

South Louisiana Community College
ASDV 1220, Programming Fundamentals

Learning Objectives

After completion of this lab, you should be able to

1. Understand arrays as parameters and arguments of methods.
2. Understand arrays as return types.
3. Understand array copying, sorting and traversing.

Create project Lab17

Problem 1

Create a class **ArrayShift1**. Add the following code that shifts the array 1 position to the left.

```
1  package lab17;
2
3  public class ArrayShift1
4  {
5      public static void main(String...args)
6      {
7          int[] ar = {1, 2, 3, 4, 5};
8          int temp = ar[0];
9          for ( int i=1; i < ar.length ;++i )
10             ar[i-1] = ar[i];
11
12             ar[ ar.length-1] = temp;
13
14             for( int value: ar )
15                 System.out.print( value + " ");
16
17         }
18     }
```

Problem 2

Create a class **ArrayShift2**. Modify the code of ArrayShift1 to shift the array 1 position to the RIGHT. The printout should be 5 1 2 3 4

Problem 3

Create a class **CopyArray1**. Add the following code that copies the source array into the target array and prints the target array.

```
1  package lab17;
2
3  public class CopyArray1
4  {
5  public static void main(String...args)
6  {
7
8      int[] sourceArray = {1, 2, 3, 4, 5};
9      int[] targetArray = new int[sourceArray.length];
10
11     for (int i = 0; i < sourceArray.length; i++)
12     {
13         targetArray[i] = sourceArray[i];
14     }
15
16     for (int i = 0; i < sourceArray.length; i++)
17     {
18         System.out.print( targetArray[i] + " " );
19     }
20 }
```

Problem 4

Create a class **ArrayCopy2**. Modify the code of **ArrayCopy1** to copy the source array into the target array in reversed order. Print the target array from 0 to length-1 to display the values 5 4 3 2 1 .

Problem 5

Create a class **ArrayCopy3**. Modify the code of **ArrayCopy1** to copy the source array into the target array by using `System.arraycopy`. Read and UNDERSTAND the javadoc of the method `arraycopy` of `System`.

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

Problem 6

Create a class **ArrayCopy4**. Modify the code of **ArrayCopy3** to copy the last 3 elements of source array into the first 3 elements of target array by using `arraycopy` and by using a `for` loop.

