

Lab Manual for Data Structures

Lab-1

Development of Arrays (Static, Dynamic and Multi-dimensional) in JAVA



Department of Computing
STMU, Islamabad

Table of Contents

| | | |
|--------|--|----|
| 1. | Introduction..... | 3 |
| 1.1. | Basic Concepts about Data Structures | 3 |
| 1.2. | Arrays in JAVA | 3 |
| 1.3. | Dynamic Memory Allocation..... | 4 |
| 1.4. | Multi-Dimensional Arrays | 4 |
| 1.5. | Relevant Lecture Readings..... | 4 |
| 2. | Activity Time boxing | 4 |
| 3. | Objectives of the experiment..... | 5 |
| 4. | Concept Map | 5 |
| 4.1. | Arrays in JAVA | 5 |
| 4.2. | Initializing Arrays | 6 |
| 4.3. | Accessing values from an array | 6 |
| 4.4. | Static Arrays | 6 |
| 4.5. | Dynamic Memory Allocation in JAVA..... | 7 |
| 4.6. | Multi-Dimensional Arrays | 10 |
| 4.6.1. | Two Dimensional Arrays in JAVA | 10 |
| 4.6.2. | Three Dimensional Arrays in JAVA | 11 |
| 5. | Homework before Lab | 12 |
| 5.1. | Problem Solution Modeling | 12 |
| 5.2. | Problem description: | 12 |
| 6. | Procedures & Tools..... | 12 |
| 6.1. | Tools..... | 12 |
| 6.2. | Editing, Compiling and Running programs in NetBeans 21[Expected time = 5mins]..... | 12 |
| 6.3. | Walk through Task [Expected time = 20mins] | 13 |
| 7. | Practice Tasks..... | 15 |
| 7.5. | Out comes | 17 |
| 8. | Evaluation Task (Unseen) [Expected time = 60mins for two tasks]..... | 18 |
| 9. | Evaluation criteria..... | 18 |
| 10. | Further Readings | 18 |
| 10.1. | JAVA Tutorial Web Sites..... | 18 |
| 10.2. | Web sites containing supporting material | 18 |

Lab 01: Development of Arrays

1. Introduction

Objective of this lab is to provide you knowledge about development of arrays in JAVA. You have come to know about the basic concepts about JAVA language in courses: Computer programming and Object-oriented programming. You have learned about the statements used in this language like input/output statements, selection statements (**if..else**, **switch**) and iterative statements/loops (**while**, **do..while**, **for**). Further you will also have knowledge about **classes**, **encapsulation**, **inheritance** and **polymorphism**.

In JAVA, every application/program contains a function called `main()`, which is a global function. Execution of every program starts from this function, so you have to write this function in each JAVA program. You may write other functions also in your program beside `main()` function, which depends on modular solution you have developed to solve your problem.

1.1. Basic Concepts about Data Structures

A program is a set of instructions that processes data which is given as input to it. If we need to develop a good (efficient) program, we need to organize data in computer's memory in effective way. This organized data is called a **Data Structure**.

Study of computer science includes the study of how data (information) is organized in computer and how it can be manipulated and utilized by programs. It is extremely important for you to understand the concepts of data (information) organization and manipulation. Data structures for storing information are lists, stacks, queues, trees, graphs and tables.

1.2. Arrays in Java

In Java arrays are used to store such items whose data type is same. Normally arrays are used to represent lists, trees and graphs; which are types of data structure. We can perform different operations on arrays such as: inserting an item, removing an item, finding a value in array, performing mathematical operations on elements of array etc. If we want to initialize any global or local array by assigning some values to its element we can do so. We can access value of an element of array in our program as it was a normal variable. We are allowed to read and modify the values of these elements.

Static arrays are allocated memory at compile time and their size is fixed, it means it cannot be changed latter during program execution. If we want to change the size of array at program execution time, we will need to declare dynamic arrays. A dynamic array is allocated memory using **new** operator and is used in the same way as a static array. Dynamic arrays are accessed using pointers.

1.3. Dynamic Memory Allocation

Assume you have a program which is used to store information about the employees of some organization, when this program starts its execution we are not sure that how many number of employee records will be stored in this program. Therefore how much memory is required for this program is dependent on the fact that how many records we will enter when program will be executing. So memory required by program is to be estimated at program execution time, so we will need a memory allocation mechanism which is called dynamic memory allocation for this program, which should allocate and de allocate memory at runtime according to requirements of program.

