Java Arrays

Introduction

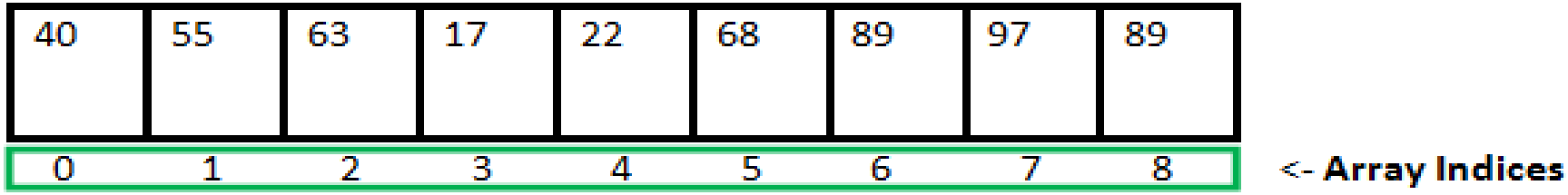
- An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.
- An array reduces the number of lines of code and complexity of a program by making it possible to declare multiple variables within a single array.
- Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables.

Important points to note about Java Arrays

- In Java, all arrays are dynamically allocated.
- Since arrays are objects in Java, we can find their length using the object property *length*.
- A Java array variable can also be declared like other variables with [] after the data type.
- The variables in the array are ordered, and each has an index beginning from 0.

Cont.

- The **size** of an array must be specified by int or short value and not long.
- A array uses indexes which start from to size-1 to refer to its members
- An array looks like shown below



Array Length = 9
First Index = 0
Last Index = 8

Types of arrays in Java

- Java has two main types of arrays namely:
  - One-dimensional array-
  - Multi-dimensional array

One dimensional Array

- As far as an array is concerned, one dimension means it has only one value per location or index.
- One-dimensional array in Java programming is an array with a bunch of values having been declared with a single index.

Creating a one-dimensional array

- The general syntax is as follows:
  - Data_type Variable_name [ ];
  - Or
  - Datatype [ ] variable_name;
- Datatype defines the datatype that the elements of the array will belong to. While variable name is the name you wish to give your array
- Example
  - double marks [ ];
  - double [ ] marks;

Instantiating an array

- There are two ways of instantiating an array.

Option One

- Datatype variableName [ ]= {elements};
- Example
  - double marks [ ]= {10, 11, 9, 21,22};
- In this example we have created an array known as marks and in it we have put five elements.
- Note that all the elements in the array belong to the same datatype

Creating an Array and printing out the values of the array (option 1)

```
package project2022;
public class Project2022 {
    public static void main(String[] args) {
   int marks []={10,11,9,21,22};
   System.out.println(marks[0]);
   System.out.println(marks[1]);
   System.out.println(marks[2]);
   System.out.println(marks[3]);
   System.out.println(marks[4]);
    }
}
```

Cont.

- In this option we have an output statement for each index.
- Note that when referring to the value of an array, you start with the array name followed by the index of the value.
- E.g marks[0]; to refer to the first value in the array known as marks which is 10.
- This method is limited because when you are dealing with an array with so many elements you will end up with so many lines of code.

Option 2

```
public class Java002 {
    public static void main (String [] arg){
    int marks []= {10, 11, 9, 21,22};
    for (int x=0; x<marks.length; x++){
    System.out.println(marks[x]+"");
    } }
}
```

The output of this program will be: 10, 11, 9, 21, 22

Explanation

- The program in the previous page uses a for…loop to print the values of an array.
- Now what if one wants to print specific elements of an array? You can use array indexing.
  - double marks [ ]= {10, 11, 9, 21,22};
- Indexing starts from zero thus 10-index [0], 11-index [1], 9-index[2], 21-index[4], 22-index[4].
- check example below

Example in a program

```
public class Java002 {

    public static void main (String [] arg){
    double marks []= {10, 11, 9, 21,22};
    System.out.println(marks[0]);
    System.out.println(marks[1]);

    }
}
```

Instantiating an array Option Two

- When an array is declared, only a reference of an array is created.
- To create or give memory to the array, you instantiate it using the new keyword and then define the size. As shown below:
  - Datatype variableName [ ]= new datatype [size]
- Example
  - double marks [ ]= new double[5];
- This shows that this array will be known as marks of double datatype and has size 5 which means it can only have a total five elements.

