



## **Java Lab – Arrays in Java**

***Reading from a file into an array***

***And***

***Two Dimensional (2D) arrays***

**IMPORTANT! Save all your work to a safe location such as oneDrive.**

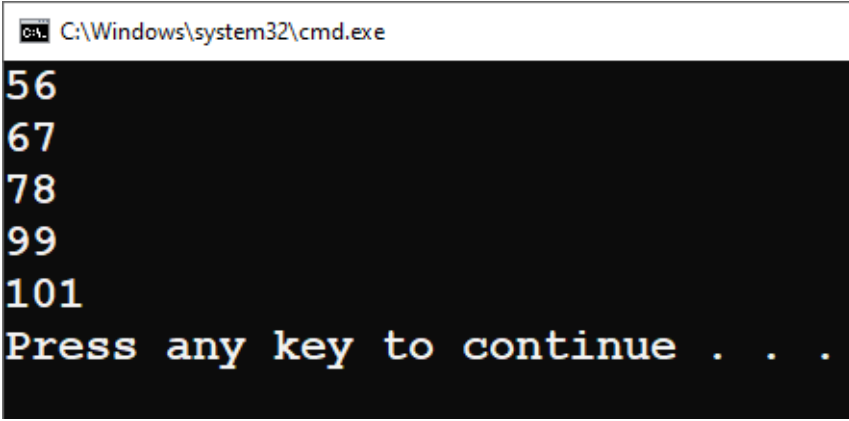
Create a folder for SDPD into which you will save all your work for this module, arranged how you wish. Ideally you should create a folder each week for your lab exercises. Note that you should create a separate file for each exercise.

## Exercise 1

In this exercise you will write a program that **reads from a file** and stores the information into an array. Create a program that reads information from a file, and stores each line (which contains an integer) into an array called *nums*. The file "numbers.txt" is provided on Moodle. There are 5 lines in the file, therefore the array should allow for 5 indexes. Sample code is provided here:

```
1 import java.util.Scanner; // Needed for the Scanner class
2 import java.io.*;         // Needed for the File class
3
4 public class ReadToArray
5 {
6     public static void main(String[] args) throws IOException
7     {
8         //Use the file called numbers.txt
9         String filename = "numbers.txt";
10        File file = new File(filename);
11        Scanner inputFile = new Scanner(file);
12
13        //Create an array of ints called nums
14        int[] nums = new int[5];
15
16        //Populate each index of the array with the data from the file
17        nums[0] = inputFile.nextInt();
18        nums[1] = inputFile.nextInt();
19        nums[2] = inputFile.nextInt();
20        nums[3] = inputFile.nextInt();
21        nums[4] = inputFile.nextInt();
22
23        inputFile.close();
24    }
25 }
```

Create a *for loop* that outputs the values to the console to confirm that your program worked, eg:



```
C:\Windows\system32\cmd.exe
56
67
78
99
101
Press any key to continue . . .
```

Amend your code so that the array is populated using a *for loop* (replacing the code on lines 17 to 21 shown above). Test your output again to confirm that it works as expected.

## Exercise 2

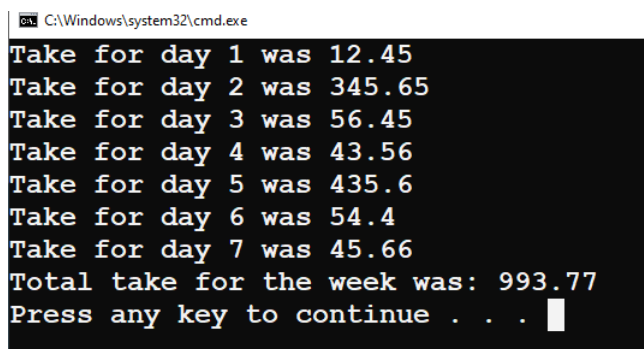
Create a program that will read the data from the file called *myfile.txt* (provided on Moodle) and store this data into an array called *namesList*. The file has 26 lines (26 names) - each line is a String. Include in your program a loop to output each name stored in the array, similar to as shown:



```
C:\Windows\system32\cmd.exe
Alice
Bob
Charlie
Donald
Ernie
Frank
Gary
Helen
Ian
Jane
Kevin
Lorraine
Michelle
Noel
Ophelia
Patrick
Quentin
Robert
Siobhan
Terry
Una
Vera
Wilfred
Xavier
Yolanda
Zach
Press any key to continue . . .
```

