Arrays and Command Line Args

Array In Java:

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with **square brackets.**

```
int[] marks;
```

Example:

If we want to store the names of 100 people then we can create an array of the string type that can store 100 names.

`String[] names = new String[100];`

Note: In Java array is a special type of object whose class is non-existence. whenever JVM encounters [], it will create an array object in the memory.

example:

```
int[] marks = new int[10];
System.out.println(marks);// it will print the address of array object
```

- The variables in the array are ordered, and each has an index beginning from 0.
- The **size** of an array must be specified by int or short value and not long.

In the above example marks is an array of int variables where the first variable is marks[0].

Note: at the time of creating an array object, all the variables inside the array will be initialized with their default value.

```
int[] marks = new int[10];

System.out.println(marks[0]); //0
System.out.println(marks[1]);//0
System.out.println(marks[15]);//Exception: ArrayIndexOutOfBoundException
```

If we try to access an element from an array beyond its size, then it will raise a runtime exception called **ArrayOutOfBoundException.**

Note: Inside every array object, there is a non-static variable called "**length**" is there, which will represent the size of an array.

```
System.out.println(marks.length);//10
```

In Java there is two way to create an array :

1. using new operator
2. using curly bracket

1. **Using new operator:**

   Using this approach we declare an array first and then we initialize each element of that array

   example:

   ```
   //declare an array
   int[] marks = new int[5];


   //initializing array
   marks[0]=100;
   marks[1]=120;
   marks[2]=150;
   --
   ```

2. Using curly bracket:

   Using this approach, we declare and initialize an array in a single statement.

   example

   ```
   int[] marks = {100,120,150,180};
   ```

   Accessing Array Elements:

I Problem

Creating an integer array with some value and printing each elements from that array

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

   // create an array
   int[] marks = {40, 40, 55, 25, 55};

   // access each array elements
   System.out.println("Accessing Elements of Array:");
   System.out.println("First Element: " + marks[0]);
   System.out.println("Second Element: " + marks[1]);
   System.out.println("Third Element: " + marks[2]);
   System.out.println("Fourth Element: " + marks[3]);
   System.out.println("Fifth Element: " + marks[4]);
 }
}
```

We can also loop through each element of the array. For example,

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

   // create an array
   int[] marks = {58, 45, 52};

   // loop through the array
   // using for loop
   System.out.println("Using for Loop:");
   for(int i = 0; i < marks.length; i++) {
     System.out.println(marks[i]);
   }
 }
}
```

We can also use the for-each loop to iterate through the elements of an array. For example,

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

   // create an array
   int[] marks = {45, 42, 55};

   // loop through the array
   // using for loop
   System.out.println("Using for-each Loop:");
   for(int m : marks) {
     System.out.println(m);
   }
 }
}
```

We Problem

Compute Sum and Average of Array Elements

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

        int[] numbers = {2, -9, 0, 5, 12, -25, 22, 9, 8, 12};
        int sum = 0;
        double average;

        // access all elements using for each loop
        // add each element in sum
        for (int number: numbers) {
            sum += number;
        }

        // get the total number of elements
        int arrayLength = numbers.length;

        // calculate the average
        // convert the average from int to double
        average =  (double)sum / arrayLength;

        System.out.println("Sum = " + sum);
        System.out.println("Average = " + average);
    }
}
```

You Problem:

Write a non-static method inside class Main, which will take an initialized integer array and return the largest number from the supplied array. call this method from the main method of main class by supplying an initialized integer array and print the returned result.

> Note: As we can create an array of primitive types like (byte, short, int, float, etc.), we can also create an array of reference types also like an array of Student, Employee, Product also.

example

```
class Student
{
    public int roll_no;
    public String name;
    Student(int roll_no, String name)
    {
        this.roll_no = roll_no;
        this.name = name;
    }

    public void printDetails(){

      System.out.println("Roll is :"+roll_no+" Name is :"+name);

    }

}
```

```
class Main{

public static void main(String[] args){

Student[] students = new Student[3];

students[0] = new Student(10,"Ram");
students[1] = new Student(20,"Ramesh");
students[2] = new Student(40,"Amit");


for(Student student:students){
  student.printDetails();
}

}
}
```

Note: in the above example total 4 objects are created in the memory one for the student array object and three for the student object.