1.4. Multi-Dimensional Arrays

Sometimes the data which we need to process in our program is more easily interpretable if represented in the form of a table. A table is a structure which contains some rows and some columns, on intersection of each row and a column some data is stored. Examples of such data representations may be a parcel weight-location chart used by some post office; a public transport rent chart and representation of a digital image etc. So we may easily understand this data representation if we use a two dimensional array to represent it (remember a two dimensional array is a special case of a multi-dimensional array).

Sometimes data to be represented is based on more than two dimensions, for example if we need to represent the sales of some departmental stores by product, year of sale and location wise, it can be easily represented by a three dimensional structure, which may be logically represented as a cube. To represent such data we will use three dimensional arrays.

1.5. Relevant Lecture Readings

- Revise Lecture No. 1 and 2 available at `\\dataserver\jinnah$` in the instructor's folder.
- From books: JAVA Data Structures by Nell Dale (Page 79-85) and Data structures using JAVA by D. S. Malik (Page 130-135).
- From books: JAVA Data Structures by Nell Dale (Page 99-102) and Data structures using JAVA by D. S. Malik (Page 138-145).

2. Activity Time boxing

Table 1 Activity Time Boxing

| Task No. | Activity Name | Activity time | Total Time |
|----------|---|--------------------------|------------|
| 5.1 | Design Evaluation | 15 mins | 15 mins |
| 6.2 | Editing and compiling a JAVA program in NetBeans 21 | 5mins | 5mins |
| 6.3 | Walk Through Task | 20mins | 20mins |
| 7 | Practice tasks | 30 mins for task 1,2 and | 70mins |

| | | | |
|---|-----------------|------------------------------|--------|
| | | 40 mins for task 3, 4 | |
| 9 | Evaluation Task | 60mins for all assigned task | 60mins |

3. Objectives of the experiment

- To get basic understanding of static and dynamic arrays in JAVA.
- To get basic understanding of dynamic memory allocation and multi-dimensional arrays in JAVA
- To write programs in JAVA using NetBeans environment.
- To get an understanding of identifying basic errors in a JAVA program by using debugging techniques.

4. Concept Map

This concept map will help students to understand the main concepts of topic covered in lab.

4.1. Arrays in JAVA

In JAVA arrays are used to store such items whose data type is same. Normally arrays are used to represent lists, trees and graphs; which are types of data structure. We may perform different operations on arrays such as: inserting an item, removing an item, finding a value in array, performing mathematical operations on elements of array etc.

Example:

An array named “Numbers” containing five elements of type integer (int) can be defined as:

```
int[] Numbers = new int[5];
```

Figure 1 represents the logical representation of this array, which shows elements of the array are located consecutively:

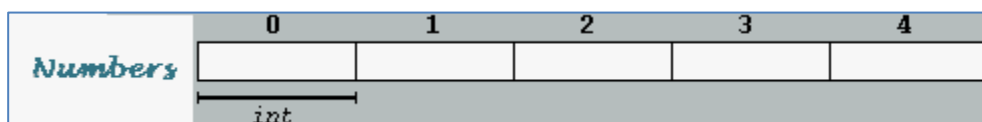


Figure 1 Logical representation of an array of integer elements

4.2. Initializing Arrays

When an array is declared in a local scope, if not specified otherwise, its elements are not initialized by some value by default. Elements of global and static arrays are automatically initialized by their default values (zeros for numeric data types).

If we want to initialize any global or local array by assigning some values to its element we can do so. Following example represents the initialization of an array of 5 integer elements.

```
int[] Numbers = {16, 2, 77, 40, 12071};
OR
Numbers [0] = 16;
Numbers [1] = 2;
Numbers [2] = 77;
Numbers [3] = 40;
Numbers [4] = 12071;
```

| | 0 | 1 | 2 | 3 | 4 |
|---------|----|---|----|----|-------|
| Numbers | 16 | 2 | 77 | 40 | 12071 |

Figure 2 An array initialized by values

4.3. Accessing values from an array

We can access value of an element of array in our program as it was a normal variable. We are allowed to read and modify the values of these elements. Following the previous examples in which Numbers array has 5 elements and each of those elements was of type int, the name which we can use to refer to each element is the following:

For example, to store the value 25 in the third element of Numbers, we could write the following statement:

```
Numbers [2] = 75;
```

To pass the value of the third element of Numbers to a variable called a, we could write:

```
int a = Numbers[2];
```

4.4. Static Arrays

Static arrays are allocated memory at compile time and their size is fixed, it means it cannot be changed latter during program execution. For example two int arrays are declared, one initialized and one without initialization.

```
int a[10];
int b[5] = {8, 20, 25, 9, 14};
```

4.5. Dynamic Memory Allocation in JAVA

In Java, when you create an object using the **new** keyword, memory is allocated on the heap, and the object is initialized. These operators can be used with primitive data types (int, float, char etc) as well as with user defined data types (UDT), such as classes. Following examples illustrate usage of new and delete operators for primitive as well as user defined data types.

