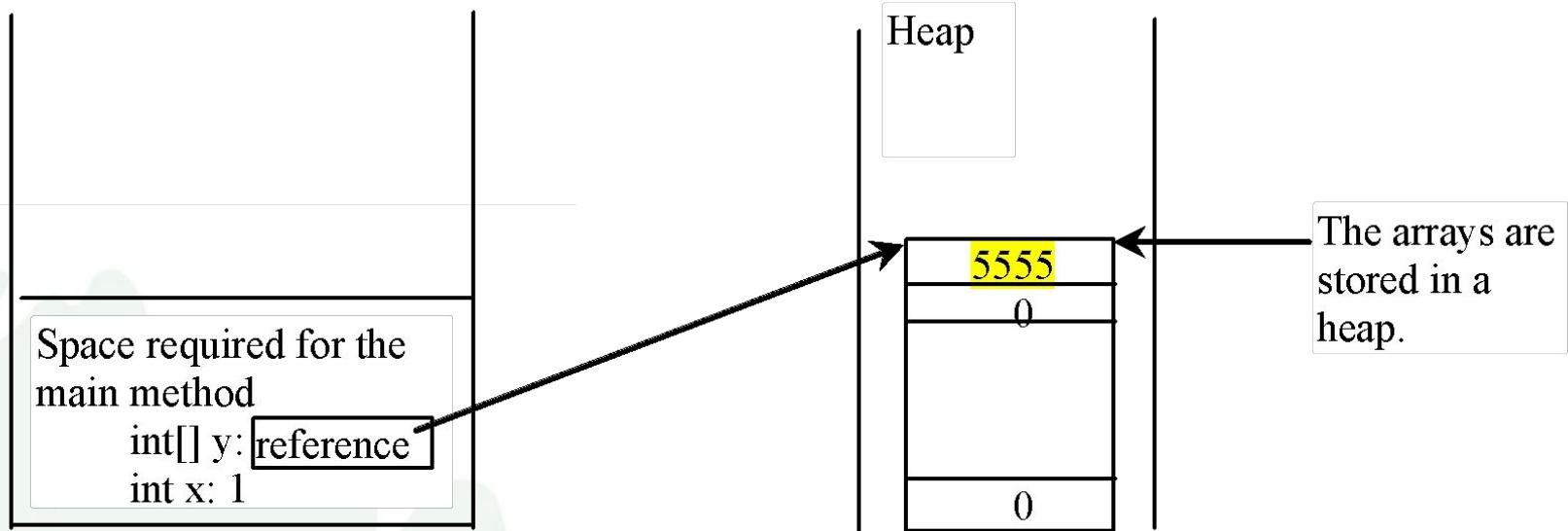
Heap

The JVM stores the array in an area of memory, called *heap*, which is used for dynamic memory allocation where blocks of memory are allocated and freed in an arbitrary order.

 Uploaded By: Jibreel Bornat

Passing Arrays as Arguments

- Objective: Demonstrate differences of passing primitive data type variables and array variables.

TestPassArray   Run

Example, cont.

Stack

Space required for the swap method

    n2: 2
    n1: 1

Space required for the main method
    int[] a [reference]

Heap

a[1]: 2
a[0]: 1

Stack
Space required for the swapFirstTwoInArray method
 int[] array [reference]

