

Unit-3: Methods and Arrays

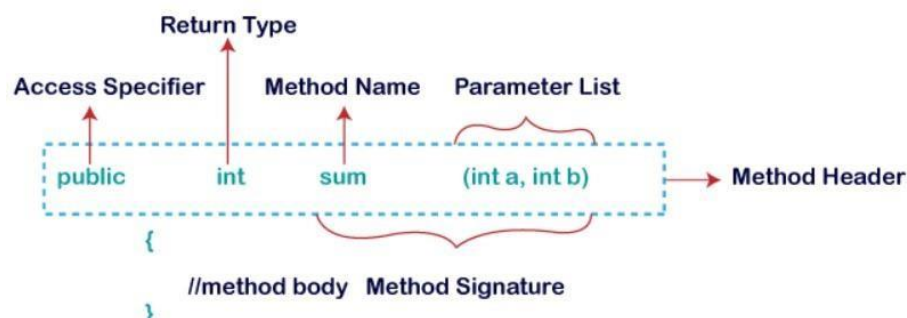
Defining and calling method, passing argument by values, Overloading methods and scope of variables, Method abstraction and stepwise refinement, Single Dimensional arrays, copying arrays, Passing and returning array from method, Searching and sorting arrays and the Array class, Two-Dimensional array and its processing, Passing Two-dimensional Array to methods, Multidimensional Arrays.

1. What is a Method in Java? Explain Method Declaration and How to Call or Invoke a User-defined Method.

- A method is a collection of statements or a set of code grouped together to perform a specific task or operation.
- It is used to make code reusable.
- We write a method once and use it many times.
- We don't need to write code over and over again.
- It also allows for easy code modification and readability by simply adding or removing a chunk of code.
- The method is only called or invoked when we call or invoke it.

Method Declaration

- The method declaration specifies method attributes like visibility, return type, name, and arguments.
- It is made up of six components known as method headers, as illustrated in the figure below.



Method Signature

- Every method has a method signature.
- It is part of the method declaration.
- It contains the method name as well as a list of parameters.

Access Specifier:

- The method's access type is specified by an access specifier or modifier.
- It defines the method's visibility.
- Java provides four types of access specifiers:
 - **Public:** When we use the public specifier in our application, all classes have access to the method.
 - **Private:** When we use a private access specifier, the method is only accessible within the classes in which it is defined.
 - **Protected:** When we use the protected access specifier, the method is only accessible within the same package or subclasses in another package.
 - **Default:** When no access specifier is specified in the method declaration, Java uses the default access specifier. It is only visible from the same package.

Unit-3: Methods and Arrays

Return Type:

- A data type that the method returns is referred to as a return type.
- It could be a primitive data type, object, collection, void, or something else.
- If the method returns nothing, we use the void keyword.

Method Name:

- It is a one-of-a-kind name used to define the name of a method.
- It must correspond to the method's functionality.
- Assume we are developing a method for the subtraction of two numbers; the method name must be subtraction ().
- The name of a method is used to call it.

Parameter list:

- The parameter list is a list of parameters separated by a comma and enclosed in parentheses.
- It contains the data type as well as the variable name.
- If the method does not have any parameters, leave the parentheses empty.

Method body:

- The method body is a component of the method declaration.
- It contains all of the actions that must be taken.
- It is surrounded by a pair of curly braces.

Naming a Method

- When defining a method, keep in mind that the method name must be a verb and begin with a lowercase letter.
- Except for the first word in the multi-word method name, the first letter of each word must be in uppercase.
- For instance:
- Single-word method name: sum() and area()
- Multi-word method names: areaOfCircle() and stringComparison ()

Types of Method

- There are two types of methods in Java:

Predefined Method:

- Predefined methods in Java are methods that have already been defined in Java class libraries.
- It is also referred to as the built-in method or the standard library method.
- We can use these methods directly in the program at any time by calling them.
- length(), equals(), compareTo(), sqrt(), and other pre-defined methods are available.
- When we call any of the predefined methods in our program, a series of codes related to the corresponding method is executed in the background and is already stored in the library.
- Each predefined method is defined within a class.
- The print() method is defined in the java.io.PrintStream class.