For int data type, new operator can be used as:

```
// Dynamic memory allocation for an array of integers
int[] dynamicArray = new int[5];

// Dynamic memory allocation for an object
MyClass myObject = new MyClass();
```

In this example, **new int[5]** allocates memory for an array of five integers on the heap. Similarly, **new MyClass()** allocates memory for an instance of the **MyClass** class on the heap.

It's important to note that, in Java, you don't have to worry about explicitly freeing memory or deallocating objects. The garbage collector automatically identifies objects that are no longer reachable and releases the associated memory. This automatic memory management simplifies memory allocation and deallocation tasks for Java developers.

We may allocate memory space dynamically to an array of integers. Following example illustrates this concept.

```
Scanner scanner = new Scanner(System.in);

System.out.print("Enter size of array: ");
int size = scanner.nextInt();

int[] array = new int[size];

for (int i = 0; i < size; i++) {
    array[i] = i + 1;
    System.out.print(array[i] + " ");
}

// Closing the scanner (good practice to avoid resource leaks)
scanner.close();
```

new operator can be used with a user defined type or class/struct as well. Assume following is a class definition for which we need to create a dynamic object.

```
public class Main {
    public static void main(String[] args) {
```

```

Rectangle myRectangle = new Rectangle();
myRectangle.setValues(3, 4);
System.out.println("Area: " + myRectangle.area());
}
}

```

Lab Task:

- Write a program that enter array length and value from user, and display the elements of array.

```

1 import java.util.*;
2 class OnedimensionalScanner
3 {
4     public static void main(String args[])
5     {
6         int len;
7         Scanner sc=new Scanner(System.in);
8         System.out.println("Enter Array length : ");
9         len=sc.nextInt();
10        int a[]=new int[len]; //declaration
11        System.out.print("Enter " + len + " Element to Store in Array :\n");
12        for(int i=0; i<len; i++)
13        {
14            a[i] = sc.nextInt();
15        }
16        System.out.print("Elements in Array are :\n");
17        for(int i=0; i<len; i++)
18        {
19            System.out.print(a[i] + " ");
20        }
21    }
22 }

```

Output:

```

1 Enter Array length :
2 4
3 Enter 4 Element to Store in Array :
4 1
5 2
6 3
7 4
8 Elements in Array are :
9 1 2 3 4

```

- Write a program that create 2D array. Enter array length and value from user, and display the elements of array.


```

1 import java.util.*;
2 class TwoDimensionalScanner
3 {
4     public static void main(String args[])
5     {
6
7         Scanner sc=new Scanner(System.in);
8         System.out.println("Enter Row length of an array : ");
9         int row=sc.nextInt();
10        System.out.println("Enter column length of an array : ");
11        int column=sc.nextInt();
12        int a[][]=new int[row][column]; //declaration
13        System.out.print("Enter " + row*column + " Elements to Store in Array :\\n");
14        for (int i = 0; i < row; i++)
15        {
16            for(int j = 0; j < column; j++)
17            {
18                a[i][j] = sc.nextInt();
19            }
20        }
21        System.out.print("Elements in Array are :\\n");
22        for (int i = 0; i < row; i++)
23        {
24            for(int j = 0; j < column; j++)
25            {
26                System.out.println("Row ["+i+"]: Column ["+j+"] :"+a[i][j]);
27            }
28        }
29    }
30 }

```

Output:

```

1 Enter Row length of an array :
2 2
3 Enter column length of an array :
4 3
5 Enter 6 Elements to Store in Array :
6 1
7 2
8 3
9 4
10 5
11 6
12 Elements in Array are :
13 Row [0]: Column [0] :1
14 Row [0]: Column [1] :2
15 Row [0]: Column [2] :3
16 Row [1]: Column [0] :4
17 Row [1]: Column [1] :5
18 Row [1]: Column [2] :6

```

4.6. Multi-Dimensional Arrays

An array of arrays is called a multi dimensional array. Number of dimensions in a multi dimensional array is represented by number of subscripts in definition of that array. A multi dimensional array can represent data from different dimensions point of view. For example product sales information of departmental store from product, location/city and year of sale dimensions point of view is an example of multi dimensional array. It can be logical represented by following figure 1.