Space required for the main method
    int[] a [reference]
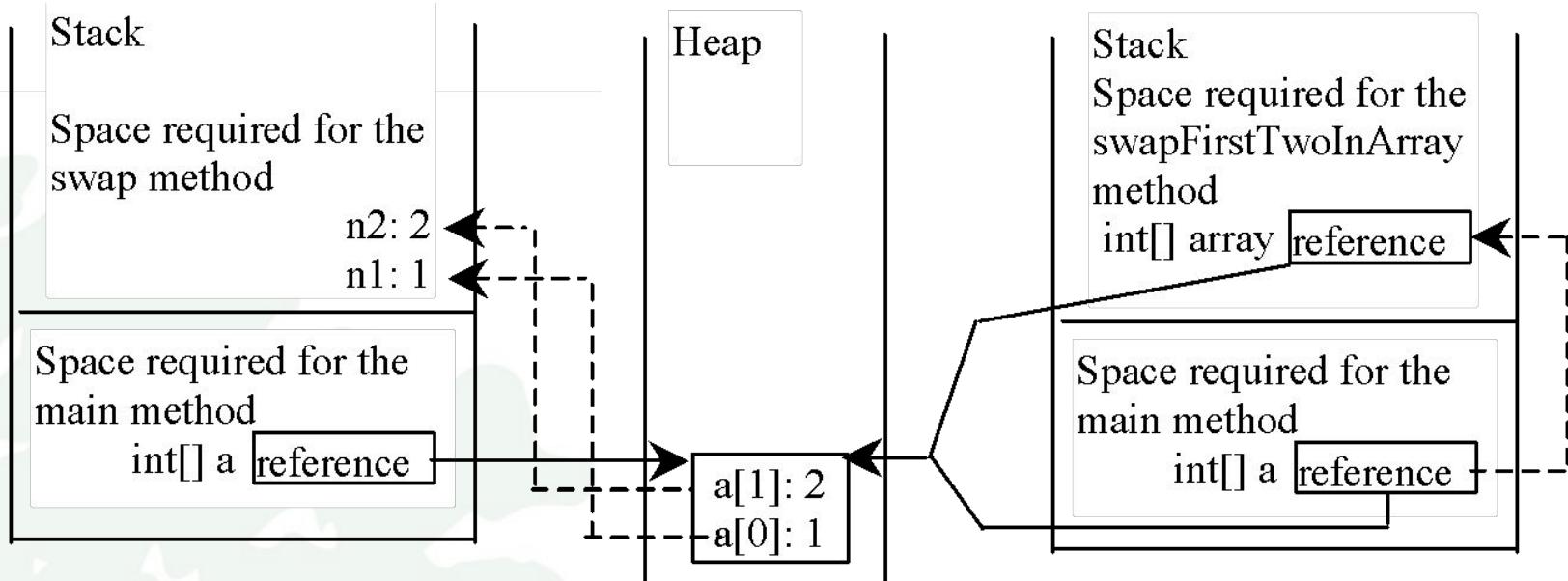
Invoke swap(int n1, int n2). The primitive type values in a[0] and a[1] are passed to the swap method.

The arrays are stored in a heap.

Invoke swapFirstTwoInArray(int[] array). The reference value in a is passed to the swapFirstTwoInArray method.

STUDENTS-HUB.com   Uploaded By: Jibreel Bornat

Returning an Array from a Method

```java
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list

result

```java
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```
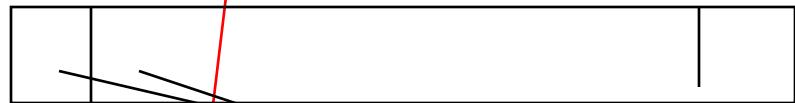
Uploaded By: Jibreel Bornat
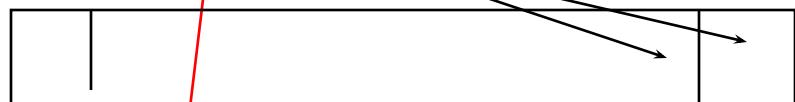
Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

Return result

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list | 1 | 2 | 3 | 4 | 5 | 6 |

list2

result | 6 | 5 | 4 | 3 | 2 | 1 |

Uploaded By: Jibreel Bornat   28
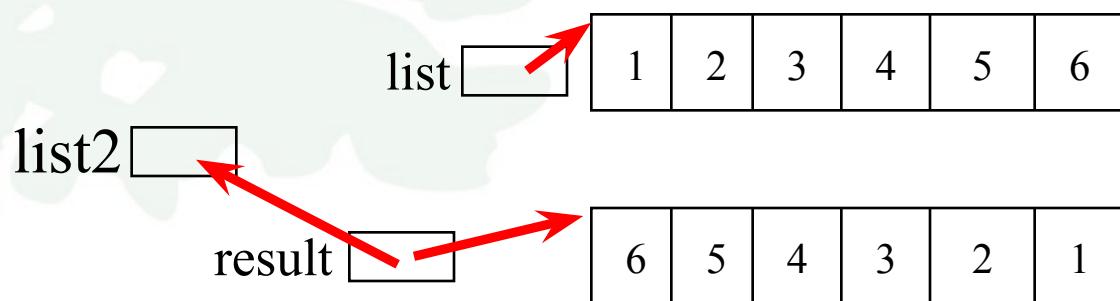
Searching Arrays

