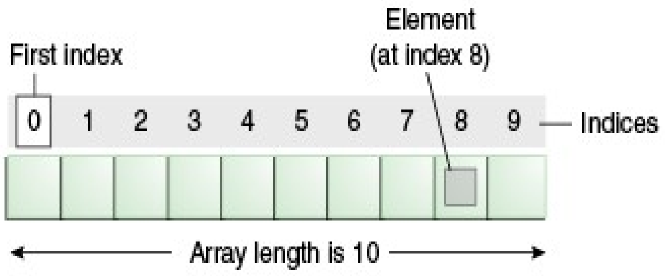
Chapter 2: Array in java

2.1 One Dimensional Array in java:

So far you have been working with variables that hold only one value. The integer variable held only one integer number and string variables just one long string of text. An array is a way to hold more than one value at a time it's like a list of items.
An array is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed.



Each item in an array is called an element, and each element is accessed by its numerical index.

Syntax of assign value into array:

```
Data type []  arrya_name={value 1, Value 2,…..value n};
```

e.g.

```
int arr [] ={ 10, 20, 30, 40, 50 };
```

the position in an array start at 0 and go up sequentially. Each position in the array can hold the value. In above example the value 10 store in arr[0] location, the value 20 store on arr[1] location and so on up to last elements in array arr.

Other way, to create array object in java see below.

Syntax:

```
Data type [] array_name = new data type [size];
OR
Data type [] array_name;
array_name=new int [size];
```

e.g
int [] arr = new int [7];
you start with your array name ,followed by the equal sign after the equal sign you need the java keyword new and then your data type again. After the datatype comes a pair of square brackets with size. In above example see we are telling java to setup an array with 7 positions in it.
Now to assign values to various positions in array see below.

```
arr [0] = 10;
arr [1] = 20;
arr [2] = 12;
arr [3] = 34;
arr [4] = 45;
arr [5] = 50;
arr [6] = 60;
```

or

```
int [] arr = { 10 ,20, 12, 34, 45,50 ,60};
```

This method of setting up an array uses curly brackets after the equal sign. In between the curly brackets you type out the values that the array will hold. The first value on position 0th ,second value on position 1st and so on.

Example: Assign any five values in one dimensional array and display using for loop:

```
Public class arraydemo
{
        public static void main(String[] args)
        {
                int [] arr = new int[5];
                  arr[0]=10;
                  arr[1]=15;
                  arr[2]=18;
                  arr[3]=22;
                  arr[4]=25;

            System.out.println("\n now display array elements \n");
        for(int i=0;i<5;i++)
            {
              System.out.println("elements = " + arr[i]);
            }
      }
}
```

Output:

```
 now display array elements
elements = 10
elements = 15
elements = 18
elements = 22
elements = 25
```

String array:

Syntax:

```
String [] stringarray_name= { “hello” , “how” , “are” ,”you” };
```

Or

You can place strings of text into array.

e.g.

```
String [] array_name= new String [size];
array_name [0] = “hello”;
array_name [1] = “how”;
array_name [2] = “are”;
array_name [3] = “you?”;
```

The above code set up a string array with 4 positions text is assign to each position in the array

Example:

```
publicclassArrayDemo
{
        public staticvoid main(String[] args)
        {
                String [] arr = new String[5];
                arr[0]="Hello";
                arr[1]="How";
                arr[2]="are";
                arr[3]="you";
              arr[4]="?";

            System.out.println("\n now display array elements \n");
        for(int i=0;i<5;i++)
            {

              System.out.println("elements = " + arr[i]);
            }
      }
}
```

Output:

```
 now display array elements
elements = Hello
elements = How
elements = are
elements = you
elements = ?
```

Boolean Array:

Syntax:

```
Boolean [] boolean_array_name = {true, false, true, true, false};

Or

Boolean [] a=new Boolean[3];
a[0]=true;
a[1]=false;
a[2]=true;
```

Example:

```
package ABC;

publicclassDemoBoolean
```

```
{
        publicstaticvoid main(String[] args)
        {
          Boolean[] a= {true, false, true, true, false};
inti;
for(i=0;i<a.length ;i++)
          {
         System.out.println("Element a["+ i +"] = "+a[i]);
          }
 }
}
```

```
Output:
Element a[0] = true
Element a[1] = false
Element a[2] = true
Element a[3] = true
Element a[4] = false
```

Sort Array elements:

Here, Array sort method is inbuilt in java.so allow you to sort your array elements. You first need to reference a java library is called arrays.