## Exercise 3

Using the file “weekly.txt” provide on Moodle, create a program that reads that data from this file into an array. The text file contains the cash taken for a coffee shop for each day of the week. Your program should output the take for each day, and show the total for the week by adding each of the array elements.



```
C:\Windows\system32\cmd.exe
Take for day 1 was 12.45
Take for day 2 was 345.65
Take for day 3 was 56.45
Take for day 4 was 43.56
Take for day 5 was 435.6
Take for day 6 was 54.4
Take for day 7 was 45.66
Total take for the week was: 993.77
Press any key to continue . . .
```

## Exercise 4

Using the file called rainfall.txt on Moodle, populate an array with the data from this file. The file contains rainfall data for every day, for a year. Your program should populate the array (type double); every days rainfall should be output to the console, as shown below. In addition, output the total rainfall for the year, average rainfall per day, week, and month. Your program should also locate the highest and lowest values in the array and output the appropriate message, as shown below:

```
C:\Windows\system32\cmd.exe
Rainfall for day 344 was 1.59
Rainfall for day 345 was 1.6
Rainfall for day 346 was 0.92
Rainfall for day 347 was 1.01
Rainfall for day 348 was 5.96
Rainfall for day 349 was 0.57
Rainfall for day 350 was 2.23
Rainfall for day 351 was 0.01
Rainfall for day 352 was 1.69
Rainfall for day 353 was 4.54
Rainfall for day 354 was 3.04
Rainfall for day 355 was 0.01
Rainfall for day 356 was 2.93
Rainfall for day 357 was 0.87
Rainfall for day 358 was 3.87
Rainfall for day 359 was 6.48
Rainfall for day 360 was 5.96
Rainfall for day 361 was 0.01
Rainfall for day 362 was 0.22
Rainfall for day 363 was 0.22
Rainfall for day 364 was 0.57
Rainfall for day 365 was 4.95
Total rainfall for the year was: 1090.99
Average daily rainfall was: 2.99
Average weekly rainfall was: 20.98
Average monthly rainfall was: 90.92
The day with highest rainfall was day 209, which had 14.68mm of rainfall that day.
The day with lowest rainfall was day 126, which had 0.0mm of rainfall that day.
```

## Exercise 5

In this exercise, you will write a program to create a simple 2d array of integers and then output the results to the console.

```
public class Arrays2D_Exercise1
{
    public static void main(String[] args)
    {
        int[][] myGrid = new int[1][3];

        myGrid[0][0] = 99;
        myGrid[0][1] = 98;
        myGrid[0][2] = 97;

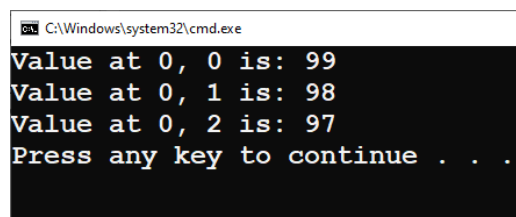
        System.out.println("Value at 0, 0 is: " + myGrid[0][0]);
        System.out.println("Value at 0, 1 is: " + myGrid[0][1]);
        System.out.println("Value at 0, 2 is: " + myGrid[0][2]);
    }
}
```

In above example, the 2D array `myGrid` can be conceptually visualized like this:

99	98	97
----	----	----

This is in effect a matrix (or table) that has one row ( `int[1][3];` ) and three columns ( `int[1][3];` )

Your output from the should be similar to as shown:



```
C:\Windows\system32\cmd.exe
Value at 0, 0 is: 99
Value at 0, 1 is: 98
Value at 0, 2 is: 97
Press any key to continue . . .
```