Cont.

- Then now initialize the array by assigning values to each index as shown below.
  - VariableName [0]= value;
  - VariableName [1]= value;
  - VaribaleName[2]=value;
  - VaribaleName[3]=value;
  - VaribaleName[4]=value;
- Lets recreate the example1 array using this approach.

Example in a Program

```
public class Java002 {
    public static void main (String [] arg){
    double marks []= new double[5];
    marks[0]=10;
    marks[1]=11;
    marks[2]=9;
    marks[3]=21;
    marks[4]=22;
    System.out.println(marks[0]); } }
```

- This program will output the value of index 0 which is 10.

Cont.

- Note that the second approach increases the complexity of the program.
- It is mainly used where the value is not known prior for instance when you want the user to enter the value for each index from the keyboard.
- See example below
- Therefore, where the values are well known its advisable to use option one.

Example Three accepting values of an array from the keyboard

```
import java.util.Scanner;
public class Java002 {


    public static void main (String [] arg){
    double marks []= new double[5];
    System.out.println("Enter five
numbers each in its own line");
    Scanner sc=new Scanner(System.in);
    marks[0]= sc.nextDouble();
    Scanner sc1= new Scanner(System.in);
    marks[1]= sc1.nextDouble();
    Scanner sc2= new Scanner(System.in);
    marks[2]=sc2.nextDouble();
    Scanner sc3= new Scanner(System.in);
    marks[3]=sc3.nextDouble();
    Scanner sc4=new Scanner(System.in);
    marks[4]=sc4.nextDouble();
    System.out.println(marks[0]);
    }
}
```