The above student array we can create by using curly bracket also.

ex:

```
Student[] students = {new Student(10,"Ram"),new Student(20,"Ramesh"),new Student(40,"Amit")};
```

Since in Java, arrays are treated as an object, the array variable will be treated as a reference variable, and the default value of any reference variable is null.

example:

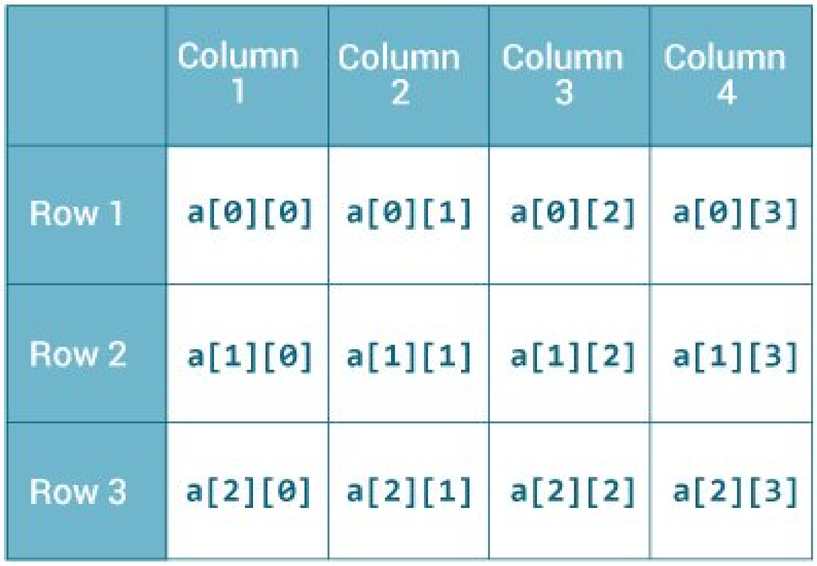
```
int[] numbers = null;
```

Multi-Dimensional array in java

In Java, we can declare a multi-dimensional array also.

A multidimensional array is an array of arrays. That is, each element of a multidimensional array is an array itself. For example,

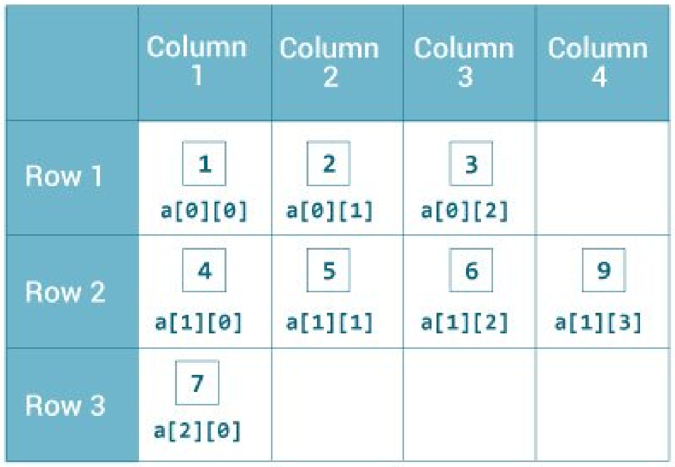
```
int [][] matrix = new int[3][4];//3 array type variable where each variable itself is an array of 4 variable
```



	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

we can initialize the 2-D array as follows :

```
int[][] a = {
      {1, 2, 3},
      {4, 5, 6, 9},
      {7},
};
```



	Column 1	Column 2	Column 3	Column 4
Row 1	1 a[0][0]	2 a[0][1]	3 a[0][2]	
Row 2	4 a[1][0]	5 a[1][1]	6 a[1][2]	9 a[1][3]
Row 3	7 a[2][0]			

Print all elements of 2d array Using Loop

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

        int[][] a = {
            {1, -2, 3},
            {-4, -5, 6, 9},
            {7},
        };