## Exercise 6

In this exercise, you will write a program to create a simple 2d array of integers and then output the results to the console.

```
int[][] myArray = new int[2][4];

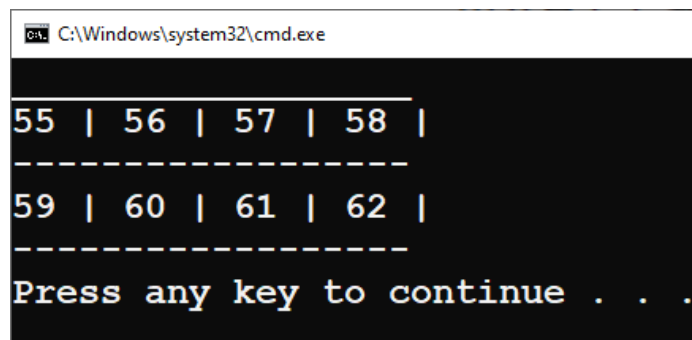
myArray[0][0] = 55;
myArray[0][1] = 56;
myArray[0][2] = 57;
myArray[0][3] = 58;
myArray[1][0] = 59;
myArray[1][1] = 60;
myArray[1][2] = 61;
myArray[1][3] = 62;
```

In the above example, the 2D array `myArray` can be conceptually visualized like this:

55	56	57	58
59	60	61	62

This is in effect a matrix (or table) that has two rows and four columns `int[2][4]`:

Add code so that the output is similar to as shown:



```
C:\Windows\system32\cmd.exe

55 | 56 | 57 | 58 |
-----
59 | 60 | 61 | 62 |
-----
Press any key to continue . . .
```

## Exercise 7

Write a program to create a 2d array of integers called *myData* and then output the results to the console.

The 2D array *myData* should be structured as follows:

10	11	12	13	14	15
16	17	18	19	20	21
22	23	24	25	26	27

Add code so that the output is similar to as shown:

```
C:\Windows\system32\cmd.exe
10 | 11 | 12 | 13 | 14 | 15 |
16 | 17 | 18 | 19 | 20 | 21 |
22 | 23 | 24 | 25 | 26 | 27 |
Press any key to continue . . .
```

Amend the array so that it is updated with the following values (marked in red):

10	11	99	13	14	15
16	17	18	50	20	21
77	23	24	25	26	27

Output the array again, confirming that the values have been changed accordingly:

```
C:\Windows\system32\cmd.exe
10 | 11 | 12 | 13 | 14 | 15 |
16 | 17 | 18 | 19 | 20 | 21 |
22 | 23 | 24 | 25 | 26 | 27 |

ARRAY AFTER UPDATING:
10 | 11 | 99 | 13 | 14 | 15 |
16 | 17 | 18 | 50 | 20 | 21 |
77 | 23 | 24 | 25 | 26 | 27 |
```

## Exercise 8

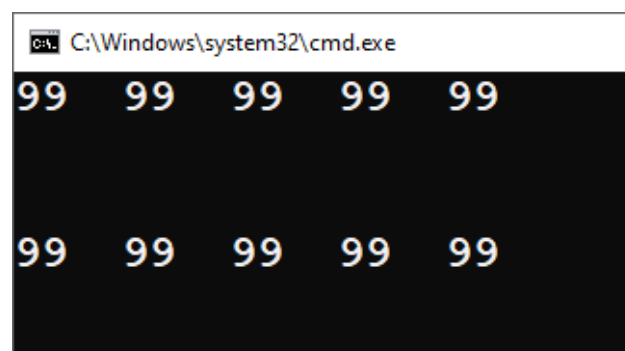
Write a program to create a 2d array of integers called *myCounter* and then output the results to the console. Use a for loop to populate the array. The finished array should have the following structure:

99	99	99	99	99
99	99	99	99	99

Note that this array has 2 rows and 5 columns. To populate this using a for loop, the following approach can be used:

```
//Outer for loop that will count rows
for(int row = 0; row <=1; row++)
{
    //Inner for loop that will count columns
    for(int col = 0; col <=4; col++)
    {
        //Value is assigned to array index
        myCounter[row][col] = 99;
    }
}
```