E .g

import java.util.Arrays;

Code:

Arrays.sort (Array_name);

Example:

```
package ABC;
import java.util.Arrays;
publicclass ArraySort
{
        publicstaticvoid main(String[] args)
        {
                int [] arr = newint[5];
                 arr[0]=45;
                 arr[1]=36;
                 arr[2]=23;
                 arr[3]=47;
                 arr[4]=22;

    System.out.println("\n now display array elements before sort\n");
for(int i=0;i<5;i++)
    {

     System.out.println("elements = " + arr[i]);
    }
    Arrays.sort(arr);
    System.out.println("\n now display array elements after sort\n");
for(int i=0;i<5;i++)
```

```
      {
            System.out.println("elements = " + arr[i]);
      }
 }
}
```

Output:

```
 now display array elements before sort

elements = 45
elements = 36
elements = 23
elements = 47
elements = 22

 now display array elements after sort

elements = 22
elements = 23
elements = 36
elements = 45
elements = 47
```

2.2 Multidimensional Array in java:

Syntax:

```
Data type [][] array_name =new int [row_size][col_size];
```

E.g.

```
int [][] arr =new int [3][3] ;

arr [0][0]=10;
arr [0][1]=12;
arr [0][2]=15;
arr [1][0]=22;
arr [1][1]=35;
arr [1][2]=47;
arr [2][0]=65;
arr [2][1]=77;
arr [2][2]=25;
```

So graphically represent see below.

10 [0][0]	**12** [0][1]	**15** [0][2]
22 [1][0]	**35** [1][1]	**47** [1][2]
65 [2][0]	**77** [2][1]	**25** [2][2]

In above syntax, in the same way as normal array declaration excepted you have two sets of square brackets. The first set of square bracket is for rows and second set of square bracket is for columns.

In above line the size of row is three (3) and size of column is also three (3) so total nine elements held by this array.

Example1: assign elements in 2D array and Display it

```
package ABC;

publicclass Array2d
{
publicstaticvoid main(String[] args)
  {
        int [][] arr =new int [2][2];
        arr[0][0]=10;
        arr[0][1]=20;
        arr[1][0]=30;
        arr[1][1]=40;

        System.out.println("show elements in array arr");
        for(int i=0;i<2;i++)
        {
                for(int j=0;j<2;j++)
                {
                  System.out.println("Elements arr["+ i +"]["+j+"]=" +arr[i][j]);
                }
        }
   }
}
```

Output:

```
show elements in array arr
Elements arr[0][0]=10
Elements arr[0][1]=20
Elements arr[1][0]=30
Elements arr[1][1]=40
```

Example 2: copy elements from fist array into other array.

```
package ABC;
public class CopyArray
{
        publicstaticvoid main(String[] args)
        {
        int [][] a =newint [2][2];
        int [][] b=newint [2][2];
        a[0][0]=10;
        a[0][1]=15;
        a[1][0]=25;
        a[1][1]=27;
        for(int i=0;i<2;i++)
        {
                for(int j=0;j<2;j++)
                 {
                        b[i][j]=a[i][j];
                 }
        }
        System.out.println("show elements in array B after copy from array A");
        for(int i=0;i<2;i++)
        {
                for(int j=0;j<2;j++)
                 {
                        System.out.println("Elements arr["+ i +"]["+j+"]=" +b[i][j]);
                 }
        }

   }
}
```

Output:

```
show elements in array B after copy from array A
Elements arr[0][0]=10
Elements arr[0][1]=15
Elements arr[1][0]=25
Elements arr[1][1]=27
```

Example 3: two 2D array multiplications into Other 2D array [summer 2013]

```
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass Mul2Darray
{
publicstaticvoidmain(String[] args) throwsNumberFormatException, IOException
        {
        int[][] a =newint [5][5];
        int[][] b =newint [5][5];
        int[][] c =newint [5][5];
        int row, col;
        DataInputStream s1=new DataInputStream(System.in);
        System.out.println ("Enter numbers of row you want...");
        row=Integer.parseInt(s1.readLine());
        System.out.println ("Enter numbers of col you want...");
        col=Integer.parseInt(s1.readLine());
        System.out.println("Now enter Elements into array A");
        for(int i=0;i<row;i++)
        {
                for(int j=0;j<col;j++)
                 {
                        a[i][j]=Interger.parseInt(s1.readLine());
                }
        }
        System.out.println("Now enter Elements into array B");
        for(int i=0;i<row;i++)
        {
                for(int j=0;j<col;j++)
                {
                        b[i][j]=Interger.parseInt(s1.readLine());
                }
                }
        for(int i=0;i<row;i++)
        {
                for(int j=0;j<col;j++)
                {
                        c[i][j]=a[i][j]*b[i][j];
                }
        }
        System.out.println("elements in C[][] after Mul of arrays A and B ");
        for(int i=0;i<row;i++)
        {
                for(int j=0;j<col;j++)
                {
                        System.out.println("Elements arr["+ i +"]["+j+"]=" +c[i][j]);
                }
        }
    }
}
```