Searching is the process of looking for a specific element in an array; for example, discovering whether a certain score is included in a list of scores. Searching is a common task in computer programming. There are many algorithms and data structures devoted to searching. In this section, two commonly used approaches are discussed, *linear search* and *binary search*.

```java
public class LinearSearch {
  /** The method for finding a key in the list */
  public static int linearSearch(int[] list, int key) {
    for (int i = 0; i < list.length; i++)
      if (key == list[i])
        return i;
    return -1;
  }
}
```

```
                          [0] [1] [2] ...
list  [ | | |          |   |   | ]

key    Compare key with list[i] for i = 0, 1, ...
```

STUDENTS-HUB.com   Uploaded By: Jibreel Bornat   29
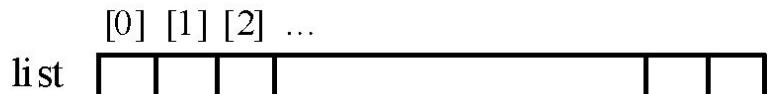
Linear Search

The linear search approach compares the key element, <u>key</u>, *sequentially* with each element in the array <u>list</u>. The method continues to do so until the key matches an element in the list or the list is exhausted without a match being found. If a match is made, the linear search returns the index of the element in the array that matches the key. If no match is found, the search returns <u>-1</u>.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

From Idea to Solution

```
/** The method for finding a key in the list */
public static int linearSearch(int[] list, int key) {
    for (int i = 0; i < list.length; i++)
        if (key == list[i])
            return i;
    return -1;
}
```

Trace the method

```
int[] list = {1, 4, 4, 2, 5, -3, 6, 2};
int i = linearSearch(list, 4);  // returns 1
int j = linearSearch(list, -4); // returns -1
int k = linearSearch(list, -3); // returns 5
```

Uploaded By: Jibreel Bornat 31

Binary Search

For binary search to work, the elements in the array must already be ordered. Without loss of generality, assume that the array is in ascending order.

e.g., 2 4 7 10 11 45 50 59 60 66 69 70 79

The binary search first compares the key with the element in the middle of the array.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Binary Search, cont.

Consider the following three cases:

- If the key is less than the middle element, you only need to search the key in the first half of the array.

- If the key is equal to the middle element, the search ends with a match.

- If the key is greater than the middle element, you only need to search the key in the second half of the array.

Uploaded By: Jibreel Bornat

From Idea to Soluton

```java
/** Use binary search to find the key in the list */
public static int binarySearch(int[] list, int key) {
  int low = 0;
  int high = list.length - 1;

  while (high >= low) {
    int mid = (low + high) / 2;
    if (key < list[mid])
      high = mid - 1;
    else if (key == list[mid])
      return mid;
    else
      low = mid + 1;
  }

  return -1 - low;
}
```

 Uploaded By: Jibreel Bornat

The Arrays.binarySearch Method

Since binary search is frequently used in programming, Java provides several overloaded binarySearch methods for searching a key in an array of int, double, char, short, long, and float in the java.util.Arrays class. For example, the following code searches the keys in an array of numbers and an array of characters.

```
int[] list = {2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79};
System.out.println("Index is " +
   java.util.Arrays.binarySearch(list, 11));
```

Return is 4

```
char[] chars = {'a', 'c', 'g', 'x', 'y', 'z'};
System.out.println("Index is " +
   java.util.Arrays.binarySearch(chars, 't'));
```

Return is –4 (insertion point is 3, so return is -3-1)

For the binarySearch method to work, the array must be pre-sorted in increasing order.

Uploaded By: Jibreel Bornat

Sorting Arrays

Sorting, like searching, is also a common task in computer programming. Many different algorithms have been developed for sorting. This section introduces a simple, intuitive sorting algorithms: *selection sort*.

Uploaded By: Jibreel Bornat
36

Selection Sort

Selection sort finds the smallest number in the list and places it first. It then finds the smallest number remaining and places it second, and so on until the list contains only a single number.