Unit-3: Methods and Arrays

- It prints the statement that we write inside the method. For example, `print("Java")` prints Java to the console.
- Example of the predefined method:

Demo.java

```
public class Demo{  
    public static void main(String[] args){  
        // using the max() method of Math class  
        System.out.print("The maximum number is:"+Math.max(10,5));  
    }  
}
```

Output:

The maximum number is: 10

- We used three predefined methods in the preceding example: `main()`, `print()`, and `max()`.
- Because these methods are predefined, we used them without declaring them.
- The `print()` method is a `PrintStream` class method that prints the result to the console.
- The `max()` method is a `Math` class method that returns the larger of two numbers.

User-defined Method:

- A user-defined method is one that has been written by the user or programmer.
- These methods are tailored to the situation.
- Let's write a user-defined method that checks whether a number is even or odd.
- First, we will define the method.

```
//user defined method  
public static void findEvenOdd(int num){  
    //method body  
    if(num%2==0)  
        System.out.println(num+" is even");  
    else  
        System.out.println(num+" is odd");  
}
```

- We named the above method `findEvenOdd()`.
- It has an `int` parameter called `num`.
- Because the method does not return any value, we have used `void`.
- The method body contains the steps for determining whether a number is even or odd.
- If the number is even, it prints the number is even; otherwise, it prints the number is odd.

How to Call or Invoke a User-defined Method

- Once we've defined a method, it should be called.
- The process of calling a method in a program is straightforward.
- When we call or invoke a user-defined method, program control is transferred to the called method.

Unit-3: Methods and Arrays

Example: EvenOdd.java

```
import java.util.Scanner;
public class EvenOdd{
    public static void main (String args[]){
        //creating Scanner class object
        Scanner scan=new Scanner(System.in);
        System.out.print("Enter the number: ");
        //reading value from user
        int num=scan.nextInt();
        //method calling / invoking a method
        findEvenOdd(num);
    }
    //user defined method
    public static void findEvenOdd(int num){
        //method body
        if(num%2==0)
            System.out.println(num+" is even");
        else
            System.out.println(num+" is odd");
    }
}
```

Output:

```
Enter the number: 10
10 is even
```

Output:

```
Enter the number: 11
11 is odd
```

- Let's look at another program that returns a value to the calling method.
- In the following program, we defined a method called add() that sums the two numbers.
- It has two integer parameters, n1 and n2.
- The values of n1 and n2 correspond to the values of a and b, respectively.
- As a result, the method adds the values of a and b, stores them in the variable s, and returns the sum.

Addition.java

```
public class Addition{

    public static void main(String[] args){
        int a = 20;
        int b = 10;
        int c = add(a, b);
        System.out.println("The sum of a and b is = " + c);
    }
    public static int add(int n1, int n2){
        int s;
        s=n1+n2;
        return s; //returning the sum
    }
}
```

Unit-3: Methods and Arrays

Output:

The sum of a and b is = 30

2. Method main is a public static method. Justify.

- The main() method in Java programmes is the point at which the programme begins its execution, or simply the entry point of Java programmes.
- As a result, it is one of the most important Java methods, and a thorough understanding of it is essential.
- When a Java programme is executed, the Java compiler or JVM searches for the main method.
- For the JVM to recognise the main method as its entry point, its signature must be in a specific format.
- If we change the method signature, the programme compiles but does not run.
- The java.exe programme is used to run a Java programme.
- In turn, the Java.exe makes Java Native Interface (JNI) calls, which load the JVM.
- The command line is parsed by java.exe, a new String array is created, and the main() method is called.
- The main method is connected to a daemon thread, which is destroyed only when the Java programme terminates execution.
- **The most common syntax for the main() method is:**

```
// Java Program to demonstrate the
// syntax of the main() function

class Demo {
    public static void main(String[] args)
    {
        System.out.println("Hello");
    }
}
```

Output:

Hello

Explanation: Every word in the public static void main statement has got a meaning to the JVM.