Output:

```
Enter numbers of row you want...
Enter numbers of col you want...
2
Now enter Elements into array A
12
12
12
12
Now enter Elements into array B
12
12
12
12
elements in C[][] after Mul of arrays A and B
Elements arr[0][0]=144
Elements arr[0][1]=144
Elements arr[1][0]=144
Elements arr[1][1]=144
```

Example 4: Transposed 2D array

```
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass ArrayTrans
{
publicstaticvoid main(String[] args)throws NumberFormatException, IOException
        {
        int [][] a =newint [5][5];

        int row,col;
        DataInputStream s1=newDataInputStream(System.in);
        System.out.println ("Enter numbers of row you want..");
        row=Integer.parseInt(s1.readLine());
        System.out.println ("Enter numbers of col you want...");
        col=Integer.parseInt(s1.readLine());
        System.out.println("Now enter Elements into array A");
        for(int i=0;i<row;i++)
        {
                for(int j=0;j<col;j++)
                 {
                        a[i][j]=Integer.parseInt(s1.readLine());
                 }
        }

        System.out.println("show elements after Transpose Metrix A ");
        for(int j=0;j<row;j++)
        {
                for(int i=0;i<col;i++)
                 {
                 System.out.println("Elements arr["+ i +"]["+j+"]=" +a[i][j]);
                 }
        }

   }
}
```

Output:

```
Enter numbers of row you want..
2
Enter numbers of col you want...
2
Now enter Elements into array A
12
23
34
45
show elements after Transpose Metrix A
Elements arr[0][0]=12
Elements arr[1][0]=34
Elements arr[0][1]=23
Elements arr[1][1]=45
```

2.3 String Class:

The String class represents character strings. String class is an immutable. The **java.lang.String** class can provide a lot of methods to work on string. By the help of these methods, we can perform operations on string such as copy, trimming, concatenating, converting, comparing, replacing strings etc.

e.g.

String string_name = "abc";

is equivalent to:

char data [] = {'a', 'b', 'c'};

String stringname= new String (data);

Example: illustration of Stringclass:

```
publicclassStringClass
{
        publicstaticvoid main(String [] args)
        {
           String s1 = "abc";
        char data[] = {'p', 'q', 'r'};
           String s2 =new String(data);

           System.out.println("string s1="+s1);
           System.out.println("string s2="+s2);
    }
}
```

Output:

```
string s1=abc
string s2=pqr
```

Let us see below list of string methods:

Return type	**Method name**	**description**
char	charAt(int index)	Return char at specified index
int	compareTo(String anotherString)	Compare two strings.
int	compareToIgnoreCase (String str)	Compare two strings but ignore case
String	concat(String str)	Connect the specified string to the end of this string
boolean	contentEquals (charseuences cs)	Compare this string to the specified charSequence.
boolean	contentEquals (StringBuffer cs)	Compare this string to the specified StringBuffer
boolean	endsWith(String suffix)	Tests if this string ends with the specified suffix
boolean	equals(Object	Compares this string to the specified object

	anotherObject)	
boolean	equalsIgnoreCase(String anotherstring)	Compare this string to another string, ignore case
int	indexOf(int ch)	Return the index of the specified character.
int	indexOf(int ch, int fromIndex)	Return the index with in this string of the first occurrence of the specified character.
int	indexOf(String str)	Return the index within this string of the first occurs of the specified substring.
boolean	isEmpty()	Return true if length is zero(0).
int	lastIndexOf(int ch)	Return the index with in this string of the last occurs of the specified character.
int	lastIndexOf(int ch, int fromIndex)	Return the index within this string of the last occurs of the specified character.
int	lastIndexOf(String str)	Return the index with in this string of the last occurs of the specified substring.
int	length()	Return the length of this string
String	replace(char oldchar, char newchar)	Return a new string resulting from replacing all occur of old chat in this string with newchar.
String	replaceAll(string reg, String rep)	Replace each substring of this string that matches the given reg with the given rep
String[]	split(string reg)	Split the string if match of the given reg.
boolean	startsWith(string pre)	Test if string starts with the specified pre.
String	substring(int stratindex)	Return a new string that is a sub string of this string.
String	substring(int stratindex, int endindex)	Return string that is a substring of this string.
String	toLowerCase()	Convert all character in this string to lower case.
String	toUpperrCase()	Convert all character in this string to uppercase.
static string	valueOf(boolean var)	Return the string representing of the boolean argument.
static string	valueOf(char ch)	Return the string representing of the char argument.
static string	valueOf(double d)	Return the string representing of the double argument.
static string	valueOf(int a)	Return the string representing of the integer argument.