Select 1 (the smallest) and swap it with 2 (the first) in the list.

swap

2 9 5 4 8 1 6

The number 1 is now in the correct position and thus no longer needs to be considered.

swap

1 9 5 4 8 2 6

Select 2 (the smallest) and swap it with 9 (the first) in the remaining list.

The number 2 is now in the correct position and thus no longer needs to be considered.

swap

1 2 5 4 8 9 6

Select 4 (the smallest) and swap it with 5 (the first) in the remaining list.

The number 4 is now in the correct position and thus no longer needs to be considered.

1 2 4 5 8 9 6

5 is the smallest and in the right position. No swap is necessary.

The number 5 is now in the correct position and thus no longer needs to be considered.

swap

1 2 4 5 8 9 6

Select 6 (the smallest) and swap it with 8 (the first) in the remaining list.

The number 6 is now in the correct position and thus no longer needs to be considered.

swap

1 2 4 5 6 9 8

Select 8 (the smallest) and swap it with 9 (the first) in the remaining list.

The number 8 is now in the correct position and thus no longer needs to be considered.

1 2 4 5 6 8 9

Since there is only one element remaining in the list, the sort is completed.

Uploaded By: Jibreel Bornat

37

The Code

/** The method for sorting the numbers */

```java
public static void selectionSort(double[] list) {
  for (int i = 0; i < list.length; i++) {
    // Find the minimum in the list[i..list.length-1]
    double currentMin = list[i];
    int currentMinIndex = i;
    for (int j = i + 1; j < list.length; j++) {
      if (currentMin > list[j]) {
        currentMin = list[j];
        currentMinIndex = j;
      }
    }

    // Swap list[i] with list[currentMinIndex] if necessary;
    if (currentMinIndex != i) {
      list[currentMinIndex] = list[i];
      list[i] = currentMin;
    }
  }
}
```

Invoke it

selectionSort(yourList)

The Arrays.sort Method

Since sorting is frequently used in programming, Java provides several overloaded sort methods for sorting an array of int, double, char, short, long, and float in the java.util.Arrays class. For example, the following code sorts an array of numbers and an array of characters.

```
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
java.util.Arrays.sort(numbers);


char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};
java.util.Arrays.sort(chars);
```

Java 8 now provides Arrays.parallelSort(list) that utilizes the multicore for fast sorting.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

The Arrays.toString(list) Method

The Arrays.toString(list) method can be used to return a string representation for the list.

Uploaded By: Jibreel Bornat 40

Main Method Is Just a Regular Method

You can call a regular method by passing actual parameters. Can you pass arguments to <u>main</u>? Of course, yes. For example, the main method in class <u>B</u> is invoked by a method in <u>A</u>, as shown below:

```
public class A {
  public static void main(String[] args) {
    String[] strings = {"New York",
      "Boston", "Atlanta"};
    B.main(strings);
  }
}
```

```
class B {
  public static void main(String[] args) {
    for (int i = 0; i < args.length; i++)
      System.out.println(args[i]);
  }
}
```

41

Command-Line Parameters

```
class TestMain {
  public static void main(String[] args) {
    ...
    }
}


java TestMain arg0 arg1 arg2 ... argn
```

Uploaded By: Jibreel Bornat

Processing Command-Line Parameters

In the main method, get the arguments from `args[0], args[1], ..., args[n],` which corresponds to `arg0, arg1, ..., argn` in the command line.

Uploaded By: Jibreel Bornat   43

Problem: Calculator

- Objective: Write a program that will perform binary operations on integers. The program receives three parameters: an operator and two integers.

java Calculator 2 + 3

java Calculator 2 - 3

Calculator   java Calculator 2 / 3

Run   java Calculator 2 . 3

Uploaded By: Jibreel Bornat   44

```java
public class Calculator
    public static void main(String[] args) {
        if (args.length != 3) {
        System.out.println( "Usage: java Calculator operand1 operator operand2");
        System.exit(1);
        }

        int result = 0;
        switch (args[1].charAt(0)) {
            case '+': result = Integer.parseInt(args[0]) + Integer.parseInt(args[2]); break;
            case '-': result = Integer.parseInt(args[0]) - Integer.parseInt(args[2]); break;
            case '.': result = Integer.parseInt(args[0]) * Integer.parseInt(args[2]); break;
            case '/': result = Integer.parseInt(args[0]) / Integer.parseInt(args[2]);
        } // Display result
        System.out.println(args[0] + ' ' + args[1] + ' ' + args[2] + " = " + result);
    }
}
```

