

Passing Arrays to Methods

Hui Chen

Department of Computer & Information Science

Brooklyn College

Objectives

- To develop and invoke methods with array arguments and return values (§§7.6–7.8).
- To copy contents from one array to another (§7.5)
- To define a method with a variable-length argument list (§7.9).

Passing Arrays to Methods

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

Invoke the method

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

Invoke the method

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

Anonymous array

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

Pass By Value

- Java uses pass by value to pass arguments to a method.
- However, there are important differences between passing a value of variables of primitive data types and passing arrays.

Pass by Value: Primitive Data Types

- 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.

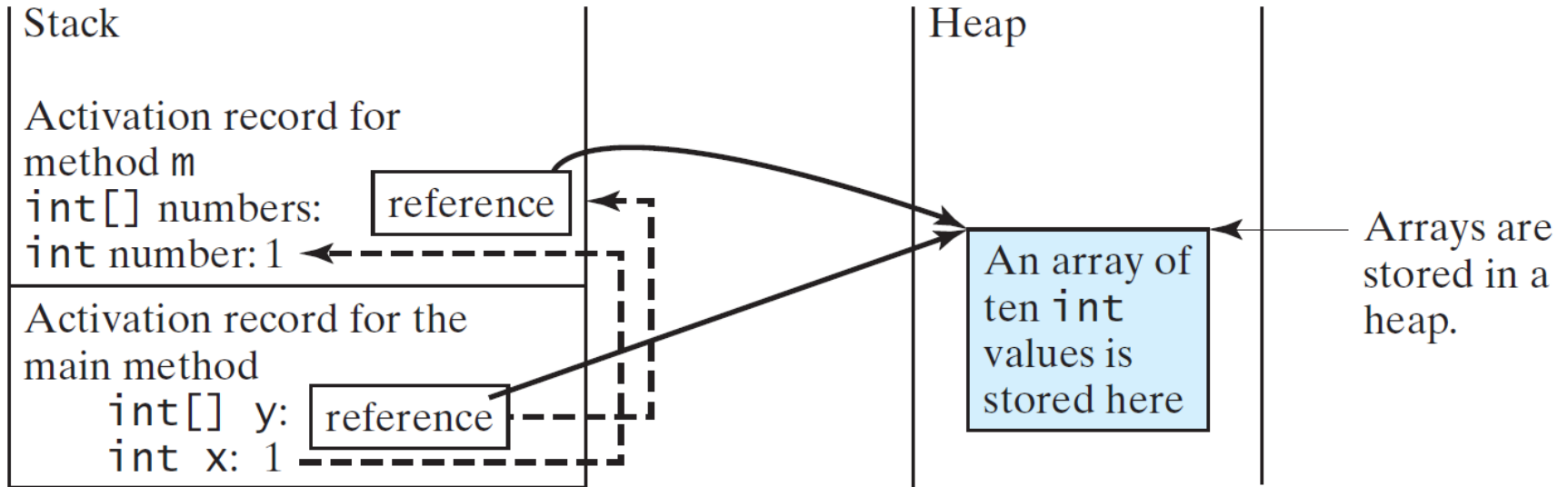
Pass by Value: Array

- 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.

