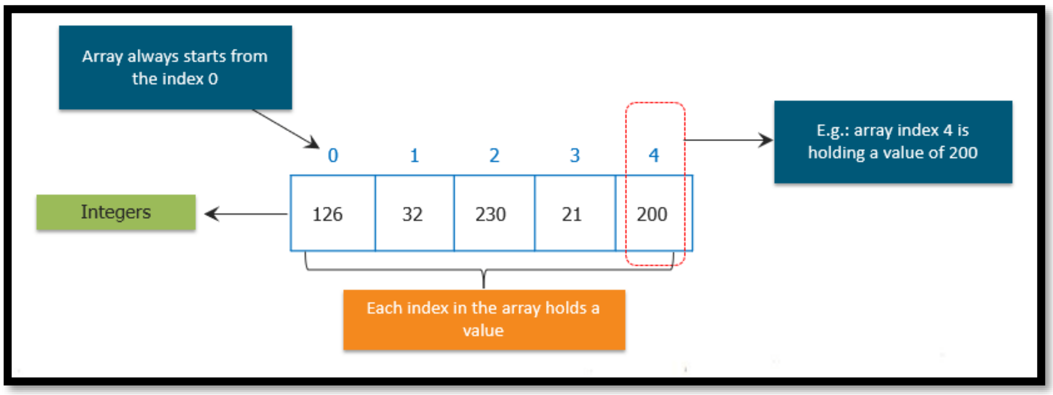
ARRAY IN JAVA

- In Java, an **array** is a data structure that stores a fixed-size sequential collection of elements of the same type.
- These elements can be primitive data types, such as **integers** or **characters**, or objects.
- Arrays in Java provide a way to store and manipulate collections of data in a more organized manner.
- Arrays in Java are declared with a specific data type, followed by square brackets **"[ ]"** indicating the size of the array.


Array always starts from the index 0
0
1
2
3
4
E.g.: array index 4 is holding a value of 200
Integers
126
32
230
21
200
Each index in the array holds a value

Advantage of Array:

- **Easy to use:** Arrays are easy to use and implement, making them a popular choice among programmers.
- **Fast access:** Arrays provide fast and efficient access to elements based on their index, which makes them ideal for storing and retrieving data quickly.
- **Memory efficiency:** Arrays are memory-efficient, as they store data in a contiguous block of memory, which makes them ideal for handling large amounts of data.
- **Easy to manipulate:** Arrays can be easily manipulated using loops, making it easy to perform operations on all elements of an array.

Disadvantage of Array:

- **Fixed size:** Arrays in Java are of fixed size, which means that the size of the array cannot be changed once it is initialized.
- **Lack of flexibility:** Arrays cannot be resized dynamically, which means that if you need to add or remove elements from an array, you need to create a new array with a different size.
- **Inefficient for certain operations:** Arrays are inefficient for certain operations, such as sorting and searching, as these operations can require a lot of computational power and time to execute.
- **Complex data types:** Arrays are not suitable for storing complex data types, such as objects and structures, as they can only store a single data type.

❖ Types of Array

There are **two** types of array:

1. **Single Dimensional Array**
2. **Multidimensional Array**

1.Single Dimensional Array

- **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 see the simple example of java array, where we are going to **declare, instantiate, initialize** and **traverse** an array.

```
//Java Program to illustrate how to declare, instantiate, initialize
//and traverse the Java array.
class Testarray{
public static void main(String args[]){
int a[]=new int[5];//declaration and instantiation
a[0]=10;//initialization
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;
//traversing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

Output:

```
10,20,70,40,50
```

Declaration, Instantiation and Initialization of Java Array

We can declare, instantiate and initialize the java array together by:

- `int a[]={33,3,4,5};` //declaration, instantiation and initialization

```java
//Java Program to illustrate the use of declaration, instantiation
//and initialization of Java array in a single line
class Testarray1{
public static void main(String args[]){
int a[]={33,3,4,5};//declaration, instantiation and initialization
//printing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

Output:

```
33,3,4,5
```

Example 2:

Array Literal in Java

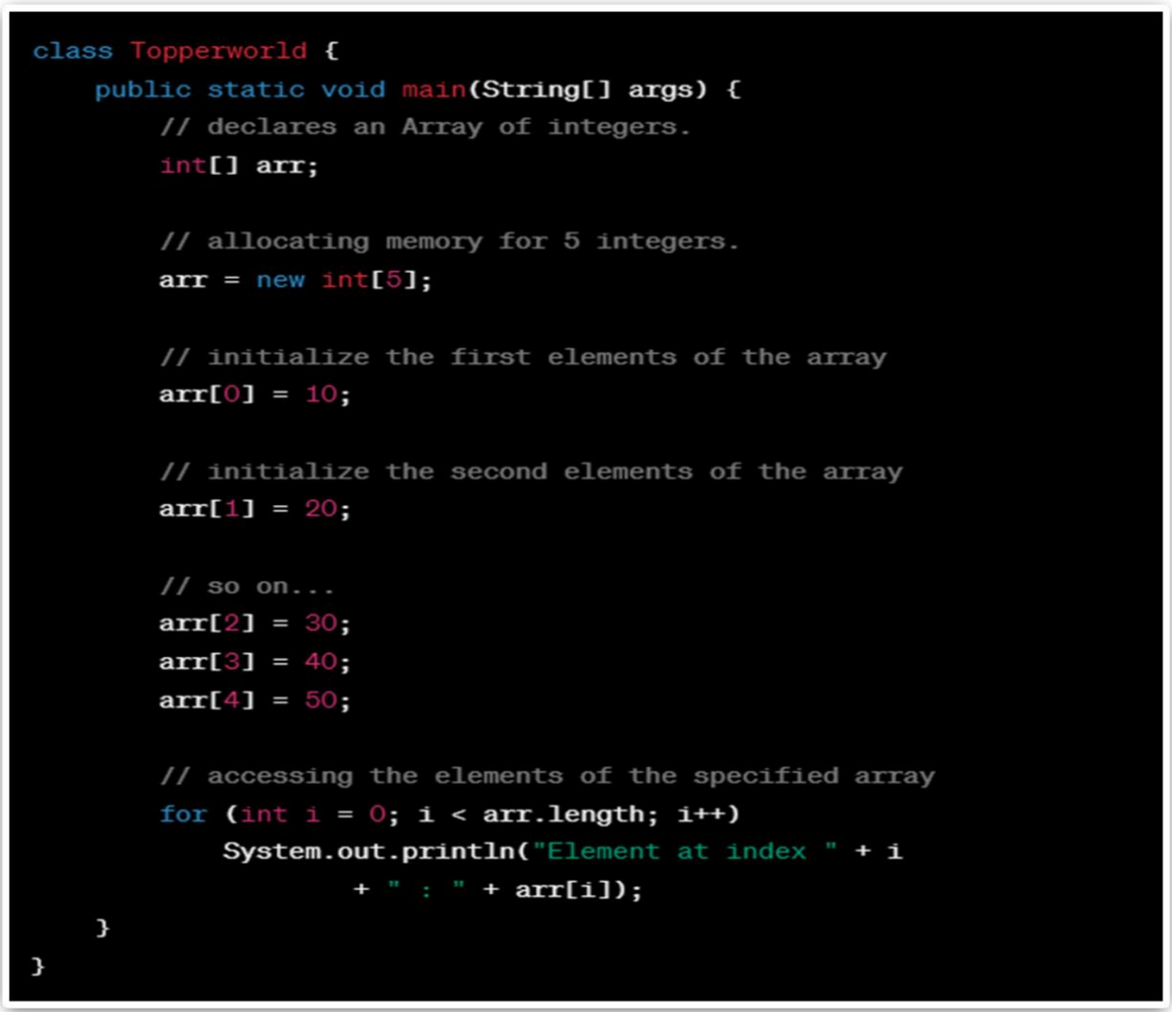
In a situation where the size of the array and variables of the array are already known, array literals can be used.