Uploaded By: Jibreel Bornat

Array of Objects

```java
Circle[] circleArray = new Circle[10];
```

To initialize `circleArray`, you can use a `for` loop as follows:

```java
for (int i = 0; i < circleArray.length; i++) {
    circleArray[i] = new Circle();
}
```

An array of objects is actually an *array of reference variables*. So invoking circleArray[1].getArea() involves two levels of referencing as shown in the next figure. circleArray references to the entire array. circleArray[1] references to a Circle object.

Array of Objects, cont.

```
Circle[] circleArray = new Circle[10];
```

Uploaded By: Jibreel Bornat   47
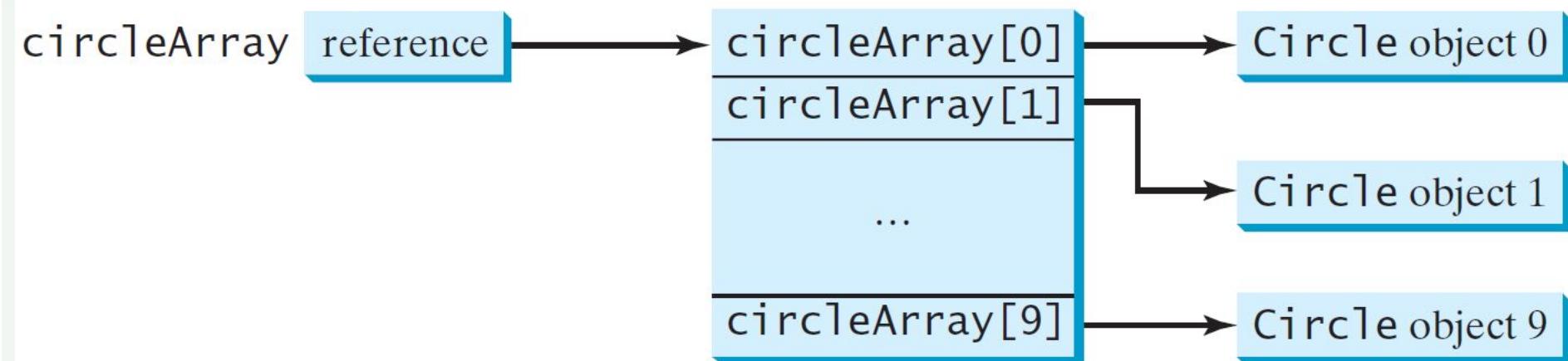
Declaring Variables of Two-dimensional Arrays and Creating Two-dimensional Arrays

```
int[][] matrix = new int[10][10];
 or
int matrix[][] = new int[10][10];
matrix[0][0] = 3;

for (int i = 0; i < matrix.length; i++)
  for (int j = 0; j < matrix[i].length; j++)
    matrix[i][j] = (int)(Math.random() * 1000);

double[][] x;
```

Uploaded By: Jibreel Bornat

Declaring, Creating, and Initializing Using Shorthand Notations

You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Same as

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

Uploaded By: Jibreel Bornat

Lengths of Two-dimensional Arrays

int[][] x = new int[3][4];