1. Public

- It is an Access modifier, which specifies from where and who can access the method.
- Making the main() method public makes it globally available.
- It is made public so that JVM can invoke it from outside the class as it is not present in the current class.

```
// Java Program to demonstrate the
// use of any other access modifier
// other than public

class Demo {
    private static void main(String[] args)
    {
        System.out.println("Hello");
    }
}
```

Unit-3: Methods and Arrays

Error: Main method not found in class, please define the main method as: `public static void main(String[] args)` or a JavaFX application class must extend `javafx.application.Application`

2. Static

- It is a keyword that is when associated with a method, making it a class-related method.
- The `main()` method is static so that JVM can invoke it without instantiating the class.
- This also saves the unnecessary wastage of memory which would have been used by the object declared only for calling the `main()` method by the JVM.

```
// Java Program to demonstrate the error occurred when we dont use
//the static keyword in the main() method
```

```
class Demo {
    public void main(String[] args)
    {
        System.out.println("Hello");
    }
}
```

Error: Main method is not static in class test, please define the main method as: `public static void main(String[] args)`

3. Void

- It is a keyword and is used to specify that a method doesn't return anything.
- As the `main()` method doesn't return anything, its return type is void.
- As soon as the `main()` method terminates, the java program terminates too.
- Hence, it doesn't make any sense to return from the `main()` method as JVM can't do anything with the return value of it.

```
// Java Program to demonstrate the error occurred when we dont use the
// void return type in the main() method
```

```
class Demo {
    public static int main(String[] args){
        System.out.println("Hello");
        return 1;
    }
}
```

Error: Main method not found in class, please define the main method as: `public static void main(String[] args)` or a JavaFX application class must extend `javafx.application.Application`

4. main

- It is the name of the Java main method.
- It is the identifier that the JVM looks for as the starting point of the java program. It's not a keyword.

```
// Java Program to demonstrate the error occurred when we name the
// main() method as main.
```

Unit-3: Methods and Arrays

```
class Demo {  
    public static void myMain(String[] args){  
        System.out.println("Hello");  
    }  
}
```

Error: Main method not found in class, please define the main method as:
`public static void main(String[] args)` or a JavaFX application class
must extend `javafx.application.Application`

5. String[] args

- It stores Java command-line arguments and is an array of type `java.lang.String` class.
- Here, the name of the String array is `args` but it is not fixed and the user can use any name in place of it.

```
// Java Program to demonstrate the working of String[] args  
// in the main() method
```

```
class Demo {  
  
    // Command-Line Code ->  
    // javac Demo.java  
    // java Demo 1 2 3  
  
    public static void main(String[] args){  
        for (String elem: args)  
            System.out.println(elem);  
    }  
}
```

Output:

```
1  
2  
3
```

- Apart from the above-mentioned signature of `main`, you could use `public static void main(String args[])` or `public static void main(String... args)` to call the main function in Java.
- The main method is called if its formal parameter matches that of an array of Strings.

3. Explain “Passing argument by values” with example.

- The method is called by passing a value in the pass by value concept.
- As a result, it is referred to as pass by value.
- Passing the parameters by values does not affect the original variable.
- Below is the example of Passing by Value:

Unit-3: Methods and Arrays

PassByValueDemo.java

```
public class Main{
    public static void main(String[] args){
        int num = 5;
        change(num);
        System.out.println(num);
    }
    public static void change(int num){
        num = 10;
    }
}
```

Output:

5

- As we can see from the output above, the pass by value mechanism has no effect on the original values. As **num** variable is a local variable and treated differently for both the methods. So passing value of **num** variable from main() to **num** variable of change() will not affect the value of original **num** variable.