Pass by Value: Example

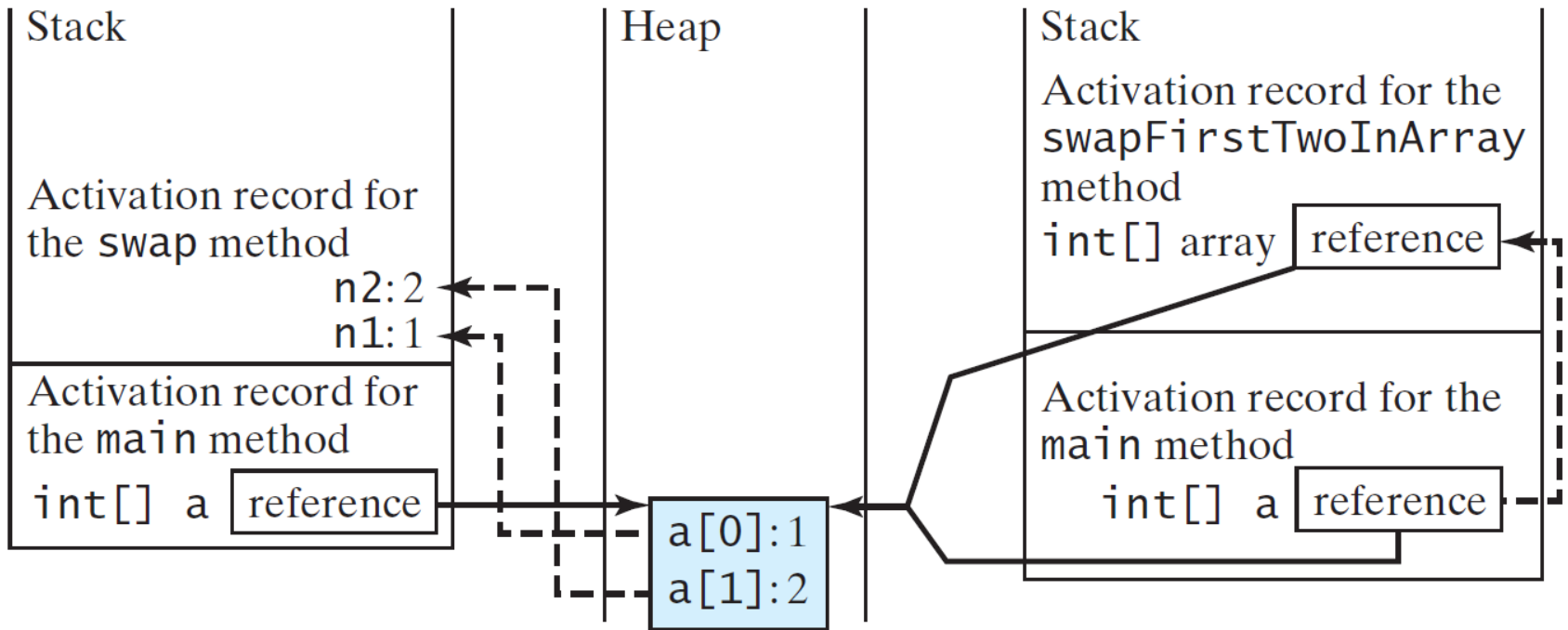
```
public class Test {  
    public static void main(String[] args) {  
        int x = 1; // x represents an int value  
        int[] y = new int[10]; // y represents an array of int values  
        m(x, y); // Invoke m with arguments x and y  
        System.out.println("x is " + x);  
        System.out.println("y[0] 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



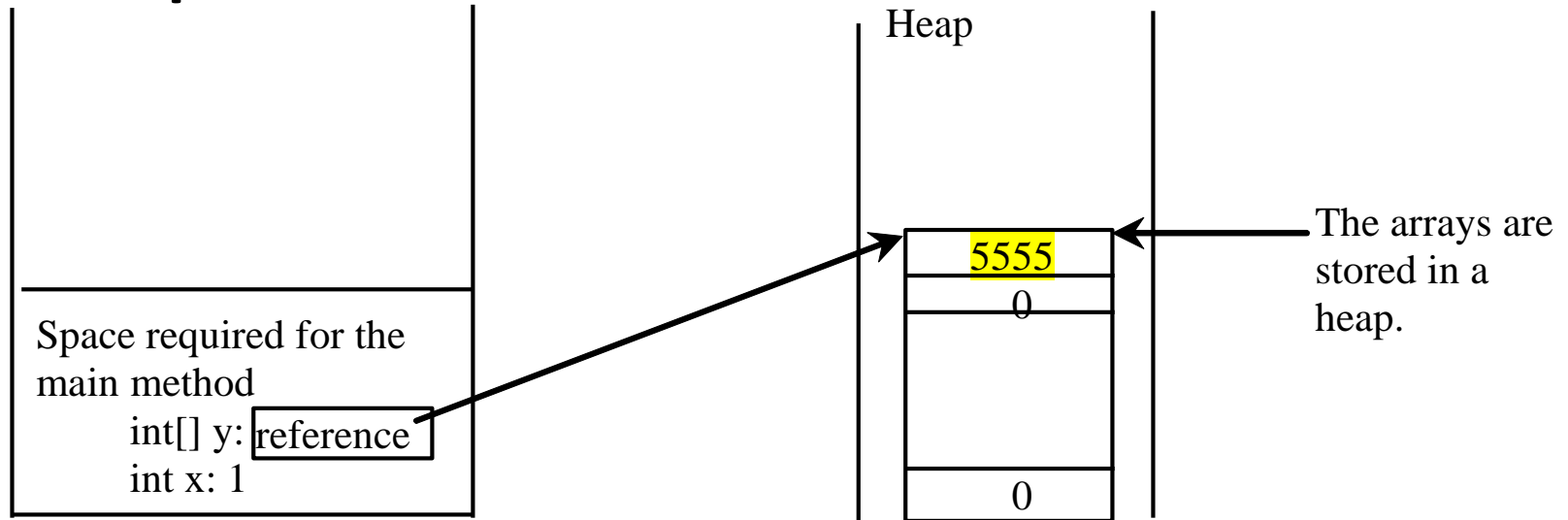
- When invoking `m(x, y)`, the values of `x` and `y` are passed to `number` and `numbers`.
- Since `y` contains the reference value to the array, `numbers` now contains the same reference value to the same array