Using an additional for loop (with a similar structure), output the array to the console – your result should look similar to as shown:





## Exercise 9

Using the for loop from the previous exercise, write a program to create a 2d array of integers called *thursday* and then output the results to the console. Use a for loop to populate and output the array. The finished array should have the following structure:

50	50	50	50	50	50
50	50	50	50	50	50
50	50	50	50	50	50
50	50	50	50	50	50
50	50	50	50	50	50

## Exercise 10

Using the for loop from the previous exercise, write a program to create a 2d array of integers called *numbers* and then output the results to the console. Use a for loop to populate and output the array. The finished array should have the following structure:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

## Exercise 11

Write a program to create a 2d array of string called *employees* and then output the results to the console. Prompt the user to enter the values (using scanner) to populate and output the array (no need to use a for loop for this exercise). The finished array should have the following structure:

Joe	Biden
Hillary	Clinton
Donald	Trump

Output the results to the console when complete.

```
C:\Windows\system32\cmd.exe
Enter first name for person 1:Joe
Enter surname for person 1:Biden
Enter first name for person 2:Hillary
Enter surname for person 2:Clinton
Enter first name for person 3:Donald
Enter surname for person 3:Trump
Person 1: Joe Biden
Person 2: Hillary Clinton
Person 3: Donald Trump
Press any key to continue . . .
```

Amend your code so that the array contains an extra column for phone numbers, and will produce the following output:

```
C:\Windows\system32\cmd.exe
Enter first name for person 1:Joe
Enter surname for person 1:Biden
Enter phone number of person 1:091-123456
Enter first name for person 2:Hillary
Enter surname for person 2:Clinton
Enter phone number of person 2:087-123456
Enter first name for person 3:Donald
Enter surname for person 3:Trump
Enter phone number of person 3:089-456789
Person 1: Joe Biden - Phone number is: 091-123456
Person 2: Hillary Clinton - Phone number is: 087-123456
Person 3: Donald Trump - Phone number is: 089-456789
Press any key to continue . . .
```

## Exercise 12

Write a program to create a 2d array of strings called *puzzle* and then output the results to the console. The array should have the following structure:

5	9	18
6	22	4
6	4	7

Your program should add all the values in one of the columns - Allow the user to specify which column:

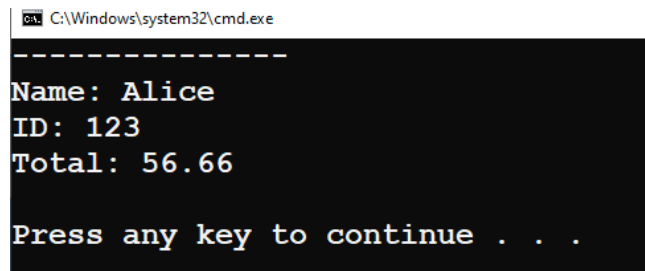
```
C:\Windows\system32\cmd.exe
5  9  18
6  22  4
6  4  7
Add all the values of what column? (Enter 0, 1 or 2):1
The addition of all the values in column 1 is:35
Press any key to continue . . .
```

## Exercise 13

Using the file *people.txt* provided on Moodle, read the file into three separate 1D arrays. The first line should be stored into a String array, the second line should be stored into an int array, and the third line should be stored into a double array. You should be able to produce this using a for loop, eg:

```
int i = 0;
while(inputFile.hasNext())
{
    names[i] = inputFile.next();
    idNum[i] = inputFile.nextInt();
    total[i] = inputFile.nextDouble();
    i++;
}
```

Output the content of each array using a for loop, eg:



```
C:\Windows\system32\cmd.exe
-----
Name: Alice
ID: 123
Total: 56.66

Press any key to continue . . .
```

## Exercise 14

Using the file *banking.txt* provided on Moodle, which stores bank details for multiple people. Your program should read the file into three separate arrays.

