15. Arrays and Methods

Java

Summer 2008

Instructor: Dr. Masoud Yaghini

Outline

- Passing Arrays to Methods
- Returning an Array from a Method
- Variable-Length Argument Lists
- References

Passing Arrays to Methods

Passing Arrays to Methods

- The following method displays the elements in an int array:

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

- Invoke the printArray method to display 3, 1, 2, 6, 4, and 2:

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

Anonymous Array

The statement

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

- creates an array using the following syntax:

```
new dataType[]{value0, value1, ..., valuek};
```

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

Pass By Value

- Java uses ***pass by value*** to pass parameters to a method. There are important differences between passing the values 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.

Simple Example

```
1  public class PassingArrayToMethod {
2    public static void main(String[] args) {
3      int x = 1; // x represents an int value
4      int[] y = new int[10]; // y represents an array of int values
5
6      m(x, y); // Invoke m with arguments x and y
7      System.out.println("x is " + x);
8      System.out.println("y[0] is " + y[0]);
9    }
10
11   public static void m(int number, int[] numbers) {
12     number = 1001; // Assign a new value to number
13     numbers[0] = 5555; // Assign a new value to numbers[0]
14   }
15 }
```

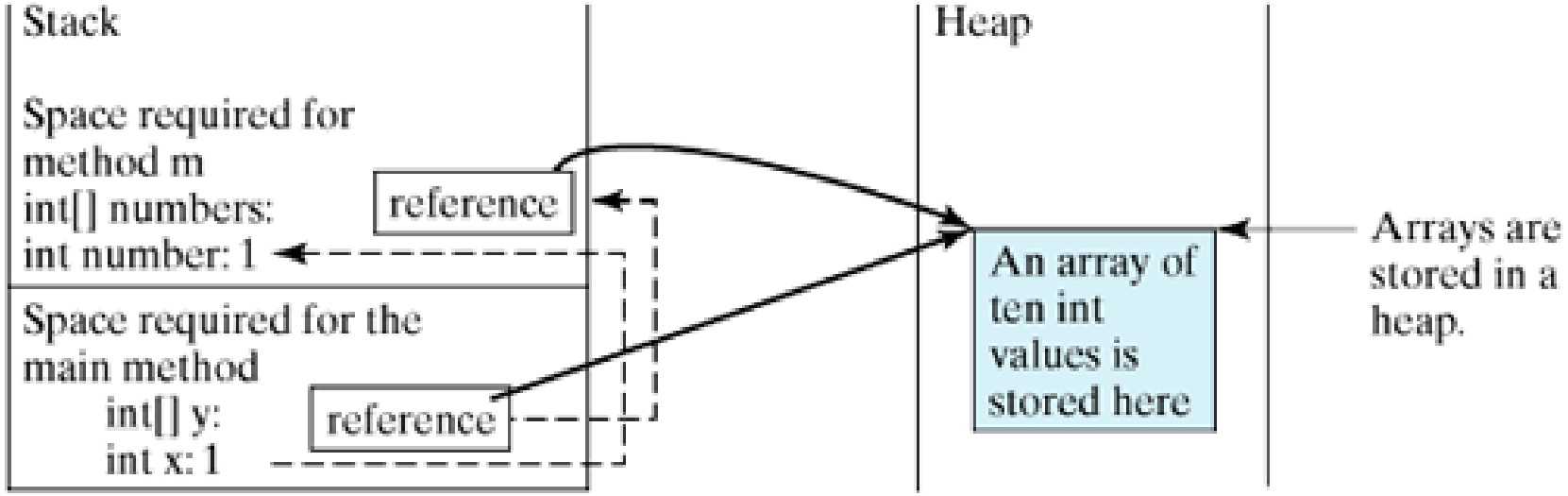
- Output?