Uploaded By: Jibreel Bornat 50
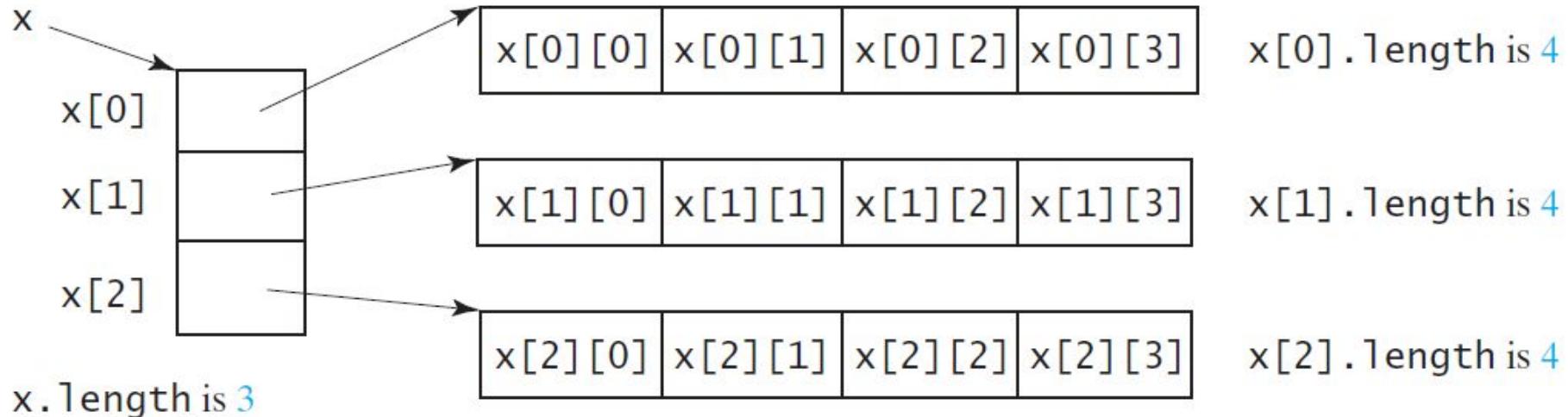
Lengths of Two-dimensional Arrays, cont.

```
int[][] array = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9},
    {10, 11, 12}
};
```

array.length
array[0].length
array[1].length
array[2].length
array[3].length

array[4].length   ArrayIndexOutOfBoundsException

Uploaded By: Jibreel Bornat   51

Ragged Arrays

Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

```
int[][] matrix = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5}
};
```

matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1

Uploaded By: Jibreel Bornat

Ragged Arrays, cont.

```
int[][] triangleArray = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5}
};
```
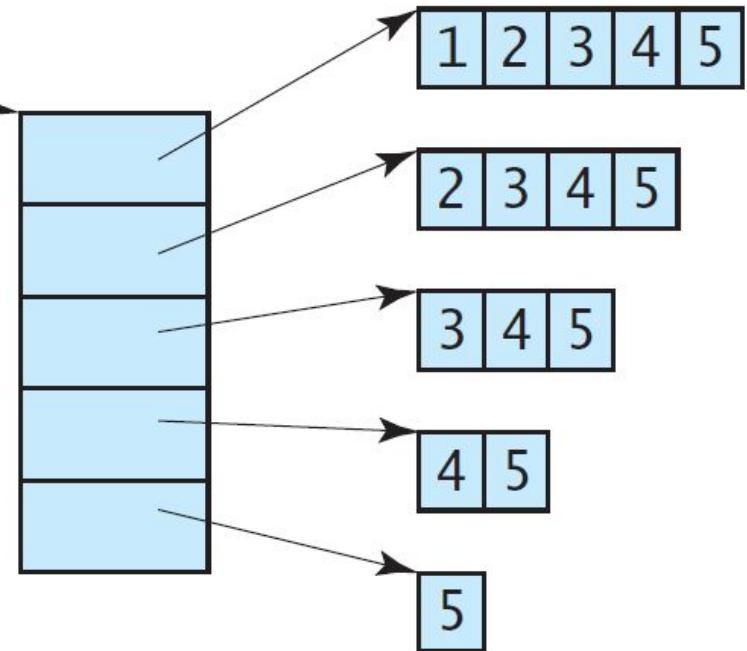
Uploaded By: Jibreel Bornat 53

Initializing arrays with input values

```java
java.util.Scanner input = new Scanner(System.in);
System.out.println("Enter " + matrix.length + " rows and " +
  matrix[0].length + " columns: ");
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = input.nextInt();
  }
}
```

Uploaded By: Jibreel Bornat

Initializing arrays with random values

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = (int)(Math.random() * 100);
  }
}
```

Uploaded By: Jibreel Bornat

Printing arrays

```java
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    System.out.print(matrix[row][column] + " ");
  }

  System.out.println();
}
```

Uploaded By: Jibreel Bornat

56

Passing Arrays to Methods