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

Heap

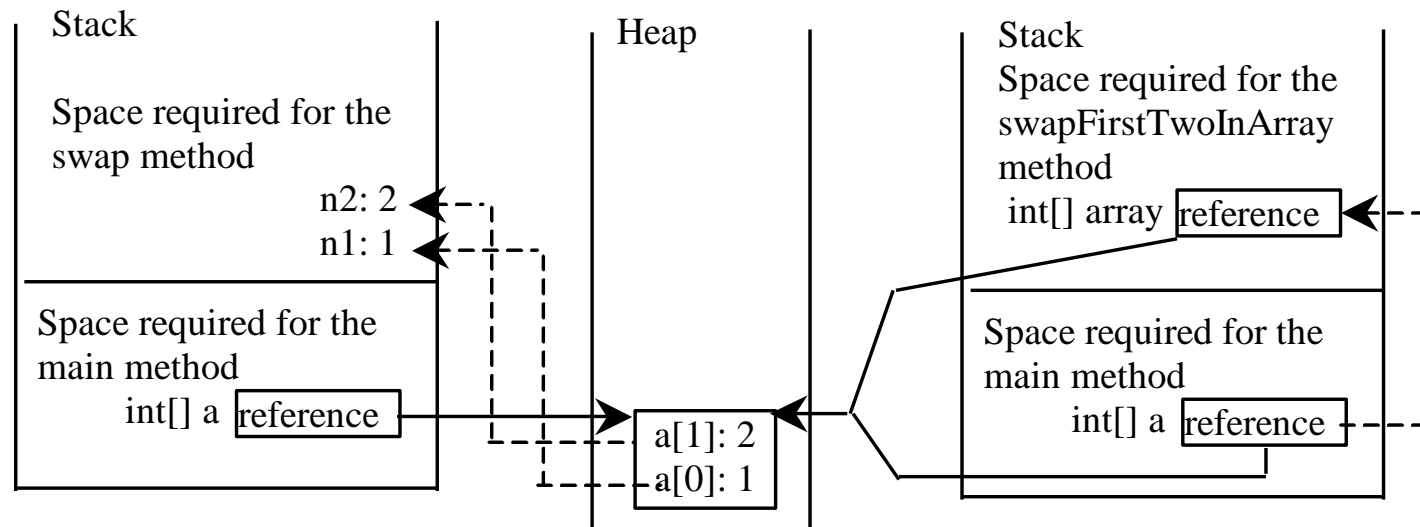


- The JVM stores the array in an area of memory, called *heap*, which is used for dynamic memory allocation where blocks of memory are allocated and freed in an arbitrary order

Questions?

Example Problem. Passing Arrays vs. Passing Primitive Data Types

- Compare `swap(int n1, int n2)` and `swapFirstTwoInArray(int[] array)`



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

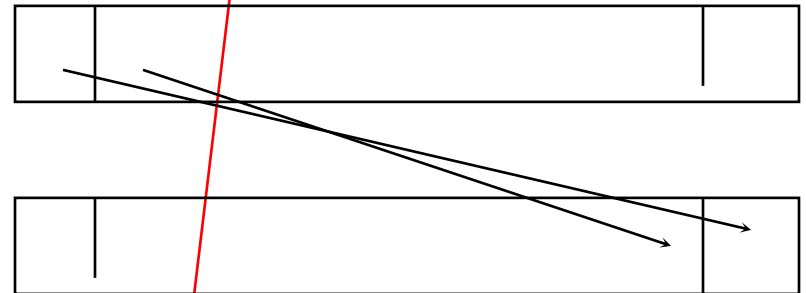
The arrays are stored in a heap.

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

Questions?

Returning an Array from a Method