  x is 1

  y[0] is 5555

Passing Arrays to Methods

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



Example: Passing Arrays as Arguments

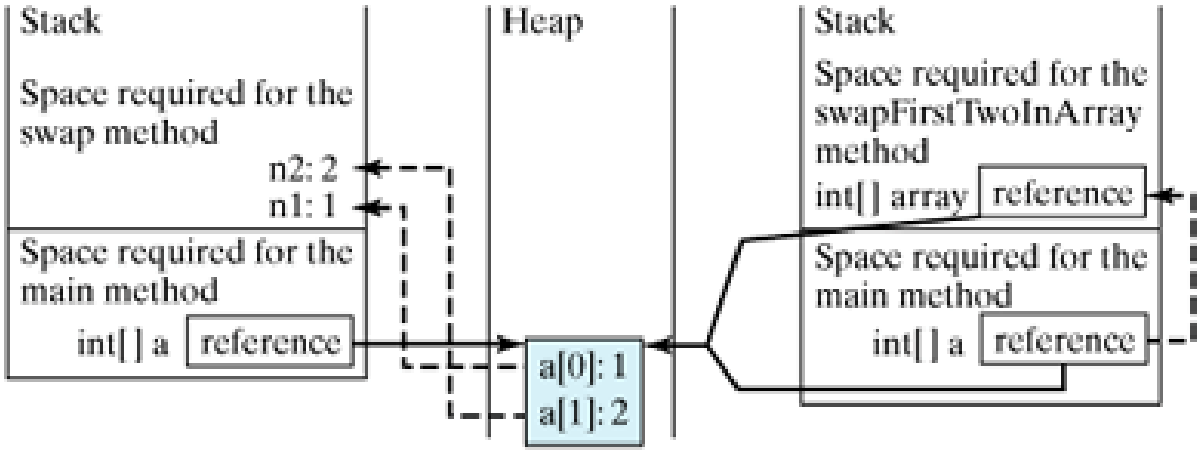
```
1  public class TestPassArray {
2    /** Main method */
3    public static void main(String[] args) {
4      int[] a = {1, 2};
5
6      // Swap elements using the swap method
7      System.out.println("Before invoking swap");
8      System.out.println("array is {" + a[0] + ", " + a[1] + "}");
9      swap(a[0], a[1]);
10     System.out.println("After invoking swap");
11     System.out.println("array is {" + a[0] + ", " + a[1] + "}");
12
13     // Swap elements using the swapFirstTwoInArray method
14     System.out.println("Before invoking swapFirstTwoInArray");
15     System.out.println("array is {" + a[0] + ", " + a[1] + "}");
16     swapFirstTwoInArray(a);
17     System.out.println("After invoking swapFirstTwoInArray");
18     System.out.println("array is {" + a[0] + ", " + a[1] + "}");
19   }
20
```

Example: Passing Arrays as Arguments

```
21    /** Swap two variables */
22    public static void swap(int n1, int n2) {
23      int temp = n1;
24      n1 = n2;
25      n2 = temp;
26    }
27
28    /** Swap the first two elements in the array */
29    public static void swapFirstTwoInArray(int[] array) {
30      int temp = array[0];
31      array[0] = array[1];
32      array[1] = temp;
33    }
34  }
```

- Output?

Example: Passing Arrays as Arguments



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.

Example: Passing Arrays as Arguments

- Output:

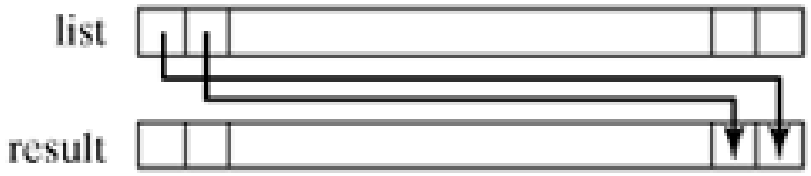
```
Before invoking swap
array is {1, 2}
After invoking swap
array is {1, 2}
Before invoking swapFirstTwoInArray
array is {1, 2}
After invoking swapFirstTwoInArray
array is {2, 1}
```