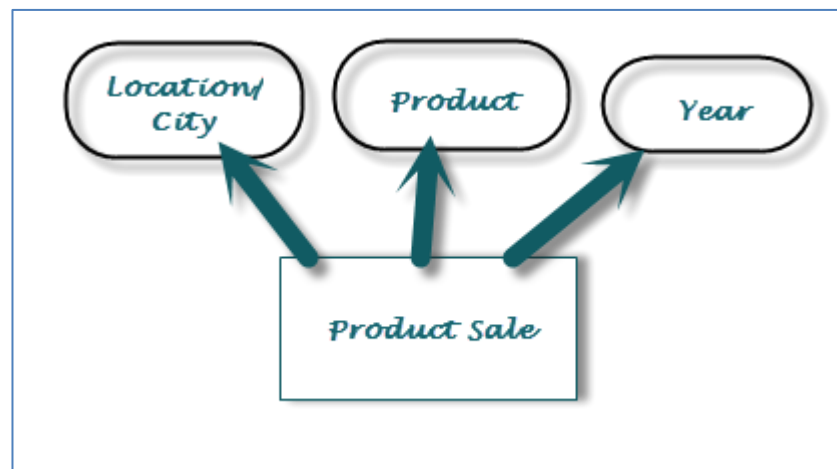


Figure 3 A logical representation of a multi dimensional (3-dimensional) array

4.6.1. Two Dimensional Arrays in JAVA

A two dimensional array is a specific case of multi-dimensional array in which two subscripts are involved in definition of array. For example “Numbers” represents an array of 3 per 5 elements. The way to declare this array in JAVA should be:

```
int numbers [3][5];
```

Following figure 2 shows a logical representation of a two dimensional array “numbers”.

| | | 0 | 1 | 2 | 3 | 4 |
|---------|---|---|---|---|---|---|
| numbers | 0 | | | | | |
| | 1 | | | | | |
| | 2 | | | | | |

Figure 4 A logical representation of a two dimensional array

Indices of first subscripts are logical represented as rows and indices of second subscripts are logically represented as columns. Now for example the way to reference the second element vertically and fourth horizontally in an expression would be:

```
numbers [1][3];
```

Following figure 3 shows the element in the two dimensional array which is referred above.

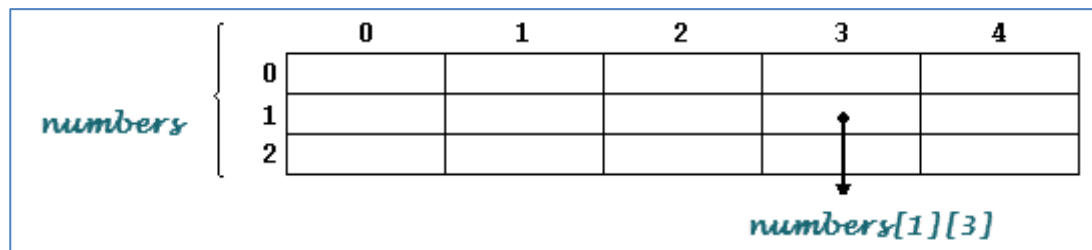


Figure 5 Referencing an element in two dimensional arrays

To assign some value to elements of a two-dimensional array, we simply use two subscripts:

```
numbers[1][3] = 7;
```

To initialize a two-dimensional array, it is easiest way to use nested braces, with each set of numbers representing a row:

```
int numbers[3][5] =
    {{ 1, 2, 3, 4, 5, }, // row 0
     { 6, 7, 8, 9, 10, }, // row 1
     { 11, 12, 13, 14, 15 } // row 2
    };
```

We may initialize a two-dimensional array of type int by values 0 by using following statement. Remember that we have to specify the size of array in terms of rows and columns for proper initialization.

```
int numbers[3][5] = { 0 };
```

4.6.2. Three Dimensional Arrays in JAVA

Multi-dimensional arrays may be larger than two dimensions, arrays which have 3 dimensions are called multi-dimensional arrays, they can be logical represented by a cube shape. An array representing product sales for departmental stores at five cities/locations, 10 products and containing sale information for 7 years can be declared as:

```
float productSales[5][10][7];
```

Where first subscript represents the location/city number, second subscript represents the product number and third subscript represents the year number for which product sale information will be stored in this array.

