Java Arrays

Introduction

An array is a collection of similar types of data.

For example, if you want to store the names of 100 people then you can create an array of the string type that can store 100 names: `String[] array = new String[100];`

Here, the above array cannot store more than 100 names. The number of values in a Java array is always fixed.

Declaring an array in Java

```
dataType[] arrayName;
```
`dataType` - it can be primitive data types like `int`, `char`, `double`, `byte`, etc. or Java objects
`arrayName` - it is an identifier

For example: `double[] data;` Here, `data` is an array that can hold values of type `double`.

To define the number of elements that an array can hold, you have to allocate memory for the array in Java. For example,

```
// declare an array
double[] data;

// allocate memory
data = new double[10];
```

Here, the array can store **10** elements. You can also say that the **size or length** of the array is 10.

You can declare and allocate the memory of an array in one single statement. For example:

```
double[] data = new double[10];
```

Initialising Arrays in Java

```
//declare and initialize and array
int[] age = {12, 4, 5, 2, 5};
```

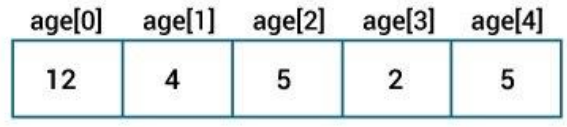
Here, you have created an array named age and initialized it with the values inside the curly brackets.

Note that you have not provided the size of the array. In this case, the Java compiler automatically specifies the size by counting the number of elements in the array (i.e., 5).

In the Java array, each memory location is associated with a number. The number is known as an array index. You can also initialize arrays in Java, using the index number.
For example,

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

```
// initialize array
age[0] = 12;
age[1] = 4;
age[2] = 5;
..
```



Java Arrays initialization

Note:

- Array indices always start from 0. That is, the first element of an array is at index 0.
- If the size of an array is `n`, then the last element of the array will be at index `n-1`.

Accessing Elements of an Array in Java

You can access the element of an array using the index number. Here is the syntax for accessing elements of an array:

```
// access array elements
array[index]
```

Example of accessing array elements using index numbers:

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

    // create an array
    int[] age = {12, 4, 5, 2, 5};

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

Output

```
Accessing Elements of Array:
First Element: 12
Second Element: 4
Third Element: 5
Fourth Element: 2
Fifth Element: 5
```

In the above example, notice that you are using the index number to access each element of the array.

Looping Through Array Elements

Example: Using For Loop

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

   // create an array
   int[] age = {12, 4, 5};

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

Output

```
Using for Loop:
12
4
5
```

In the above example, you are using the for Loop in Java to iterate through each element of the array. Notice the expression inside the loop:

```
age.length
```

Here, you are using the `length` property of the array to get the size of the array.
You can also use the for-each loop to iterate through the elements of an array. For example,

Example: Using the for-each Loop

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

   // create an array
   int[] age = {12, 4, 5};

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

Output

```
Using for-each Loop:
12
4
5
```

Example: Compute Sum and Average of Array Elements

```
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 / (double)arrayLength);

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

Output:

```
Sum = 36
Average = 3.6
```

In the above example, you have created an array of named `numbers`. You have used the `for...each` loop to access each element of the array.
Inside the loop, you are calculating the sum of each element. Notice the line,

```
int arrayLength = number.length;
```

Here, you are using the length attribute of the array to calculate the size of the array. You then calculate the average using:

```
average = ((double)sum / (double)arrayLength);
```

As you can see, you are converting the `int` value into `double`. This is called type casting in Java. To learn more about typecasting, visit Java Type Casting.