4. Write a program which shows an example of method(function) overloading

- Method Overloading occurs when a class contains multiple methods with the same name but different parameters.
- If we only need to perform one operation, having the methods have the same name improves the readability of the programme.
- If you have to perform addition of the given numbers but there can be any number of arguments, writing the method as a(int,int) for two parameters and b(int,int,int) for three parameters may make it difficult for you and other programmers to understand the behaviour of the method because the names differ.
- So we use method overloading to quickly figure out the programme.

Advantage of method overloading

- Method overloading increases the readability of the program.

Different ways to overload the method

There are two ways to overload the method in java

- By changing number of arguments
- By changing the data type

Method Overloading: changing no. of arguments

- In this example, we have written two methods: the first add() method performs two-number addition, and the second add() method performs three-number addition.
- In this example, we are creating static methods so that we don't have to create instances when calling methods.

Unit-3: Methods and Arrays

```
public class Adder{
    public static int add(int a,int b){return a+b;}
    public static int add(int a,int b,int c){return a+b+c;}

    public static void main(String[] args){
        System.out.println(add(22,22));
        System.out.println(add(22,22,22));
    }
}
```

Output:

44
66

Method Overloading: changing data type of arguments

- In this example, we've created two methods with different data types.
- The first add method takes two integer arguments, and the second add method takes two double arguments.

```
class Adder{
    public static int add(int a, int b){return a+b;}
    public static double add(double a, double b){return a+b;}

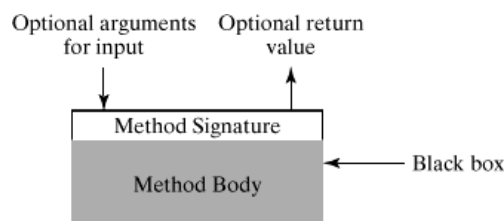
    public static void main(String[] args){
        System.out.println(add(10,10));
        System.out.println(add(11.3,11.6));
    }
}
```

Output:

20
22.9

5. Write a short note on - Method abstraction and stepwise refinement.

- The key to developing software is to apply the concept of abstraction.
- Method abstraction is achieved by separating the use of a method from its implementation.
- The details of the implementation are encapsulated in the method and hidden from the client who invokes the method. This is also known as information hiding.



- When writing a large program, you can use the divide-and-conquer strategy, also known as stepwise refinement, to decompose it into subproblems.
- The subproblems can be further decomposed into smaller, more manageable problems.

Unit-3: Methods and Arrays

Benefits of Stepwise Refinement

- Simpler Program: Rather than writing a long sequence of statements in one method, stepwise refinement breaks it into smaller methods.
- Reusing Methods : The isLeapYear method is defined once and invoked from the getTotalNumberOfDays and getNumberOfDayInMonth methods. (To check if Feb has 28 or 29 days & year is of 365 days or not)
- Easier Developing, Debugging, and Testing: Each subproblem is solved in methods individually that isolates the errors from other methods
- Better Facilitating Teamwork: subproblems can be assigned to different programmers.

6. Explain about arrays, Type of arrays

- An array is generally a collection of similar types of elements with contiguous memory locations.
- An array in Java is an object that contains elements of the same data type.
- Furthermore, array elements are stored in a contiguous memory location.
- It is a data structure in which similar elements are stored.
- A Java array can only hold a fixed number of elements.
- In Java, arrays are index-based; the first element of the array is stored at the 0th index, the second element at the 1st index, and so on.
- we can obtain the length of the array using the length member.

Advantages:

- Code Optimization: It makes the code optimized, we can retrieve or sort the data efficiently.
- Random access: We can get any data located at an index position.

Disadvantages:

- Size Limit: We can store only the fixed size of elements in the array.
- It doesn't grow its size at runtime.
- To solve this problem, collection framework is used in Java which grows automatically.