Returning an Array from a Method

Returning an Array from a Method

- The method shown below returns an array that is the reversal of another array:

```
1 public static int[] reverse(int[] list) {
2   int[] result = new int[list.length];
3
4   for (int i = 0, j = result.length - 1;
5        i < list.length; i++, j--) {
6     result[j] = list[i];
7   }
8
9   return result;
10 }
```



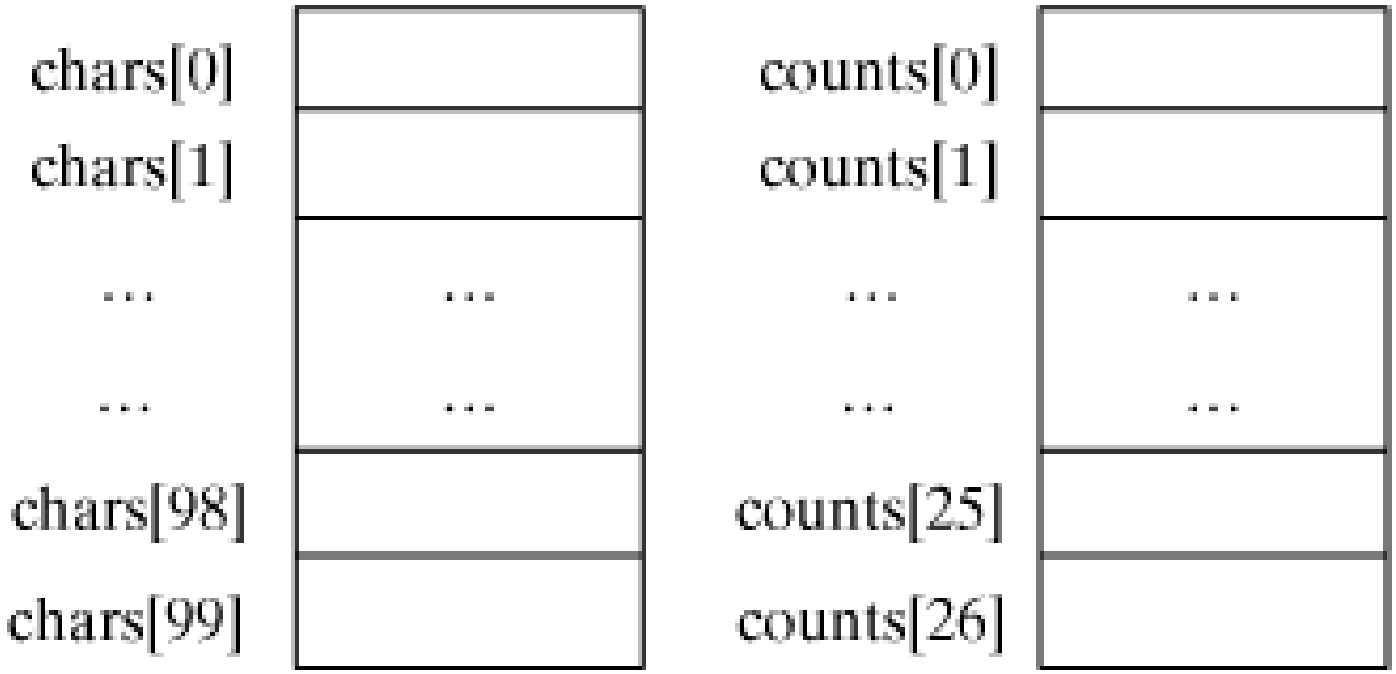
- The method can be invoked as below:

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

- Trace the reverse Method

Example: Counting the Occurrences of Each Letter

- Generate one hundred lowercase letters randomly and assign them to an array of characters and count the occurrences of each letter in the array.