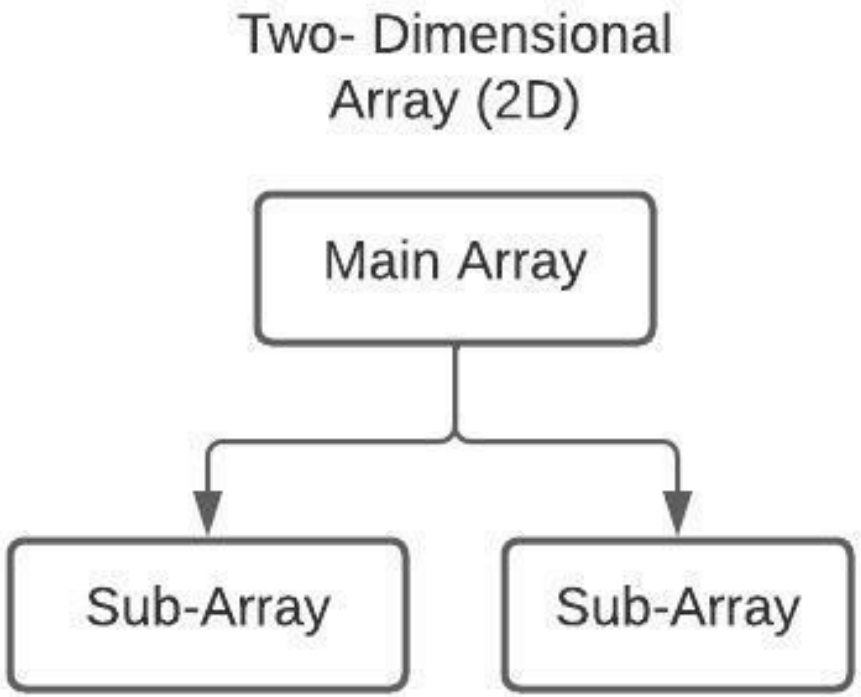
Multi-dimensional Array

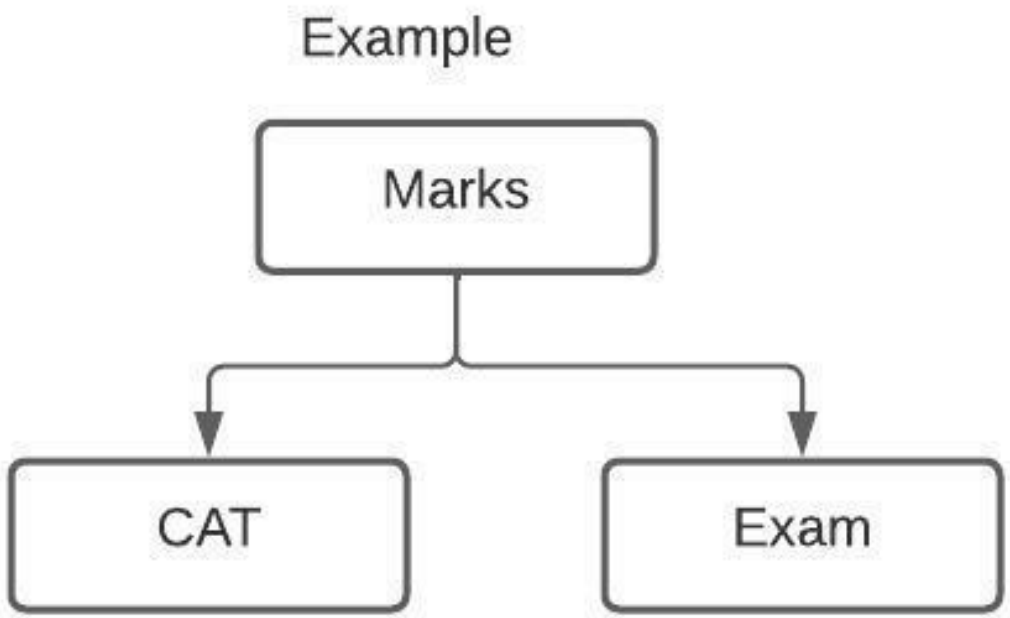
- A multi-dimensional is an array that contains other arrays.
- The most commonly used multi-dimensional array is a two dimension array
- A two dimension array can be represented in the form of a matrix (x, y) where x represents the number of arrays and y represents the number values in each array
- They are very useful especially when one is querying a database because it makes it possible for one to represent multiple columns and rows.

Cont.

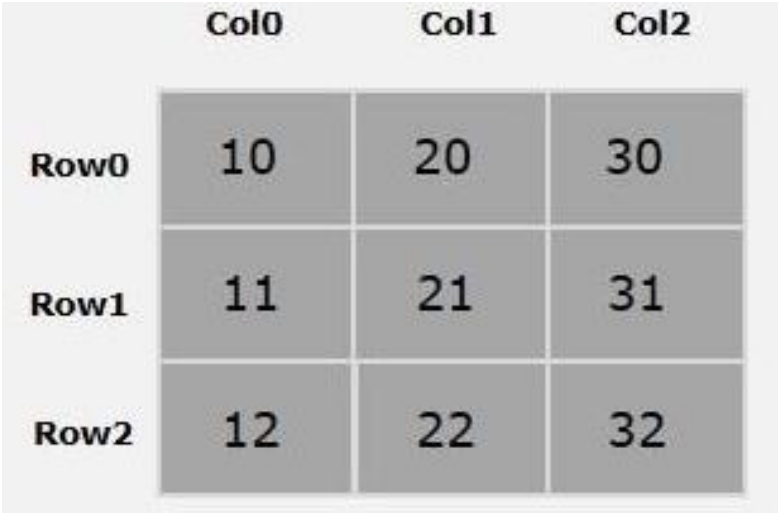
- Think of an example in which you would like to represent the CAT marks for three students as well as exam marks in a single array.
- The array would look as follows: where the first set {5, 10,12} represents the CAT marks and the second set {34, 46,56} represents the exam marks
  - double marks[ ][ ]= { {5, 10, 12}, {34, 46, 56}}
  - So in this case we can easily say that this is (2*3) array
- As you can see the main array is known as marks and inside it we have two other arrays.
- When declaring a two dimension array you have to use two [][] the first one represents the position of the array while the second one represents indexing of the value in the sub-array

Diagrammatic Representation of a 2D array





Cont.