Types of Array in java:

- There are two types of array: Single Dimensional Array and Multidimensional Array

Single Dimensional Array in Java

Unit-3: Methods and Arrays

- **Syntax to Declare an Array in Java**

```
dataType[] arr; (or)
dataType []arr; (or)
dataType arr[];
```

- **Instantiation of an Array in Java**

```
arrayRefVar=new datatype[size];
```

Example of Java Array

- Let's look at a simple java array example in which we declare, instantiate, initialize, and traverse an array.

```
class Testarray{
    public static void main(String args[]){
        int a[]=new int[3]; //declaration and instantiation
        a[0]=10; //initialization
        a[1]=20;
        a[2]=30;
        //traversing array
        for(int i=0;i<a.length;i++){
            //length is the property of array
            System.out.println(a[i]);
        }
    }
}
```

Output:

```
10
20
30
```

Multidimensional Array - 2D and 3D Array

- In such case, data is stored in row and column based index (also known as matrix form).

- **Syntax:**

```
dataType[ ][ ] arrayRefVar; (or)
dataType arrayRefVar[ ][ ]; (or)
```

- **Example to instantiate 2D Array:**

```
int[ ][ ] arr=new int[3][3]; //3 row and 3 column
```

- **initialize 2D Array element:**

```
arr[0][0]=1;
```

Example of 2D Array

- Let's see the simple example to declare, instantiate, initialize and print the 2Dimensional array.

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

        //declaring and initializing 2D array
        int arr[][]={{1,2,3},{4,5,6},{7,8,9}};
```

Unit-3: Methods and Arrays

```
//printing 2D array
for(int i=0;i<3;i++){
    for(int j=0;j<3;j++){
        System.out.print(arr[i][j]+" ");
    }
    System.out.println();
}
}}
```

Output:

```
1 2 3
4 5 6
7 8 9
```

Example of Multidimensional Array using 3D Array

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

        // arr is the name of 3d array
        int[][][] arr = {{ {10, 20}, {30, 40}}, { {40, 50}, {60, 70}}};

        // for loop to iterate through each element of 3D array f
        for (int tables = 0; tables < 2; tables++){
            for (int rows = 0; rows < 2; rows++){
                for (int columns = 0; columns < 2; columns++){
                    System.out.print ("arr[" + tables + "][" + rows + "]["
                        + columns + "] = " + arr[tables][rows][columns] + "\t");
                }
                System.out.println ();
            }
            System.out.println ();
        }
    }
}
```

```
arr[0][0][0] = 10      arr[0][0][1] = 20
arr[0][1][0] = 30      arr[0][1][1] = 40

arr[1][0][0] = 40      arr[1][0][1] = 50
arr[1][1][0] = 60      arr[1][1][1] = 70
```

7. Methods of Arrays class

```
import java.util.Arrays;
public class Main{
    public static void main(String args[]){
        //defining an array of type String
        String[] countries = {"Poland", "Nepal", "India", "Austria",
            "Japan", "Bhutan", "Canada", "France", "China", "Italy",
            "Germany"};

        Arrays.sort(countries);
    }
}
```

Unit-3: Methods and Arrays

```
//prints the sorted array in ascending order
System.out.println(Arrays.toString(countries));
System.out.println(Arrays.binarySearch(countries, "India"));

}

}

[Austria, Bhutan, Canada, China, France, Germany, India, Italy, Japan, Nepal, Poland]
6
```

The **java.util.Arrays** class contains useful methods for common array operations such as sorting and searching.

- `java.util.Arrays.sort(array_name);` //sorts whole array
- `java.util.Arrays.sort(array_name, 0, 4);` // sorts array from 0 to 3 index
- `Arrays.toString(array_name);` // prints array
- `java.util.Arrays.binarySearch(array_name, key)` // searches key from array using binary search algorithm
- `java.util.Arrays.equals(array1_name, array2_name)` //compares if two array objects point to same memory or not
- `java.util.Arrays.fill(array_name, start_index, end_index, value);` // fills array with given value from start to end where end index is excluded

8. Explain arraycopy() method with the help of a program.

- We can copy an array to another by the arraycopy() method of System class.
- Syntax of arraycopy method

```
public static void arraycopy( Object src, int srcPos, Object dest, int destPos, int length )
```