// Declaring array literal

```java
int[] intArray = new int[]{ 1,2,3,4,5,6,7,8,9,10 };
```

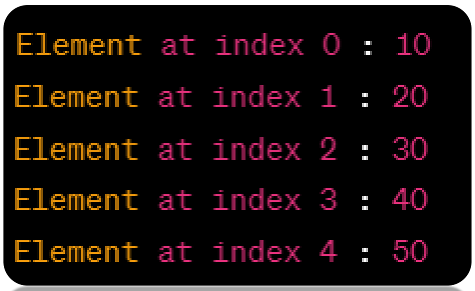
The length of this array determines the length of the created array.

There is no need to write the new **int[]** part in the latest versions of Java.



```
class Topperworld {
    public static void main(String[] args) {
        // declares an Array of integers.
        int[] arr;

        // allocating memory for 5 integers.
        arr = new int[5];

        // initialize the first elements of the array
        arr[0] = 10;

        // initialize the second elements of the array
        arr[1] = 20;

        // so on...
        arr[2] = 30;
        arr[3] = 40;
        arr[4] = 50;

        // accessing the elements of the specified array
        for (int i = 0; i < arr.length; i++)
            System.out.println("Element at index " + i
                    + " : " + arr[i]);
    }
}
```

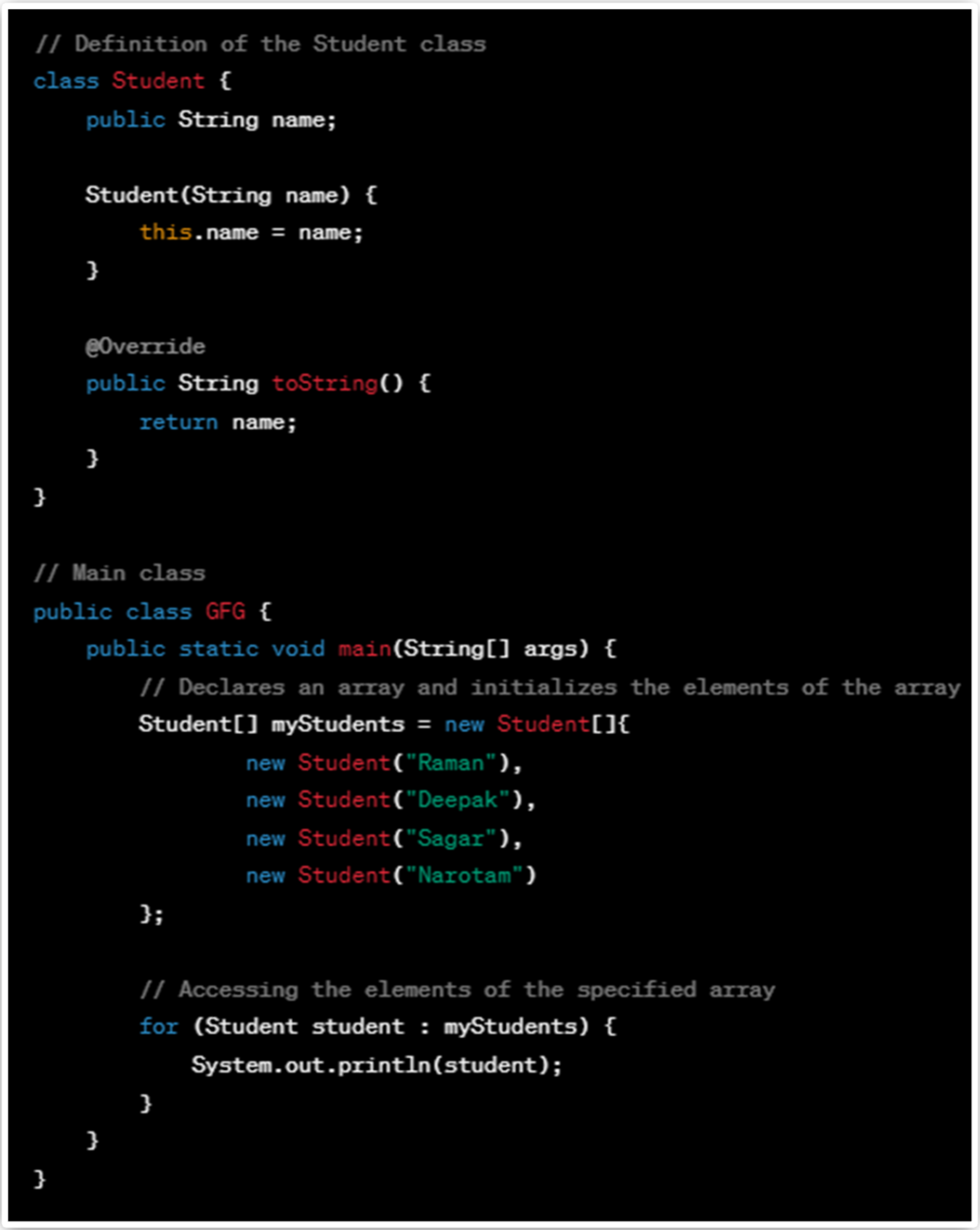
Output:



```
Element at index 0 : 10
Element at index 1 : 20
Element at index 2 : 30
Element at index 3 : 40
Element at index 4 : 50
```