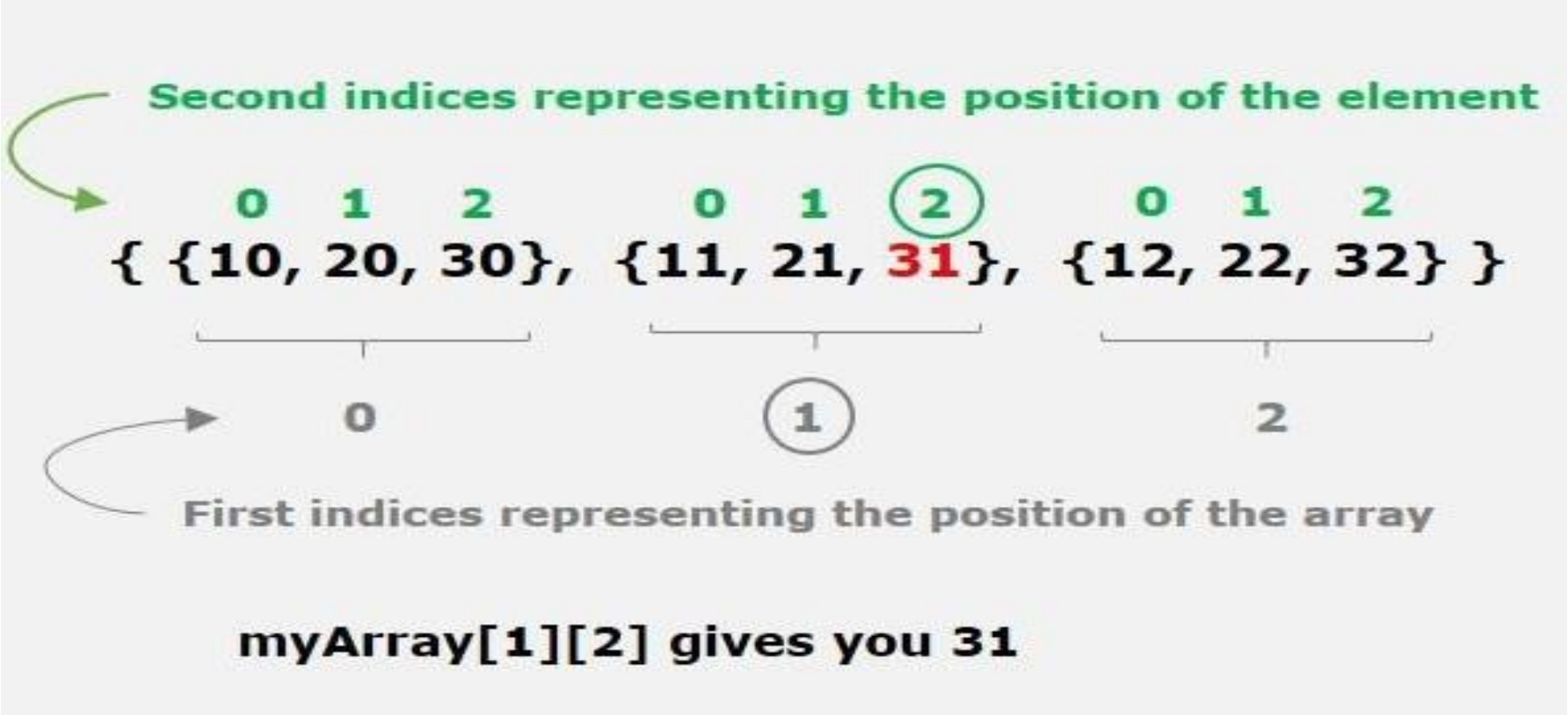


	Col0	Col1	Col2
Row0	10	20	30
Row1	11	21	31
Row2	12	22	32

- This table has three rows and three columns thus it can be said to be (3*3) table.
- How can this be represented in an array?
  - Int[][] myArray = {{10, 20, 30}, {11, 21, 31}, {12, 22, 32} }
- Observe that each row forms a sub-array

Now how can we access the values of this array?

- Lets start by understanding how indexing is done here.



Cont.

- From the above diagram you can see that you are required to pass two index values when you call the array name.
- i.e myArray[1][2].
- The first index represents the position of the sub-array in the main array. Here we have three sub-arrays thus the first one has index [0], the second one has index[1] and the third one has index[2]
- In this case the value we are interested in is 31 which is found in sub-array 2 thus index [1].
- The second index represents the position of the element in its sub-array. In this case 31 is index 2 of its sub-array that’s why when we represent the position of 31 as myArray[1][2].

Example in a Program

```
public class Java002 {


    public static void main (String [] arg){
    int myArray[][]={{10,20,30},{11,21,31}, {12,22,32}};
    System.out.println(myArray[1][2]);
    }
}
```

The output is 31.

Example V2

```
public class Java002 {


    public static void main (String [] arg){
    int myArray[][]={{10,20,30},{11,21,31}, {12,22,32}};
    System.out.println(myArray[0][1]);
    }
}
```

What will be the output?

The End