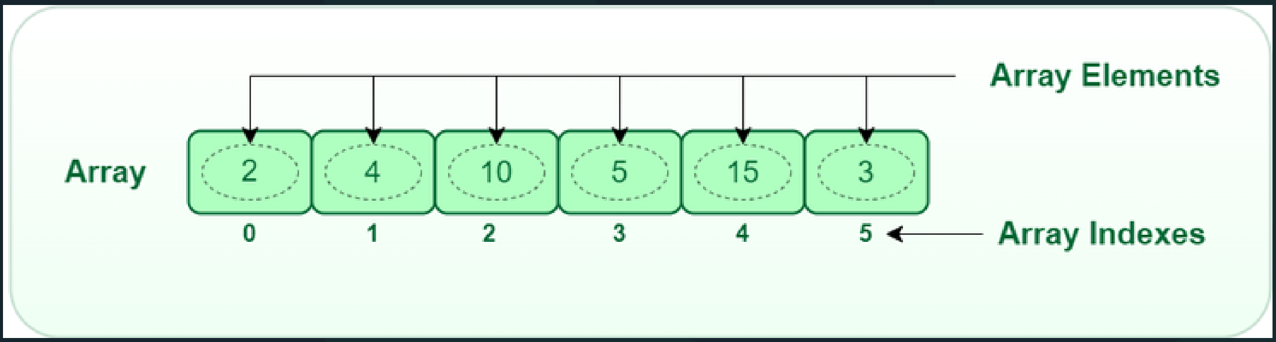
Array

Array:

Suppose you need to take 'n' integer inputs from the user. In such a scenario, arrays come in handy as they provide a means to store and access these values conveniently. Rather than declaring separate variables for each input, you can use an array to store all 'n' inputs together. Arrays allow you to store a collection of values of the same data type in a contiguous memory location.

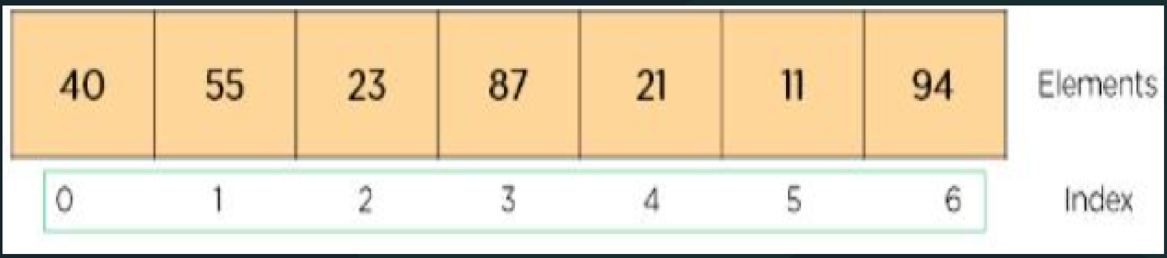


By using an array, you can allocate memory for 'n' integers and store each input value at a specific index in the array. This way, you have a structured storage mechanism for all the inputs, making it easier to access and manipulate them later. You can iterate over the array, perform calculations or operations on the stored values, or use them in any other way required by your program.

In summary, arrays provide an efficient and organized way to store multiple integer inputs from the user when you know the number of inputs in advance. They simplify the process of storing, accessing, and processing the values, making your code more manageable and concise.

What is an Array in Java?

An array refers to a data structure that contains homogeneous elements. This means that all the elements in the array are of the same data type. Let's take an example:



This is an array of seven elements. All the elements are integers and homogeneous. The green box below the array is called the index, which always starts from zero and goes up to n-1 elements. In this case, as there are seven elements, the index is from zero to six. There are three main features of an array:

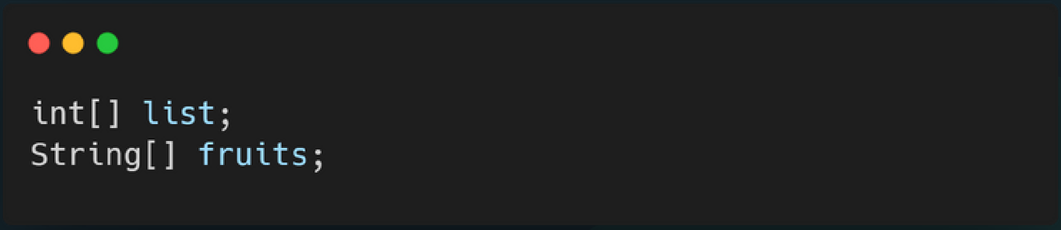
- Dynamic allocation: In arrays, the memory is created dynamically, which reduces the amount of storage required for the code.

