Arrays

18-02-2013 | MONDAY

P109

1)
```
package com.jspiders.arrays;

public class Demo1
{
  psvm (String[] args)
  {
    Sop("Program starts...");
    int[] intArray = new int[5];
    Sop("array name : "+intArray);
    Sop("Total elements: "+intArray.length);
    Sop("2nd element : "+intArray[1]);
    Sop("Array elements are");
  // normal for loop
    for(int i=0; i<intArray.length; i++)
    {
      Sop(i+" element is "+intArray[i]);
    }
    Sop("-----------");
  // for each loop
    for(int k:intArray)
    {
      Sop("element "+k);
    }
    Sop("Program ends...");
  }
}
```

O/P:
```
Program starts...
array name : [I@3479404a
Total elements : 5
2nd element : 0
Array elements are
```

1*

Declaration:

arraytype [ ] arrayname ;

or

arraytype arrayname[ ];

2*

Initialize:

arrayname = new arraytype[size]

always int type

O/p continued...

```
0 element is 0
1 element is 0
2 element is 0
3 element is 0
4 element is 0
-----------
element 0
element 0
element 0
element 0
element 0
Program ends
```

2) package com.jspiders.[redacted]array;

```
import java.util.Arrays;

public class Demo2
{
  psvm (String[] args)
  {
    Sop ("Program starts...");
    int[] intArray = new int[5];
    intArray[0] = 67;
    intArray[1] = 12;
    intArray[2] = 35
    intArray[3] = (int) 28.98;
    intArray[4] = 41;
    Sop ("array elements before sort:");
    for (int k: intArray)
    {
      Sop (k);
    }

    Arrays.sort (intArray);
    Sop ("array elements after sort:");
    for (int k: intArray)
    {
      Sop (k);
    }

    Sop ("Program ends...");
  }
}
```

before sort

0	67	int type
1	12	-"-
2	35	-"-
3	28	-"-
4	41	-"-

after sort

12	0
28	1
35	2
41	3
67	4

O/P: Program starts:
array elements before sort:
67
12
35
28
41

Continued....

array elements after sort
12
28
35
41
67
Program ends...

3) P111

```
package com.jspiders.arrays;
import java.util.Arrays;
public class Demo3
{
    psvm (String[] args)
    {
        Sop ("Program starts...");
        String[] strArray = new String[5]; // array of String type
        strArray[0] = "demo";
        strArray[1] = "zen";
        strArray[2] = "abc";
        strArray[3] = "Sample";
        strArray[4] = "132";
        Sop ("array elements");
        for (String s1 : strArray)
        {   Sop (s1);
        }
        Arrays.sort (strArray);
        Sop ("sorted elements");
        for (String s1 : strArray)
        {   Sop (s1);
        }
        Sop ("Program ends...");
    }
}
```

O/p :-

```
Program starts...
array elements
demo
zen
abc
Sample
132
```

continued...

```
sorted elements
132
abc
demo
Sample
zen
Program ends...
```

4)
```
package com.jspiders.arrays;
class A
{
}
class B extends A
{
}
class C extends B
{
}
public class Demo4
{
  psvm(String[] args)
  {
    Sop("Program starts...");
    A[] array1 = new A[5]; // array of A type
    array1[0] = new A();
    array1[1] = new C(); // C casted to A type
    array1[2] = new B(); // B casted to A type
    array1[3] = new A();
    array1[4] = new C(); // C casted to A type
    Sop("Array elements");
    for(A a1: array1)
    {
      Sop(a1);
    }
    Sop("Program ends...");
  }
}
```

19-02-2013 | Tuesday

Arrays

1*) Array is a Collection of homosenins
In an array we can store similar

2*) Derived array are declared using derived Class.

3*) Array can be intialized using
* Dimension
* Array intializer

4*) Ex:- Dimension:
int[] array1 = new int[5]

While using dimension the size of the array should be specified. Whenever an array is intialized using dimension the default value will be stored in each element.

5*) Ex:- Array Initialization:
int[] array1 = {10,20,30,40,50};

Using array initializer the elements are specified with comma separation.

6*) We can refer each element of an array using index.

7*) Java provides class named Array which contains several methods like sort and search...

8*) If we declare the array of Object type, we can store array type of data.

9*) In this type of array each element is of Object type.

P113

3)
```
import java.util.Scanner
class A
{
}
class B
{
}
~~public class Demo~~
```

```
public class Demo1
{
  psvm (String[] args)
  {
    Object[] objArray = new Object[5];
    ObjArray[0] = "test";
    ObjArray[1] = new class A;
    ObjArray[2] = new class B;
    ObjArray[3] = new StringBuilder("demo");
    ObjArray[4] = new Scanner(System.in);
    Sop("obj array elements");
    for (Object obj : objArray)
    {
      Sop(Obj);
    }
    Sop("Program ends...");
  }
}
```

O/p:-
```
Program starts...
Object array element
test
Com.jspiders.arraydemo.A@3eca90
Com.js
demo
java.util.Scanner[delimiters
```

Wrapper Class

1) Java provides Wrapper class to convert primitive type to an Object type.

2) For every primitive type, Wrapper class type is existing

3) All wrapper classes are available in java.lang package.

4) The numeric inherits an abstract class by Name Number.

5) All Wrapper Classes are final Class.

6) In each Wrapper Class the toString() method, equals() and hashcode() methods are Overrided.

Data type	Wrapper Class	
byte	Byte	byte Obj
short	Short	short obj
int	Integer	int obj
long	Long	long obj
float	Float	float obj
double	Double	double obj
char	Character	char obj
boolean	Boolean	boolean obj

P114

4) package com.jspiders.arraydemo;

```
public class Demo2
{ psvm (String[] args)
  {
    Sop ("Program starts...");
    Integer intObj = new Integer(56); // boxing Operation
/*  Integer intObj = 56;  auto boxing [from jdk 1.5 onwards]*/
    Sop ("intObj:" + intObj);
    int k = intObj.intValue(); // unboxing operation
    Sop ("k = " + k);
    Sop ("Program ends...");
  }
}
```

O/P:

```
Program starts
intObj: 56
k: 56
Program ends
```

17) Converting primitive type to Object type is known as boxing operation. The boxing operation is done by using Wrapper Class.

18) Converting a boxed Object back to a primitive type is known as unboxing operation

19) If compiler does boxing on its own its known as auto boxing.

20) Each wrapper class provides a method to get the primitive value [intValue()].