        for (int i = 0; i < a.length; ++i) {
            for(int j = 0; j < a[i].length; ++j) {
                System.out.println(a[i][j]);
            }
        }
    }
}
```

You Problem

Print the above 2d array using for..each loop.

Command Line Argument:

The command-line arguments in Java allow the programmers to pass the arguments during the execution of a program. The users can pass the arguments during the execution by passing the command-line arguments inside the main() method.

We can use these command-line arguments as input in our Java program.

We need to pass the arguments as space-separated values. We can pass both strings and primitive data types(int, double, float, char, etc.) as command-line arguments. These arguments convert into a string array and are provided to the main() function as a string array argument.

When command-line arguments are supplied to JVM, JVM wraps these and supplies them to args[]. It can be confirmed that they are actually wrapped up in an args array by checking the length of args using args.length.

We Problem :

Get 2 numbers from the CLA and print the sum of both numbers.

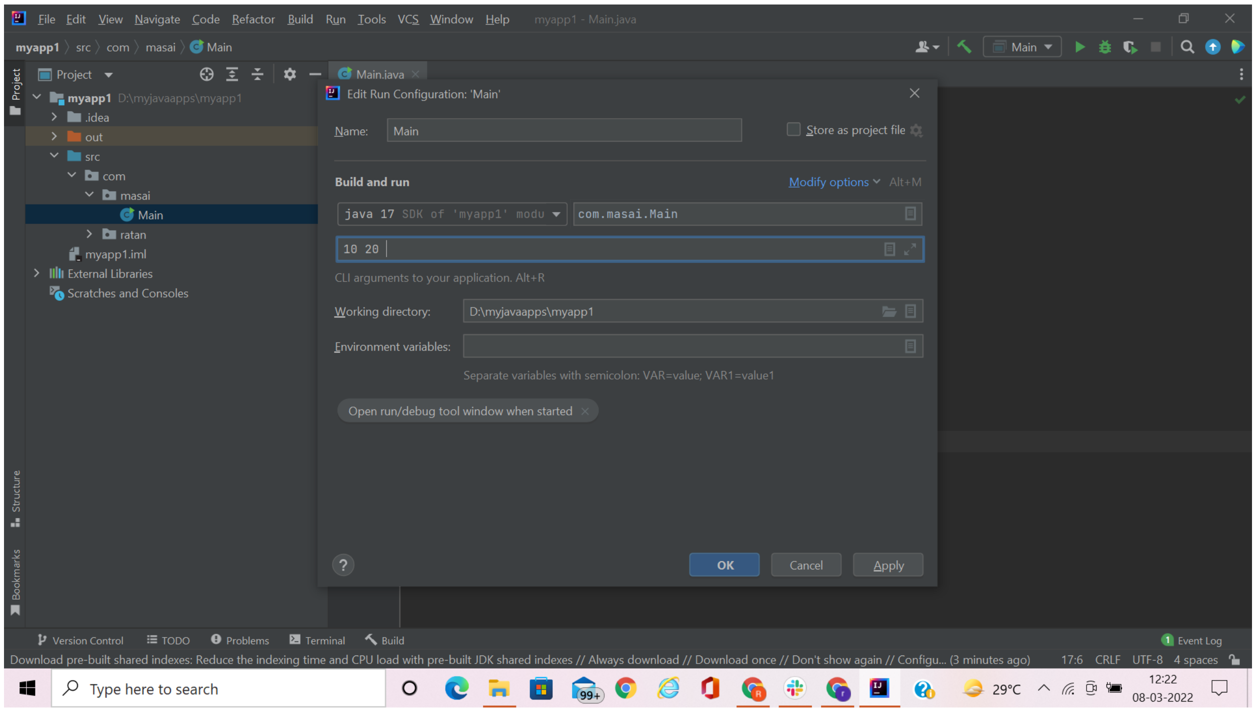
if supplied number count is less than or more than 2 number then it should print the proper message like (”Please supply only 2 numbers”)

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

        if(args.length == 2){

            int num1 = Integer.parseInt(args[0]);
            int num2 = Integer.parseInt(args[1]);

            System.out.println("The Result is "+ (num1+num2));

        }else
            System.out.println("Please supply only 2 numbers");


    }
}
```

Note: Here we have used the parseInt method of the Integer class, which is a static method, used to convert the string representation of a number to the primitive number.

To supply the command line argument with IntelliJ IDE :

Step 1: right click on the Main class → select “modify run Configuration...”


Edit Run Configuration: 'Main'

Name: Main

Store as project file

Build and run

```
java 17 SDK of 'myapp1' modu
com.masai.Main
10 20
```

CLI arguments to your application. Alt+R

Working directory: D:\myjavaapps\myapp1

Environment variables:

Separate variables with semicolon: VAR=value; VAR1=value1

Open run/debug tool window when started

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
myapp1 - Main.java
myapp1 src com masai Main
Project
myapp1 D:\myjavaapps\myapp1
.idea
out
src
com
masai
Main
ratan
myapp1.iml
External Libraries
Scratches and Consoles
Main.java
Modify options Alt+M
OK
Cancel
Apply
Version Control
TODO
Problems
Terminal
Build
Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configu... (3 minutes ago)
17:6 CRLF UTF-8 4 spaces
Type here to search
29°C
12:22
08-03-2022

Step 2: Run this Main class once again.