- Elements stored under a single name: All the elements are stored under one name. This name is used any time we use an array.

- Occupies contiguous location: The elements in the arrays are stored at adjacent positions. This makes it easy for the user to find the locations of its elements.

Declaring Array Variables

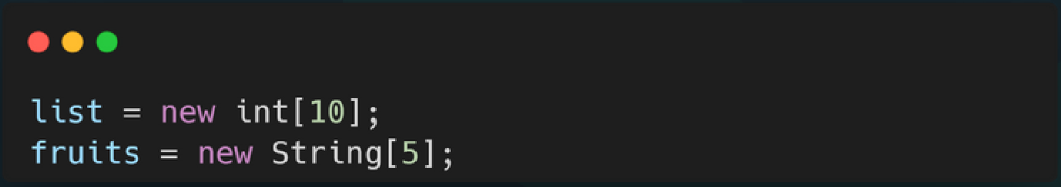
The following statements are examples of array variable declarations:



```
int[] list;
String[] fruits;
```

Creating Array Object

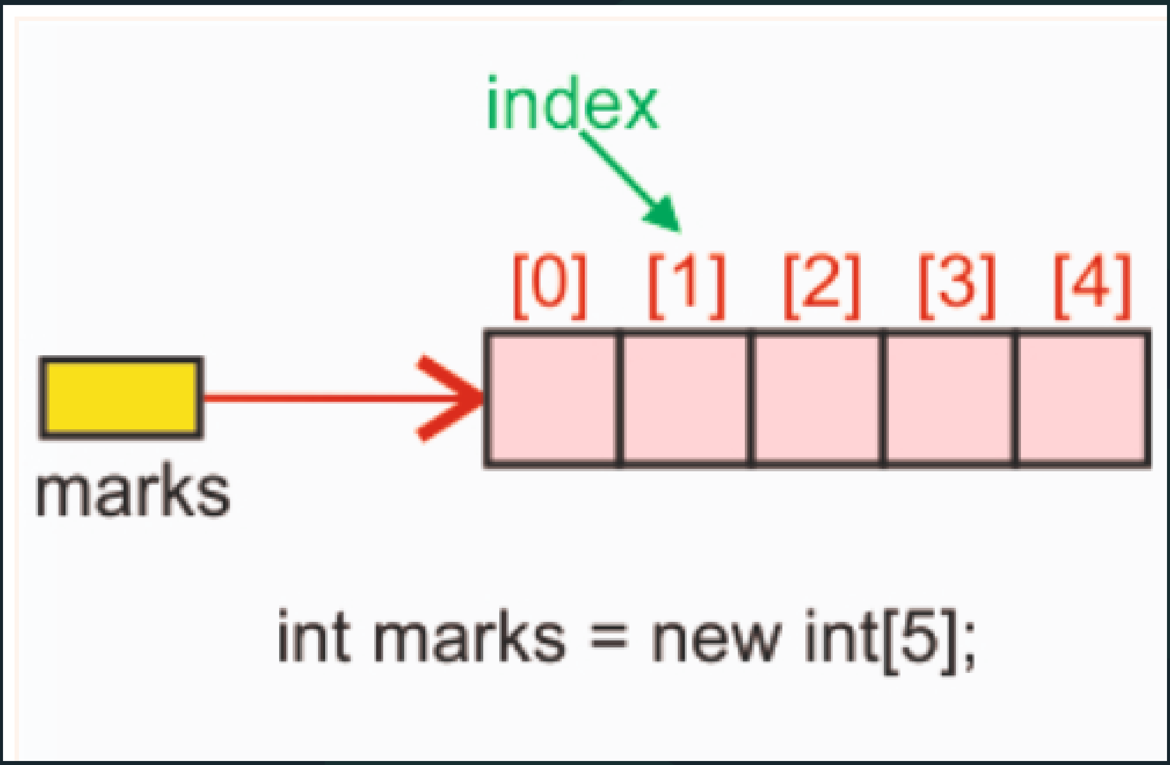
After you declare the array variable, the next step is to create an array object and assign it to that variable, as in the following statement :