The first line should be stored into a string array called `names`

The second line should be stored into an integer array called `accountNumber`

The third line should be stored into a double array called `current`

Each subsequent set of lines in the file should be read into the appropriate array.

Output all array data to the console, similar to as shown below:

```
C:\Windows\system32\cmd.exe
-----
Name: Alice
ID: 123
Total: 56.66
-----
Name: Bob
ID: 456
Total: 565.67
-----
Name: Charlie
ID: 567
Total: 456.66
-----
Name: Donna
ID: 876
Total: 676.43
-----
Name: Eve
ID: 97
Total: 45.66
```

Amend your program so that the user is prompted to enter an index, and the appropriate information is displayed to the console, eg:

```
C:\Windows\system32\cmd.exe
Select user?: 3
-----
Name: Donna
ID: 876
Total: 676.43
```

## Exercise 15

Using the file *numbers.txt* provided on Moodle, create a program that read the file data into an array (40 lines of text).

Your program should perform the following operations:

- Determine the difference between the lowest and highest values in the array

```
C:\Windows\system32\cmd.exe
Highest value is: 9856 and is at index 30
Lowest value is: 1265 and is at index 3
The difference between the highest and the lowest values is : 8591
```

- Sort the array from the lowest value to the highest value

*You can use the Arrays class:*

```
import java.util.Arrays;
```

*And use as follows (example below is sorting an array called number):*

```
Arrays.sort(number);
```

- Prompt the user on whether they would like to output the list to the console in order or in reverse order:

```
Output array in order (enter o)
or in reverse order (enter r)?
```

- Output every 5<sup>th</sup> value in the array:

```
Outputting every 5th value
1265
2484
2875
4567
4589
5456
5541
7665
```

- Prompt the user on what to output from the array, eg: where to start output from and how many elements to output:

```
-- Array Output --
Specify output starting at index: 20
Specify how many elements should be output: 6
Index: 20; value: 4589
Index: 21; value: 4662
Index: 22; value: 4678
Index: 23; value: 4978
Index: 24; value: 5434
Index: 25; value: 5456
Press any key to continue . . .
```

Your program should perform validation on this feature. For example, if the user specifies an index value to begin that is great than the size of the array, they should be prompted again to enter a number that will work for the array. Similarly, if the user enters a value for outputting array elements

that would go beyond the length of the array, they should be prompted to enter a number that will work. For example, this array holds 40 values:

```
-- Array Output --
Specify output starting at index: 45
Specify output starting at index: 35
Specify how many elements should be output: 60
Specify how many elements should be output: 3
Index: 35; value: 7665
Index: 36; value: 7894
Index: 37; value: 8753
Press any key to continue . . .
```

*45 is too high a number to specify as the array has only 40 elements  
60 elements to output is too large at this would be larger than the array size.*

## Exercise 16

Using the files *capitals.txt* and *countries.txt* provided on Moodle, create a program that reads the file data from both files into two arrays (196 lines of text each).

The files contain a list of countries and capital cities. These files are related – for example, Ireland is on line 81 of the *countries.txt* file, and Dublin is on line 81 of the *capitals.txt* file.

Your program should display a simple menu and allow the user to search by country and the program will display the capital city of the country, or vice versa. See screenshot below for sample output. The program should continue to run until the user enters 0 in the menu.

```
C:\Windows\system32\cmd.exe
-----Country and Capital City Search-----
Enter 1 to search by country or 2 to search by city. Enter 0 to end program: 1
Enter country name: Ireland
The capital of Ireland is Dublin
Enter 1 to search by country or 2 to search by city. Enter 0 to end program: 2
Enter city name: Rome
Rome is the capital of Italy
Enter 1 to search by country or 2 to search by city. Enter 0 to end program: 2
Enter city name: London
London is the capital of United Kingdom
Enter 1 to search by country or 2 to search by city. Enter 0 to end program: 1
Enter country name: Australia
The capital of Australia is Canberra
Enter 1 to search by country or 2 to search by city. Enter 0 to end program: 0
Press any key to continue . . .
```