Example: Counting the Occurrences of Each Letter

```
1  public class CountLettersInArray {
2    /** Main method */
3    public static void main(String args[]) {
4
5      // Declare and create an array
6      char[] chars = createArray();
7
8      // Display the array
9      System.out.println("The lowercase letters are:");
10     displayArray(chars);
11
12     // Count the occurrences of each letter
13     int[] counts = countLetters(chars);
14
15     // Display counts
16     System.out.println();
17     System.out.println("The occurrences of each letter are:");
18     displayCounts(counts);
19   }
20
```

Example: Counting the Occurrences of Each Letter

```
21    /** Create an array of characters */
22    public static char[] createArray() {
23      // Declare an array of characters and create it
24      char[] chars = new char[100];
25
26      // Create lowercase letters randomly and assign
27      // them to the array
28      for (int i = 0; i < chars.length; i++)
29        chars[i] = getRandomLowerCaseLetter();
30
31      // Return the array
32      return chars;
33    }
34
35    /** Generate a random lowercase letter */
36    public static char getRandomLowerCaseLetter() {
37      return (char)('a' + (Math.random() * ('z' - 'a' + 1)));
38
39    }
40
```

Example: Counting the Occurrences of Each Letter

```
41    /** Display the array of characters */
42    public static void displayArray(char[] chars) {
43      // Display the characters in the array 20 on each line
44      for (int i = 0; i < chars.length; i++) {
45        if ((i + 1) % 20 == 0)
46          System.out.println(chars[i] + " ");
47        else
48          System.out.print(chars[i] + " ");
49      }
50    }
51
52    /** Count the occurrences of each letter */
53    public static int[] countLetters(char[] chars) {
54      // Declare and create an array of 26 int
55      int[] counts = new int[26];
56
57      // For each lowercase letter in the array, count it
58      for (int i = 0; i < chars.length; i++)
59        counts[chars[i] - 'a']++;
60
61      return counts;
62    }
```

Example: Counting the Occurrences of Each Letter

```
63
64      /** Display counts */
65      public static void displayCounts(int[] counts) {
66        for (int i = 0; i < counts.length; i++) {
67          if ((i + 1) % 10 == 0)
68            System.out.println(counts[i] + " " + (char)(i + 'a'));
69          else
70            System.out.print(counts[i] + " " + (char)(i + 'a') + " ");
71        }
72      }
73    }
```

Returning an Array from a Method

- Output:

```
The lowercase letters are:
h y w f m f p h k r j r q l x d d o r f
o w w b f b r b c n z c y j z v t y q x
i e d q o f w o s c a n c d q d l s m e
y c d e r c g e m u o e i f d s f q b q
p w h p f w p u t w m h q z v f b o v f

The occurrences of each letter are:
1 a 5 b 6 c 7 d 5 e 10 f 1 g 4 h 2 i 2 j
1 k 2 l 4 m 2 n 6 o 4 p 7 q 5 r 3 s 2 t
2 u 3 v 7 w 2 x 4 y 3 z
```

Variable-Length Argument Lists

Variable-Length Argument Lists

- Java enables you to pass a variable number of arguments of the same type to a method.
- The parameter in the method is declared as follows:

  ```
  typeName... parameterName
  ```

- Only one variable-length parameter may be specified in a method
- This parameter must be the last parameter. Any regular parameters must precede it.
- Java treats a variable-length parameter as an array.

Example

```
1  public class VarargsDemo {
2    public static void main(String args[]) {
3      printMax(34, 3, 3, 2, 55.5);
4      printMax(new double[]{1, 2, 3});
5      printMax();
6    }
7
8    public static void printMax(double... numbers) {
9      if (numbers.length == 0) {
10       System.out.println("No argument passed");
11       return;
12     }
13
14     double result = numbers[0];
15
16     for (int i = 1; i < numbers.length; i++)
17       if (numbers[i] > result)
18         result = numbers[i];
19
20     System.out.println("The max value is " + result);
21   }
22 }
```

Example

- Output:

```
The max value is 55.5
The max value is 3.0
No argument passed
```

References

References

- Y. Daniel Liang, **Introduction to Java Programming**, Sixth Edition, Pearson Education, 2007. (Chapter 5)
- S. Zakhour and et. al., **The Java Tutorial: A Short Course on the Basics**, 4th Edition, Prentice Hall, 2006. (Chapter 3)

The End