Array in JAVA

by Tripti Majumdar

Introduction to Arrays

- An array is a data structure used to process a collection of data that is all of the same type .
- Alternatively, the array, which stores a fixed-size sequential collection of elements of the same type.
- Arrays are indexed by a sequence of integers .
- Classes can use arrays as instance variables to store databases of value/references
- The array elements are accessed through the index.
- Array indices start from 0 to arrayRefVar.length-1.

Declaring Array Variables

Syntax

- dataType[] arrayVariable;

Example :

int roll[]; // declare array variable

roll = **new** int[12]; // create array

<u>We can create arrays of objects too</u>

String b[] = new String[100];

Creating Arrays

- We can create an array by using the **new operator.**
- arrayRefVar = new dataType[arraySize];
- The above statement does two things:
- It creates an array using new dataType[arraySize];
- It assigns the **reference** of the newly created array to the variable arrayRefVar.
- Declaring an array variable, creating an array, and assigning the reference of the array to the variable can be combined in one statement, as shown below: **dataType[] arrayRefVar = new dataType[arraySize];**
- double[] myArray = new double[10];

Array Initialization

- Items enclosed in braces ({})
- Items in list separated by commas
- int number[] = { 10, 20, 30, 40, 50 };

Declare **number** as an array of ints

Compiler uses initializer list to allocate **number** as an array

The *length* Instance Variable

- An array is considered to be an object
- Since other objects can have instance variables, so can arrays
- Every array has exactly one instance variable named length – When an array is created, the instance variable length is automatically set equal to its size .
- The value of length cannot be changed (other than by creating an entirely new array with **new**)
- double[] score = new double[5];
- Given score above, score.length has a value of 5

The *length* Instance Variable

Find the Output :

```
class Test
{
public static void main(String args[])
 {
int i;
int [] myArr=new int[5];
for (i=0;i<10;i++)
   myArr[i]=i; 0,1,2,3,4
   myArr.length = 10;
      for (i=0;i<myArr.length;i++)
      System.out.println(myArr[i]);
  }
}
```

Compilation Error will generate :
Test2.java:7: cannot assign a value to final variable length
myArr.length = 10;

Processing Arrays

Find out the Output :

```
class Test
{
public static void main(String args[])
 {
int i;
long [] myArr=new long[5];
   for ( i=0;i<myArr.length;i++)
    myArr[i]=i;
       for (i=0;i<myArr.length;i++)
```

Note:
Using a value of type long as an array index results in a compilation error. An index must be an int value or a value of a type that can be promoted to int—namely, byte, short or char, but not long.

Array Index Out of Bounds

- Array indices always start with 0, and always end with the integer that is one less than the size of the array
- The most common programming error made when using arrays is attempting to use a non existent array index
- When an index expression evaluates to some value other than those allowed by the array declaration, the index is said to be ***out of bounds***
- An out of bounds index will cause a program to terminate with a run-time error message :

  ArrayIndexOutOfBoundsException
- Array indices get out of bounds most commonly at the first or last iteration of a loop.

Enhanced *for* Statement

- It also known as enhanced for loop
- Suitable for Array
- Introduced 1.5 version onwards
- Can access array elements
- Cannot modify array elements
- Cannot access the counter indicating the index
- Allows iterates through elements of an array or a collection without using a counter –
- Syntax

```
for ( parameter : arrayName ) statement
```

for each loop

To print the value this ARRAY : int [] a={5,10,15,20};

Using normal for loop

```
for (int i=0;i<a.length;i++)
System.out.println(a[i]);
```

Using for each loop

```
for (int a1:a)
System.out.println(a1);
//for each int value of a will print through a1
```

for each loop

Find the output :

==========

```
class Test
{
public static void main(String args[])
 {
 String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[1]="fiat";
for (String i : cars) {
  System.out.println(i);
}
  }
}
```

```
C:\prac-java>java Test
Volvo
fiat
Ford
Mazda
```

Two-Dimensional Array

- A two dimensional array is nothing but list of one dimension array.
- We have to specify the size of both dimensions
- Example :

```
int [][] number=new int [10][40];
```

<u>Other way of declaration :</u>

```
int [][] number=new int [2][];
Number[0]=new int [4];
Number[1]=new int [3];
```

Find the Output

```
class Test
{
public static void main(String args[])
 {
int [] myArr=new int[5];
int [] myArr1={1,2,3};
int [][]matrix={      {1,2},
                {4,5,6},
                {7,8},
                {9,10,11,12}
          };
System.out.println(myArr.length);
System.out.println(myArr1.length);
System.out.println(matrix.length);
System.out.println(matrix[0].length);
```

5

3

Multi-Dimensional Array

- datatype [][][][]…..[]=new type[size1][size2][size3]……[size n]
- int [][][] number=new int[4][3][5];

Find the Output

```
class Test
{
   public static void main(String[] args)
   {
      int[] a = new int[10];

      int[] b = new int[100];
a=b  ;
}
//No Error will generate because Compiler checks only type, not the size
```

Find the Output

```
class Test
{
   public static void main(String[] args)
   {
      int[] a = new int[10];

      int[] b = new int[100];
a=b ;
if(a ==b)
      System.out.println("ok");
else
  System.out.println("not possible");
   }

}
```

Output : ok

Find the Output

```
class Test
{
   public static void main(String[] args)
   {
      int[] a = new int[10];

      int[] b = new int[100];
if(a ==b)
      System.out.println("ok");
else
  System.out.println("not possible");
   }

}
```

Output : not possible

Find the Output

```
class Test
{
public static void main(String args[])
 {
int[] a1 = {42, -7, 1, 15};
int[] a2 = {42, -7, 1, 15};
if (a1 == a2)
{ System.out.println(“Equal");// false!
 }
else
 System.out.println(“Not Equal");
}
}
```

If returns false because separate memory allocation for a1 and a2

Find the Output

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

 //An array does not know how to print itself:
int[] a11 = {42, -7, 1, 15};
System.out.println(a11);
  }
}
```

Note: An array does not know how to print itself. Returns garbage.