Example of Copying an Array in Java

```
//Java Program to copy a source array into a destination array in Java
class TestArrayCopyDemo {
    public static void main(String[] args) {
        //declaring a source array
        char[] copyFrom = { 'h', 'e', 'l', 'l', 'o', 'w', 'o',
                           'r', 'l', 'd' };
        //declaring a destination array
        char[] copyTo = new char[copyFrom.length];
        //copying array using System.arraycopy() method
        System.arraycopy(copyFrom, 5, copyTo, 0, 5);
        //printing the destination array
        System.out.println(String.valueOf(copyTo));
    }
}
```

Output:

```
world
```

Unit-3: Methods and Arrays

9. Explain how to pass an array to a method with the help of a program.

- We can pass the java array to method so that we can reuse the same logic on any array.
- Let's see the simple example to get the minimum number of an array using a method.

```
//Java Program to demonstrate the way of passing an array
//to method.
class Testarray2{
//creating a method which receives an array as a parameter
static void min(int arr[]){
    int min=arr[0];
    for(int i=1;i<arr.length;i++) {
        if(min>arr[i]){
            min=arr[i];}
    }
    System.out.println(min);
}
public static void main(String args[]){
    int a[]={30,2,10,5};//declaring and initializing an array
    min(a);//passing array to method
}}
```

Output:

2

10. Explain how to return an array to a method with the help of a program.

- We can return an array from the method in Java.

```
//Java Program to return an array from the method
class TestReturnArray{
    //creating method which returns an array
    static int[] get(){
        int arr[]={10,20,30,40,50};
        return arr;
    }
    public static void main(String args[]){
        //calling method which returns an array
        int arr[]=get();
        //printing the values of an array
        for(int i=0;i<arr.length;i++)
            System.out.println(arr[i]);
    }
}
```

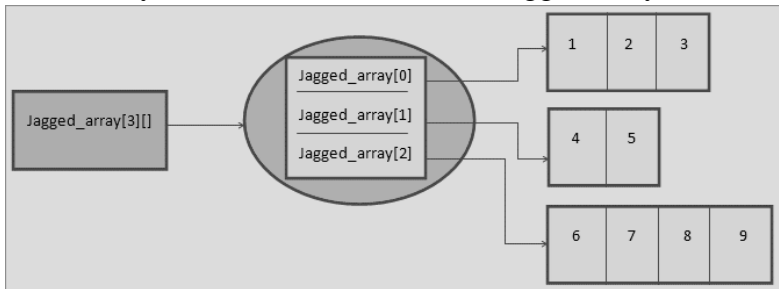
Output:

10
20
30
40
50

Unit-3: Methods and Arrays

11. Explain concept of Jagged /ragged array.

- Jagged array is array of arrays such that member arrays can be of different sizes, i.e., we can create a 2-D arrays but with variable number of columns in each row. These types of arrays are also known as ragged arrays.
- Each row in a two-dimensional array is itself an array. Thus, the rows can have different lengths. An array of this kind is known as a ragged array.



Example of Jagged Arrays

```
class Main{
    public static void main(String[] args){
        // Declare a 2-D array with 3 rows
        int myarray[][] = new int[3][];

        // define and initialize jagged array
        myarray[0] = new int[]{10,20,30};
        myarray[1] = new int[]{40,50};
        myarray[2] = new int[]{60,70,80,90};

        // display the jagged array
        System.out.println("Two dimensional Jagged Array:");
        for (int i=0; i<myarray.length; i++){
            for (int j=0; j<myarray[i].length; j++){
                System.out.print(myarray[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

```
Two dimensional Jagged Array:
10 20 30
40 50
60 70 80 90
```

12. Explain how to pass two-dimensional array to a method to find sum of all the elements of the array.

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int[][] arr = {{1,1},{1,1}}; // Get an array

        // Display sum of elements
        System.out.println("Sum of all elements is: "+ sum(arr));
    }
}
```