Three dimensional arrays cannot be easily initialized as compared to two dimensional arrays initialization, so normally they are initialized by zero/blank value and then further initialization is performed by using nested loops.

5. Homework before Lab

This homework will help students to study the concepts of topic before start of lab.

5.1. Problem Solution Modeling

After studying the introduction and concept map sections you should be ready to provide the solution of following problems. Design the solutions of the problems in JAVA and bring your code to lab so that lab instructor should assess and grade it.

5.2. Problem description:

Design a program which should contain two arrays, both arrays should contain elements of type integer (int). One of these arrays should be a static array and other should be a dynamic array. User will provide numerical values as input for these arrays. Your program should read value from each element and add these values to a single variable. After completion of addition operation for two arrays, value of this variable should be displayed on computer screen.

6. Procedures & Tools

This section provides information about tools and programming procedures used for the lab.

6.1. Tools

NetBeans 21 JAVA compiler configured.

6.2. Editing, Compiling and Running programs in NetBeans 21[Expected time = 5mins]

You are required to use NetBeans 21 environment for programming on data structures. It is expected that you are already familiar with this environment in your courses of Computer Programming and Object-Oriented Programming. You should be able to write code in source code files (java files) and perform compilation of code. Beside that you are also required to be proficient in performing line by line debugging and break point based debugging to remove the errors from your code.

6.3. Walk through Task

[Expected time = 20mins]

Following program represents the concept related to static and dynamic arrays; you are required to type this program in editor of NetBeans 21, compile and run to see the output of this program. Figure 3 shows the source code window of this program in NetBeans 21.

```
package com.mycompany.project1;

public class Project1 {

    public static void main(String[] args) {
        int[] Numbers = {16, 2, 77, 40, 12071};
        for (int i = 0; i < 5; i++) {
            System.out.println(Numbers[i]);
        }

        int[] arrayPtr = new int[10];
        for (int i = 0; i < 10; i++) {
            arrayPtr[i] = i;
            System.out.println(arrayPtr[i]);
        }
    }
}
```

Figure 6 A programs about static and dynamic arrays in NetBeans 21

Output of this program is shown in figure 4, when program is compiled and executed.

```

16
2
77
40
12071
0
1
2
3
4
5
6
7
8
9
-----
BUILD SUCCESS
-----

```

Figure 7 Output window displaying results of program after compilation

```

package com.mycompany.project1;

public class Project1 {

    public static void main(String[] args) {
        int[][] Number = {
            {1, 2, 3, 4, 5},
            {6, 7, 8, 9, 10},
            {11, 12, 13, 14, 15}
        };

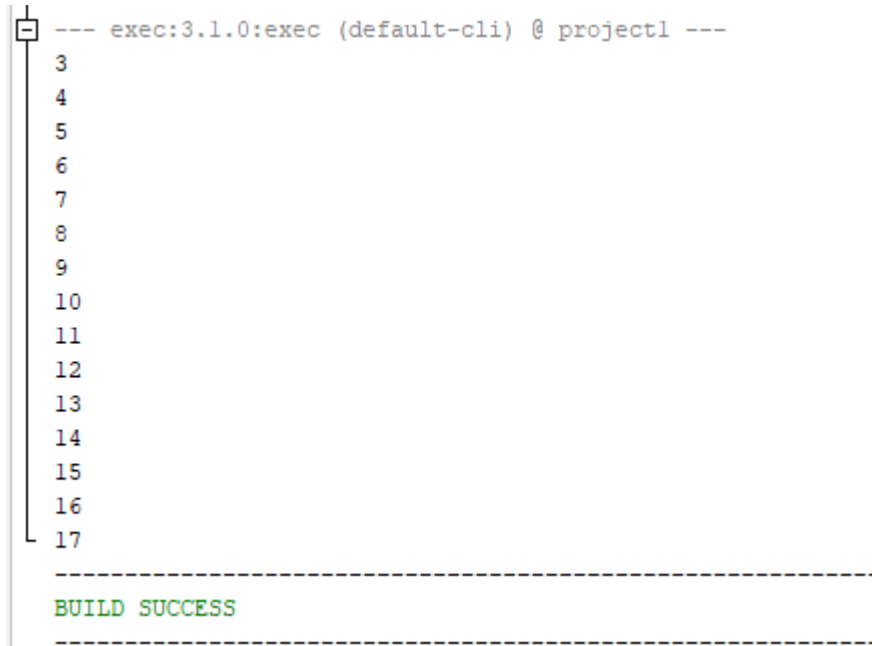
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 5; j++)
                Number[i][j] += 2;

        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print(Number[i][j] + " \n");
            }
        }
    }
}

```

Figure 8 Use of two dimensional arrays in Microsoft NetBeans 21

Figure 6 represents the output of this program when executed.