Problem 7

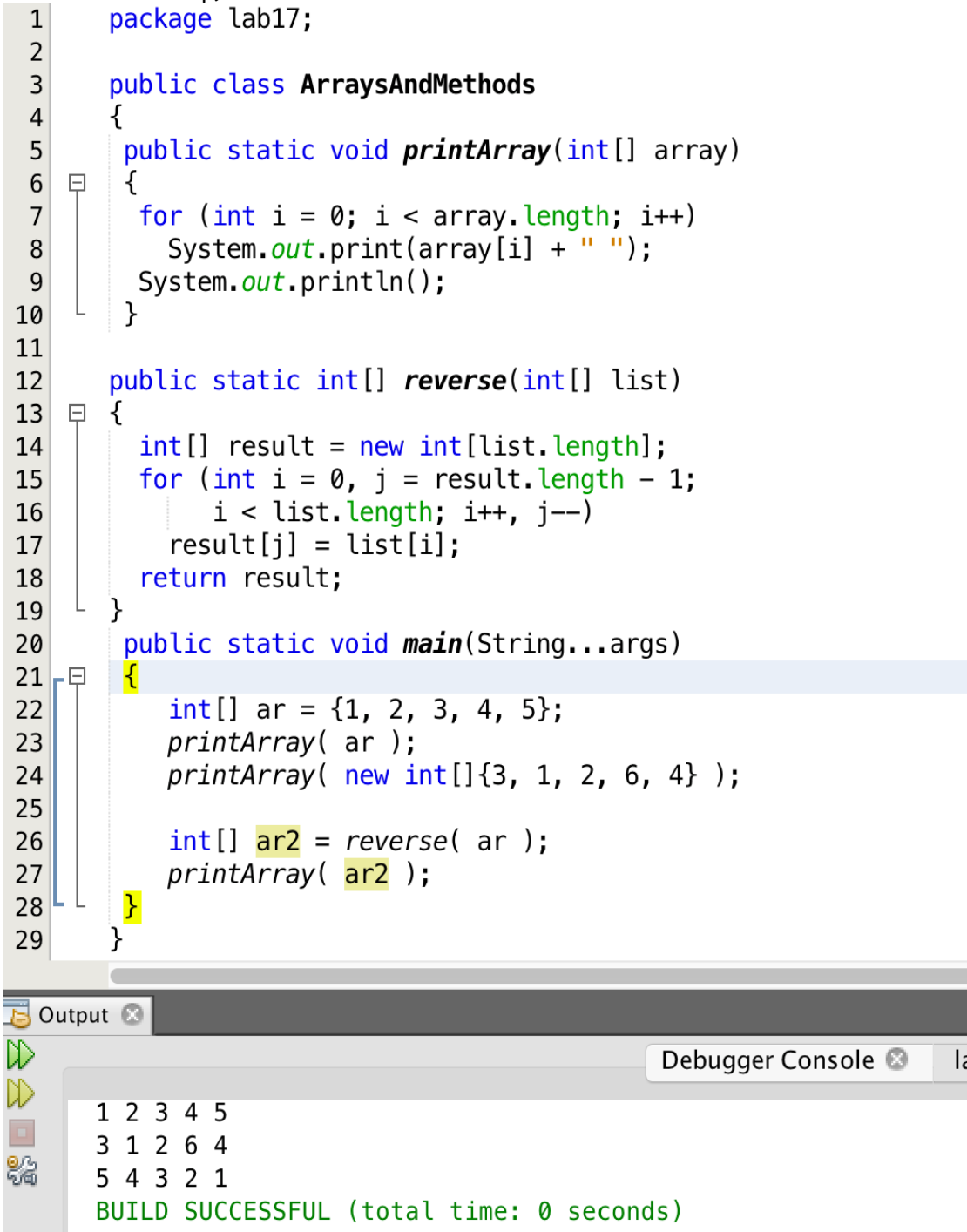
Create a class **ArraysAndMethods**. Add the following code that passes the array *ar* as argument to the method. Observe line 16 where we PASS an ANONYMOUS ARRAY as an argument. The **address** of the anonymous array is passed as an argument to the method. It is a call by value (copy of the address).

```
1  package lab17;
2
3  public class ArraysAndMethods
4  {
5      public static void printArray(int[] array)
6      {
7          for (int i = 0; i < array.length; i++)
8              System.out.print(array[i] + " ");
9              System.out.println();
10     }
11     public static void main(String...args)
12     {
13
14         int[] ar = {1, 2, 3, 4, 5};
15         printArray( ar );
16         printArray( new int[]{3, 1, 2, 6, 4} );
17     }
18 }
```

Problem 8

Modify class **ArraysAndMethods**. Add the method **reverse** as shown in lines 12 to19, and then call method **reverse** as shown in line 26. Observe that method **reverse** returns an array **int[]** and we save the ADDRESS of the returned array INTO a variable-reference named **ar2**. Then we print **ar2**. Understand how we return arrays from methods (we return the address of the array which is located somewhere on the heap). Understand LINE 26.

```
1 package lab17;
2
3 public class ArraysAndMethods
4 {
5     public static void printArray(int[] array)
6     {
7         for (int i = 0; i < array.length; i++)
8             System.out.print(array[i] + " ");
9         System.out.println();
10    }
11
12    public static int[] reverse(int[] list)
13    {
14        int[] result = new int[list.length];
15        for (int i = 0, j = result.length - 1;
16             i < list.length; i++, j--)
17            result[j] = list[i];
18        return result;
19    }
20    public static void main(String...args)
21    {
22        int[] ar = {1, 2, 3, 4, 5};
23        printArray( ar );
24        printArray( new int[]{3, 1, 2, 6, 4} );
25
26        int[] ar2 = reverse( ar );
27        printArray( ar2 );
28    }
29 }
```



Output

Debugger Console

```
1 2 3 4 5
3 1 2 6 4
5 4 3 2 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problem 9

Modify class **ArraysAndMethods**, by adding the method **linearSearch** as shown in lines 19 to 26. The method `linearSearch` searches the array parameter for the key parameter. If the key is found in the array then the method *returns the index of the key element*, otherwise it *returns -1*. The method is called multiple times from inside the conditional statement of `main()`. Understand lines 35 to 38.

