Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Java: Array - Enhanced for loop/5aa9f918746fb501b1efcaf9

TUTORIAL

Java: Array - Enhanced for loop

Array and for-each loop

Java 5.0 introduced a new form of for loop, "for-each" loop. The for-each loop is meant specifically for processing all the values in a data structure. When used to process an array, for-each loop can be used to perform the same operation on each value that is stored in the array. If anArray is an array of type BaseType[], then a for-each loop for anArray has the form:

```
for ( BaseType item : anArray )
{
         // process the item
}
```

In this loop, item is the loop control variable. It is being declared as a variable of type BaseType, where BaseType is the base type of the array. (In a for-each loop, the loop control variable must be declared in the loop.) When this loop is executed, each value from the array is assigned to item in turn and the body of the loop is executed for each value. Thus, the above loop is exactly equivalent to:

```
for ( int index = 0; index < anArray.length; index++ )
{
        BaseType item;
        item = anArray[index];  // Get one of the values from the
array
        // process the item
}
```

For example, if A is an array of type int[], then we could print all the values from A with the for-each loop:

```
for (int item : A )
System.out.println( item );
```

and we could add up all the positive integers in A with:

```
int sum = 0;   // This will be the sum of all the positive
numbers in A
for ( int item : A )
{
        if (item > 0)
                sum = sum + item;
}
```

The for-each loop is not always appropriate. For example, there is no simple way to use it to process the items in just a part of an array. However, it does make it a little easier to process all the values in an array, since it eliminates any need to use array indices.

It's important to note that a for-each loop processes the values in the array, not the elements (where an element means the actual memory location that is part of the array). For example, consider the following incorrect attempt to fill an array of integers with 17's:

```
int[] intList = new int[10];
for ( int item : intList )
{   // INCORRECT! DOES NOT MODIFY THE ARRAY!
        item = 17;
}
```

The assignment statement item = 17 assigns the value 17 to the loop control variable, item. However, this has nothing to do with the array. When the body of the loop is executed, the value from one of the elements of the array is copied into item. The statement item = 17 replaces that copied value but has no effect on the array element from which it was copied; the value in the array is not changed.

Following program will print all elements of an array on screen and at last it prints the sum of these elements.

Java

```java
import java.util.Scanner;

class Main
{
    public static void main(String[] args)
    {
      int arr[] = {23, 54, 13, 87, 56};
      int sum=0;
      for(int elem:arr) // elem now is the element itself not the index so we can use it directly.
      {                                  // JVM automatically take care of array size and traversal.
        System.out.print(elem + "\t");
        sum = sum+elem;
      }
      System.out.println("\nSum of array elements is : " + sum);
    }
}
```