The image shows a screenshot of the NetBeans IDE's Output window. At the top, it displays the command prompt path: `--- exec:3.1.0:exec (default-cli) @ project1 ---`. Below this, a list of integers is shown, representing the elements of a two-dimensional array. The numbers are: 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, and 17. At the bottom of the window, the text `BUILD SUCCESS` is displayed in green, indicating that the program compiled without errors. The window is framed by a vertical scrollbar on the left and horizontal dashed lines above and below the output text.

```
--- exec:3.1.0:exec (default-cli) @ project1 ---
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
-----
BUILD SUCCESS
-----
```

Figure 9 Output window displaying values of elements of two dimensional arrays in NetBeans

7. Practice Tasks

This section will provide information about the practice tasks which are required to be performed in lab session. Design solutions of problems discussed in each task and place solution code in a folder specified by your lab instructor.

- 7.1. Create a dynamic array of user defined size. Now take input in array from user. Take a new (integer) value from user and search that how many times the entered number is present in the array.

```

--- exec:3.1.0:exec (default-cli) @ project1 ---
Enter the size of the array: 5
Enter elements for the array:
Element 1: 13
Element 2: 31
Element 3: 42
Element 4: 24
Element 5: 13
Enter a new integer value to search: 13
The value 13 occurs 2 times in the array.

-----
BUILD SUCCESS
-----

Total time: 18.098 s
Finished at: 2024-03-11T13:19:52+05:00
-----

```

- 7.2. Write a program to create a dynamic array of user defined size. Array should be of character type. Write a function ChangeCase() that should convert all the small alphabets to capital and vice versa.

```

-----
Enter the size of the character array: 6
Enter characters for the array:
Character 1: a
Character 2: s
Character 3: d
Character 4: F
Character 5: g
Character 6: T
Updated character array after changing case:
A S D f G t

-----
BUILD SUCCESS
-----

Total time: 27.323 s
Finished at: 2024-03-11T13:16:21+05:00
-----

```


- 7.3. Write a program to create a dynamic array of user defined size. Size should be in the range of 0 to 15. Write a function FindLarge() that should ask user to enter a non-negative number. Function should find the next largest number than the input number in the list.

Sample input:

Enter size: 5

After insertion:

| | | | | |
|----|----|----|----|---|
| 14 | 55 | 40 | 78 | 5 |
|----|----|----|----|---|

Sample output:

Enter number: 50

Next largest number from 50 is: 55

```

--- exec:3.1.0:exec (default-cli) @ project1 ---
Enter the size of the array (between 0 and 15): 5
Enter integers for the array:
Element 1: 14
Element 2: 55
Element 3: 40
Element 4: 78
Element 5: 5
Enter a non-negative number: 50
Next largest number after 50 is: 55

```

BUILD SUCCESS

Total time: 31.905 s
 Finished at: 2024-03-11T13:18:27+05:00

7.4. Out comes

After completing this lab, student will be able to understand and develop programs related to static and dynamic arrays in JAVA using Microsoft NetBeans 21 environment.

8. Evaluation Task (Unseen) [Expected time = 60mins for two tasks]

The lab instructor will give you unseen task depending upon the progress of the class.

9. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 2 Evaluation of the Lab

| Sr. No. | Task No | Description | Marks |
|---------|---------|----------------------------|-------|
| 1 | 4 | Problem Modeling | 20 |
| 2 | 6 | Procedures and Tools | 10 |
| 3 | 7 | Practice tasks and Testing | 35 |
| 4 | 8 | Evaluation Tasks (Unseen) | 20 |
| 5 | | Comments | 5 |
| 6 | | Good Programming Practices | 10 |

10. Further Readings

10.1. JAVA Tutorial Web Sites

1. <http://www.cplusplus.com/doc/tutorial/>
2. <http://www.learncpp.com/>
3. <http://www.tutorialspoint.com/cplusplus/>

10.2. Web sites containing supporting material

1. <http://www.engppt.com/2012/08/data-structures-with-c-ppt-slides.html>
2. <http://www.compgeom.com/~piyush/teach/3330/slides/>