```
15     i < list.length; i++, j--)
16         result[j] = list[i];
17     return result;
18 }
19 public static int linearSearch(int[] list, int key)
20 {
21     int indexFound = -1;
22     for (int i = 0; i < list.length; ++i )
23         if ( list[i] == key )
24             indexFound = i;
25     return indexFound;
26 }
27 public static void main(String...args)
28 {
29     int[] ar = {1, 2, 3, 4, 5};
30     printArray( ar );
31     printArray( new int[]{3, 1, 2, 6, 4} );
32
33     int[] ar2 = reverse( ar );
34     printArray( ar2 );
35     System.out.println ( linearSearch( ar, 5) != 1 ?
36         "found at index" + Integer.toString ( linearSearch( ar, 5) ) :
37         "not found at index" + Integer.toString ( linearSearch( ar, 5) ));
38
39 }
40 }
41 }
```

Problem 9

Modify class **ArraysAndMethods**, by adding lines 39 to 48 as shown below. The code uses the utility method *sort* of the java class *Arrays* to sort an array of primitive type. Then, we print the sorted arrays.

```
27 public static void main(String...args)
28 {
29     int[] ar = {1, 2, 3, 4, 5};
30     printArray( ar );
31     printArray( new int[]{3, 1, 2, 6, 4} );
32
33     int[] ar2 = reverse( ar );
34     printArray( ar2 );
35     System.out.println ( linearSearch( ar, 5) != 1 ?
36         "found at index" + Integer.toString ( linearSearch( ar, 5) ) :
37         "not found at index" + Integer.toString ( linearSearch( ar, 5) ));
38
39     double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
40     java.util.Arrays.sort(numbers);
41     char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};
42     java.util.Arrays.sort(chars);
43     for ( double d: numbers )
44         System.out.print( d + " ");
45     System.out.println();
46     for ( char c: chars )
47         System.out.print( c + " ");
48     System.out.println();
49 }
```

Output

Debugger Console

lab17 (run)

```
run:
1 2 3 4 5
3 1 2 6 4
5 4 3 2 1
found at index4
1.9 2.9 3.4 3.5 4.4 6.0
4 A D F P a
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problem 10

Modify class **ArraysAndMethods**, by adding method *modifyTheArray* lines 27 to 31 as shown below. Then, call the method *modifyTheArray* by passing as argument the array *ar* which contains the numbers 1 2 3 4 5, as shown in line 54. Observe that when we print the array *ar* inside main the array has been already modified and each of each element is now incremented by 10. Also, observe line 30 where the index of the array is POST-incremented and there is no `++i` for the *for-increment*.

```
27 public static void modifyTheArray(int[] array)
28 {
29     for (int i = 0; i < array.length; )
30         array[i] = array[i++] + 10;
31 }
32 public static void main(String...args)
33 {
34     int[] ar = {1, 2, 3, 4, 5};
35     printArray( ar );
36     printArray( new int[]{3, 1, 2, 6, 4} );
37
38     int[] ar2 = reverse( ar );
39     printArray( ar2 );
40     System.out.println ( linearSearch( ar, 5) != 1 ?
41         "found at index" + Integer.toString ( linearSearch( ar, 5) ) :
42         "not found at index" + Integer.toString ( linearSearch( ar, 5) ));
43
44     double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
45     java.util.Arrays.sort(numbers);
46     char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};
47     java.util.Arrays.sort(chars);
48     for ( double d: numbers )
49         System.out.print( d + " ");
50     System.out.println();
51     for ( char c: chars )
52         System.out.print( c + " ");
53     System.out.println();
54     modifyTheArray( ar );
55     printArray( ar );
56 }
```

Problem 11

Create a class **ArraysAndMethodsGames**. Write a method that returns the average of an array with the following header:

```
public static double average(double[] array);
```

Write a method to initialize the array with the following header:

```
public static double[] readArray();
```

The *readArray* method prompts the user to enter ten double values and initializes a newly created array with these values. It returns the array to *main*. The method *main* calls the method *average* and prints the average. Method *main* prints the array as well.

Problem 12

Modify class **ArraysAndMethodsGames** by adding a method that finds the smallest element in an array of double values using the following header:

```
public static double min(double[] array);
```

Call it from main to test it with an array you create in main.