```java
public class PassTwoDimensionalArray {
    public static void main(String[] args) {
        int[][] m = getArray(); // Get an array // Display sum of elements
        System.out.println("\nSum of all elements is " + sum(m));
    }

    public static int[][] getArray() { // Create a Scanner
        Scanner input = new Scanner(System.in); // Enter array values
        int[][] m = new int[3][4];
        System.out.println("Enter " + m.length + " rows and " +
            m[0].length + " columns: ");
        for (int i = 0; i < m.length; i++)
            for (int j = 0; j < m[i].length; j++)
                m[i][j] = input.nextInt();
        return m;
    }

    public static int sum(int[][] m) {
        int total = 0;
        for (int row = 0; row < m.length; row++) {
            for (int column = 0; column < m[row].length; column++) {
                total += m[row][column];
            }
        }
        return total;
    }
}
```

Run

PassTwoDimensionalArray

Uploaded By: Jibreel Bornat

Multidimensional Arrays

Occasionally, you will need to represent n-dimensional data structures. In Java, you can create n-dimensional arrays for any integer n.

The way to declare two-dimensional array variables and create two-dimensional arrays can be generalized to declare n-dimensional array variables and create n-dimensional arrays for n >= 3.

 Uploaded By: Jibreel Bornat

Multidimensional Arrays

```
double[][][] scores = {
    {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
    {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
    {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
    {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
    {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
    {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}
};
```

Which student	Which exam	Multiple-choice or essay

```
scores[ i ] [ j ] [ k ]
```

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

59
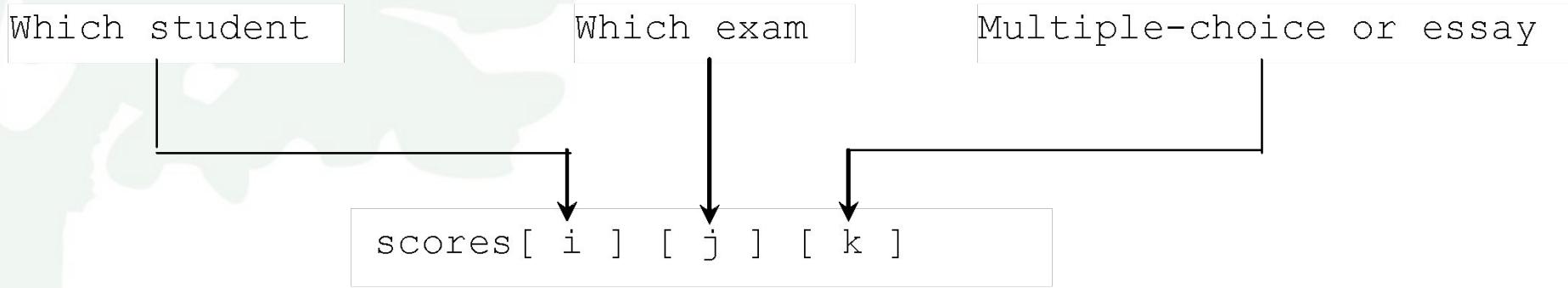
Problem: Calculating Total Scores

Objective: write a program that calculates the total score for students in a class. Suppose the scores are stored in a three-dimensional array named <u>scores</u>. The first index in <u>scores</u> refers to a student, the second refers to an exam, and the third refers to the part of the exam. Suppose there are 7 students, 5 exams, and each exam has two parts--the multiple-choice part and the programming part. So, <u>scores[i][j][0]</u> represents the score on the multiple-choice part for the <u>i</u>'s student on the <u>j</u>'s exam. Your program displays the total score for each student.

TotalScore   Run

STUDENTS-HUB.com   Uploaded By: Jibreel Bornat

```java
public class TotalScore { /** Main method */
    public static void main(String[] args) {
        double[][][] scores =
        { {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
        {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
        {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
        {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
        {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
        {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}},
        {{1.5, 29.5}, {6.4, 22.5}, {14, 30.5}, {10, 30.5}, {16, 6.0}}};
        // Calculate and display total score for each student

        for (int i = 0; i < scores.length; i++) {
            double totalScore = 0;
            for (int j = 0; j < scores[i].length; j++)
                for (int k = 0; k < scores[i][j].length; k++)
                    totalScore += scores[i][j][k];
            System.out.println("Student " + i + "'s score is " + totalScore);
        }
    }
}
```