```
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;  
}
```

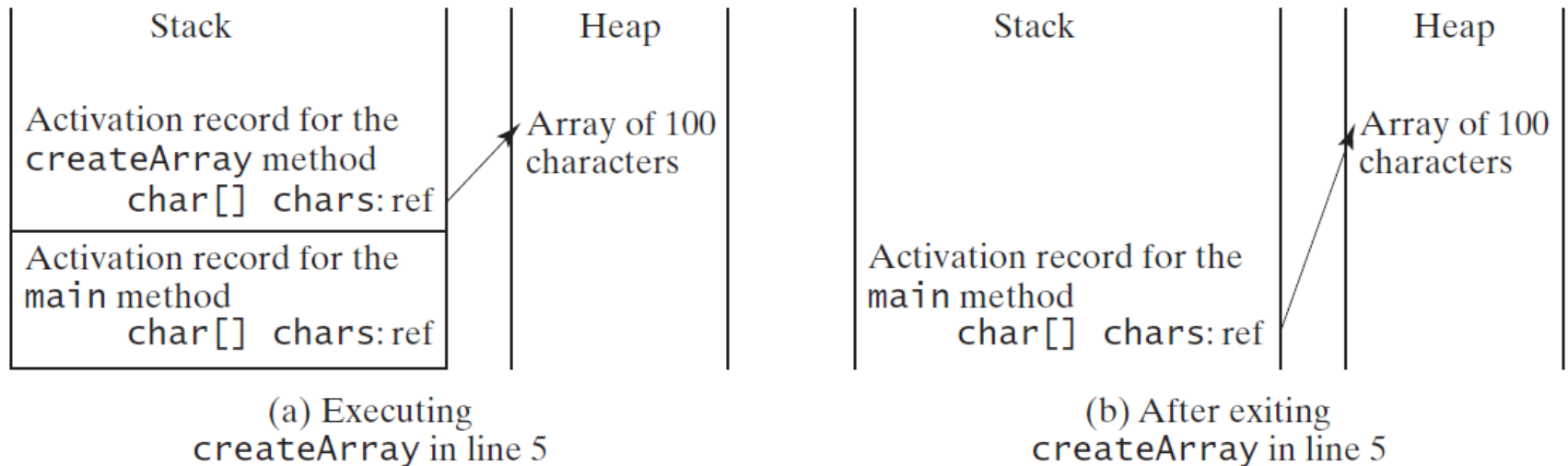


```
int[] list1 = {1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

Questions

Problem. Counting Occurrence of Each Letter

- Generate 100 lowercase letters randomly and assign to an array of characters.
- Count the occurrence of each letter in the array.



Variable-Length Arguments and Array

- You can pass a variable number of arguments of the same type to a method

Example

```
public class VarArgsDemo {
    public static void main(String[] args) {
        printMax(34, 3, 3, 2, 56.5);
        printMax(new double[]{1, 2, 3});
    }

    public static void printMax(double... numbers) {
        if (numbers.length == 0) {
            System.out.println("No argument passed");
            return;
        }

        double result = numbers[0];

        for (int i = 1; i < numbers.length; i++)
            if (numbers[i] > result)
                result = numbers[i];

        System.out.println("The max value is " + result);
    }
}
```

Questions?