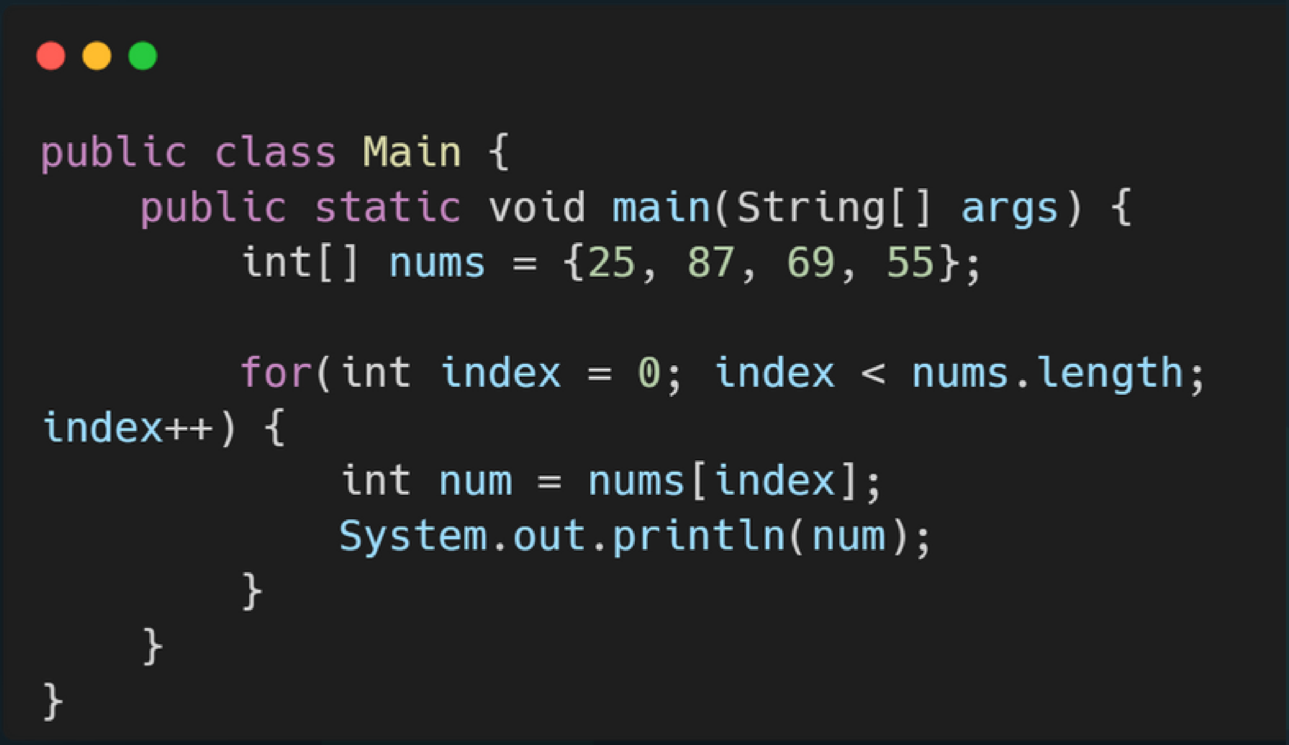
```
list = new int[10];
fruits = new String[5];
```

Declaring array variable and creating array object can be done in single statement, as given

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



Print Array Elements using For Loop



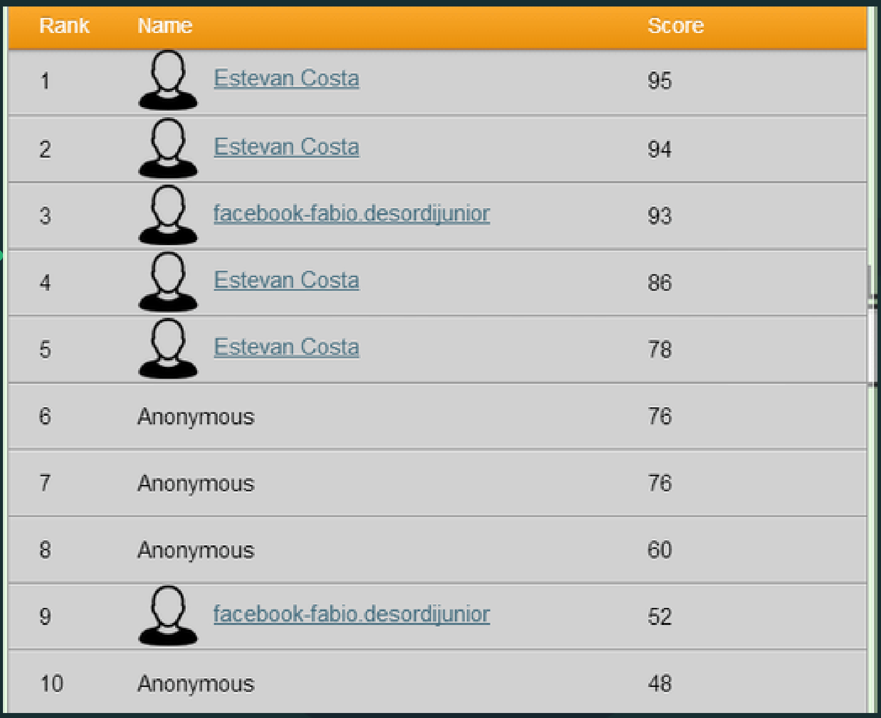
```java
public class Main {
    public static void main(String[] args) {
        int[] nums = {25, 87, 69, 55};

        for(int index = 0; index < nums.length;
index++) {
            int num = nums[index];
            System.out.println(num);
        }
    }
}
```