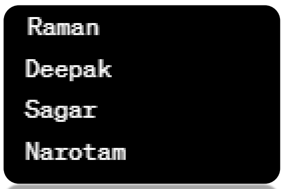
Example 3:

An array of objects is also created like :



```java
// Definition of the Student class
class Student {
    public String name;

    Student(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return name;
    }
}

// Main class
public class GFG {
    public static void main(String[] args) {
        // Declares an array and initializes the elements of the array
        Student[] myStudents = new Student[]{
                new Student("Raman"),
                new Student("Deepak"),
                new Student("Sagar"),
                new Student("Narotam")
        };

        // Accessing the elements of the specified array
        for (Student student : myStudents) {
            System.out.println(student);
        }
    }
}
```

Output:



```
Raman
Deepak
Sagar
Narotam
```

2. Multidimensional Arrays

- Arrays we have mentioned till now are called one-dimensional arrays. However, we can declare multidimensional arrays in Java.
- A multidimensional array is an array of arrays. That is, each element of a multidimensional array is an array itself.

Syntax to Declare Multidimensional Array in Java:

```
dataType[][] arrayRefVar;
```

Example to instantiate **Multidimensional** Array in Java

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

Example of Multidimensional Java Array

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

```
//Java Program to illustrate the use of multidimensional array
class Testarray3{
public static void main(String args[]){
//declaring and initializing 2D array
int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
//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

2 4 5

4 4 5
```

Jagged Array in Java

If we are creating odd number of columns in a 2D array, it is known as a jagged array. In other words, it is an array of arrays with different number of columns.

```java
/Java Program to illustrate the jagged array
class TestJaggedArray{
    public static void main(String[] args){
        //declaring a 2D array with odd columns
        int arr[][] = new int[3][];
        arr[0] = new int[3];
        arr[1] = new int[4];
        arr[2] = new int[2];
        //initializing a jagged array
        int count = 0;
        for (int i=0; i<arr.length; i++)
            for(int j=0; j<arr[i].length; j++)
                arr[i][j] = count++;

        //printing the data of a jagged array
        for (int i=0; i<arr.length; i++){
            for (int j=0; j<arr[i].length; j++){
                System.out.print(arr[i][j]+" ");
            }
            System.out.println();//new line
        }
    }
}
```

Output:

```
0 1 2

3 4 5 6

7 8
```

```
//Java Program to demonstrate the addition of two matrices in Java
class Testarray5{
public static void main(String args[]){
//creating two matrices
int a[][]={{1,3,4},{3,4,5}};
int b[][]={{1,3,4},{3,4,5}};

//creating another matrix to store the sum of two matrices
int c[][]=new int[2][3];

//adding and printing addition of 2 matrices
for(int i=0;i<2;i++){
for(int j=0;j<3;j++){
c[i][j]=a[i][j]+b[i][j];
System.out.print(c[i][j]+" ");
}
System.out.println();//new line
}
}}
```

Output:

```
2 6 8

6 8 10
```