Unit-3: Methods and Arrays

```
    }

    public static int sum(int[][] a) {
        int total = 0;
        for (int row = 0; row < a.length; row++) {
            for (int column = 0; column < a.length; column++){
                total += a[row][column];
            }
        }
        return total;
    }
}
```

Output:

Sum of all elements is: 4

PROGRAMS

1. Write a java program to find minimum of two numbers using a function.

```
// Java program to demonstrate the
// use of min() function
// two double data-type numbers are passed as argument
public class Demo {
    public static void main(String args[]){
        double a = 12.123;
        double b = 12.456;

        // prints the minimum of two numbers
        System.out.println(Math.min(a, b));
    }
}
```

Output:

12.123

2. Write a program that creates and initializes a four-integer element array. Calculate and display the average of its values.

```
public class Program{
    public static void main(String[] args) {
        int a []={10,20,30,40};
        int avg=0,sum=0;
        for(int i=0;i<a.length;i++)
            sum=sum+a[i];
        avg=sum/(a.length);
        System.out.println("Sum =" +sum+ " Average="+avg);
    }
}
```

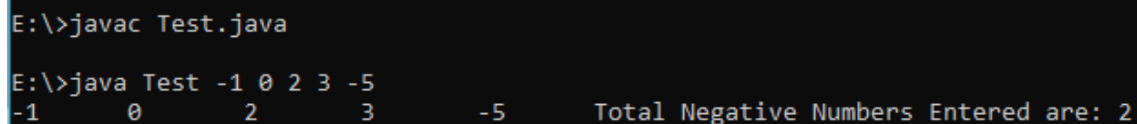
Output:

Sum = 100 Average = 25

Unit-3: Methods and Arrays

3. Write a program which takes five numbers as command line argument from user, store them in one dimensional array and display count of negative numbers.

```
public class Test{
    public static void main(String[] args){
        int neg_count=0;
        int[] num = new int[args.length];
        for(int i=0;i<args.length;i++){
            num[i] = Integer.parseInt(args[i]);
            if(num[i] <0)
                neg_count++;
        }
        for (int e: num)
            System.out.print(e + "\t");
        System.out.print("Total Negative Numbers Entered are: "
            +neg_count);
    }
}
```



```
E:\>javac Test.java

E:\>java Test -1 0 2 3 -5
-1      0      2      3      -5      Total Negative Numbers Entered are: 2
```

4. Write a program that creates an integer array and then uses a for loop to check whether the array is sorted from smallest to largest. If so, it prints “sorted” otherwise it prints “Not sorted”.

```
public class Demo {
    public static void isSorted(int[] a) {
        int i,count=0;
        for(i = 0; i < a.length-1; i++){
            if (a[i] < a[i+1]) {
                count=count+1;
            }
        }
        if(count==a.length-1)
            System.out.println("Sorted");
        else
            System.out.println("Not Sorted");
    }
    public static void main(String[] args){
        int ar[] = {3,5,6,7};
        isSorted(ar);
    }
}
```

Output:



```
Sorted
```

5. Write a java program to perform matrix multiplication using two-dimensional array.

Unit-3: Methods and Arrays

```
public class MatrixMultiplicationExample{
    public static void main(String args[]){
        //creating two matrices
        int a[][]={{1,1,1},{2,2,2},{3,3,3}};
        int b[][]={{1,1,1},{2,2,2},{3,3,3}};

        //creating another matrix to store the multiplication
        //of two matrices
        int c[][]=new int[3][3]; //3 rows and 3 columns

        //multiplying and printing multiplication of 2 matrices
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                c[i][j]=0;
                for(int k=0;k<3;k++){
                    c[i][j]+=a[i][k]*b[k][j];
                } //end of k loop
                System.out.print(c[i][j]+" ");
                //printing matrix element
            } //end of j loop
            System.out.println();//new line
        }
    }
}
```

Output:

```
6 6 6
12 12 12
18 18 18
```