Real Life Example:

Arrangement of the leader-board of a game can be done simply through arrays to store the score and arrange them in descending order to clearly make out the rank of each player in the game.

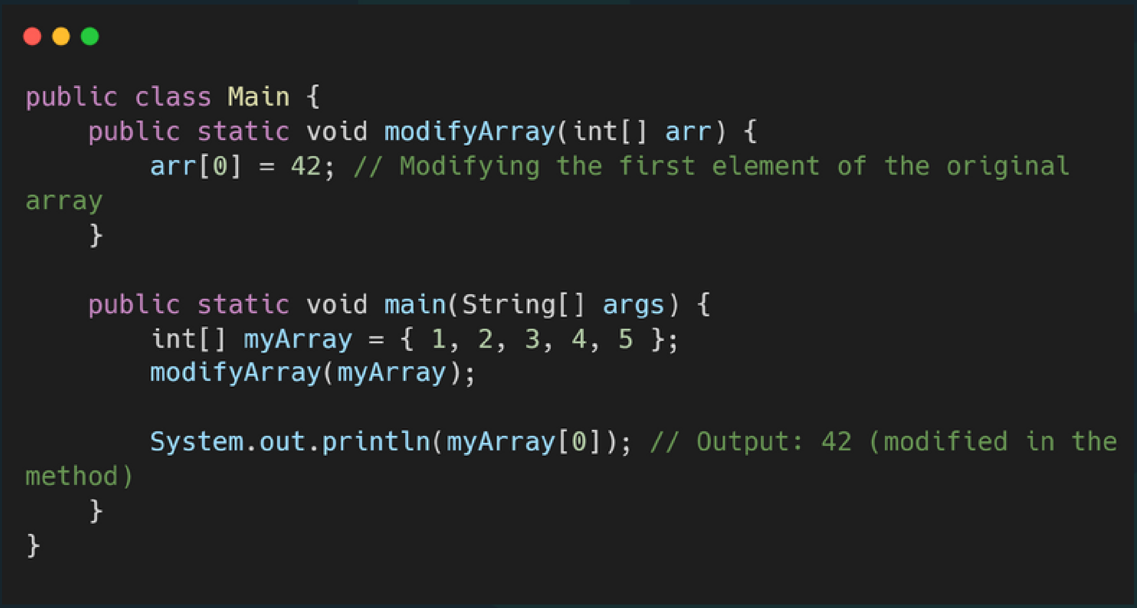
Used of Array to Store the name



Rank	Name	Score
1	Estevan Costa	95
2	Estevan Costa	94
3	facebook-fabio.desordijunior	93
4	Estevan Costa	86
5	Estevan Costa	78
6	Anonymous	76
7	Anonymous	76
8	Anonymous	60
9	facebook-fabio.desordijunior	52
10	Anonymous	48

Pass by Reference

In Java, when you pass an array to a method, you are still passing a copy of the reference to the array. This means that you can modify the contents of the array within the method, and those modifications will affect the original array outside the method. This is sometimes informally referred to as "passing by reference for arrays."



```
public class Main {
    public static void modifyArray(int[] arr) {
        arr[0] = 42; // Modifying the first element of the original
array
    }

    public static void main(String[] args) {
        int[] myArray = { 1, 2, 3, 4, 5 };
        modifyArray(myArray);

        System.out.println(myArray[0]); // Output: 42 (modified in the
method)
    }
}
```