2.4 StringBuffer Class:

In java, StringBuffer class is a mutable (modifiable) means we can modify the string using StringBuffer class methods. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

Let us see below list of StringBuffer Constructor:

Constructor	**description**
StringBuffer():	Creates an empty string buffer with the initial capacity of 16.
StringBuffer(String str):	Creates a string buffer with the specified string.
StringBuffer(int capacity):	Creates an empty string buffer with the specified capacity as length.

Let us see below list of StringBuffer Method name:

Return Type	Method name	Description
synchronized StringBuffer	append(String s)	This method is used to append the specified string with this string.
synchronized StringBuffer	insert(int offset, String s)	This method is used to insert the specified string with this string at the specified position.
synchronized StringBuffer	replace(int startIndex, int endIndex, String str)	This method is used to replace the string from specified startIndex and endIndex.
synchronized StringBuffer	delete(int startIndex, int endIndex)	This method is used to delete the string from specified startIndex and endIndex.
synchronized StringBuffer	reverse()	This method is used to reverse the string.
int	capacity()	This method is used to return the current capacity.
char	charAt(int index)	This method is used to return the character at the specified position.
int	length()	This method is used to return the length of the string i.e. total number of characters.
String	substring(int beginIndex):	This method is used to return the substring from the specified beginIndex.
String	substring(int beginIndex, int endIndex):	This method is used to return the substring from the specified beginIndex and endIndex.

Example 1:

```
public class StringBufferClassDemo
{
        public static void main(String [] args)
        {
                StringBuffer sb=new StringBuffer("Hello ");
                sb.append("world");
                System.out.println(sb);                    //print Hello world

                StringBuffer sb2=new StringBuffer("Hello ");
                sb2.insert(1,"world");
                System.out.println(sb2);                   //print  Hworldello

                StringBuffer sb3=new StringBuffer("Hello");
                sb3.replace(1,3,"world");
                System.out.println(sb3);                   // print Hworldlo

                StringBuffer sb4=new StringBuffer("Hello");
                sb4.delete(1,3);
                System.out.println(sb4);                    // print Hlo

                StringBuffer sb5=new StringBuffer("Hello");
                sb5.reverse();
```

```
                        System.out.println(sb5);                        //print olleH

                        StringBuffer sb6=new StringBuffer();
                        System.out.println(sb6.capacity());             //print 16

                        StringBuffer sb7=new StringBuffer();
                        sb7.append("java is good programming language");
                        System.out.println(sb7.capacity());             //print 34
                        }
}
```

Output:

```
Hello world
Hworldello
Hworldlo
Hlo
olleH
16
34
```

2.5 Operation on String:

(A) Find out string length:

In java, we can find length of string using length () method. Length method always returns integer value.

Example:

```
publicclass stringoperation
{
            public static void main(String[] args)
            {
             String s1="Welcome";
            int len;
             len=s1.length();
             System.out.println("Length of string is ="+ len);
            }
}
```

Output:

```
Length of string is =7
```

(B) Compare two strings:

The equals () method used for compare two string objects. If this method return true then both string object contents are same. We can compare two string objects using compareTo () method compareTo () is quite similar to equals () method.

Example: using equals () method

```
publicclass stringoperation
{
            public staticvoid main(String[] args)
            {
```

```
            String s1="Welcome";
            String s2="Welcome";

if(s1.equals(s2))
{
           System.out.println("Both strings are same");
            }
else
{
            System.out.println("Both strings are not same");
}
}
}
```

Output:

```
Both strings are same
```

Example 2: using compareTo () method.

```
publicclass stringoperation
{
           publicstaticvoid main(String[] args)
           {
            String s1="Welcome";
            String s2="Welcome";

if(s1.compareTo(s2)==0)
{
            System.out.println("Both strings are same");
           }
else
{
           System.out.println("Both strings are not same");
}
}
}
```

Output:

```
Both strings are same
```

(C) Connect two strings:

Two or more strings putting together is called concatenation.in java concatenation is perform using "+" operator and concat () method.

Example:

```
publicclass stringoperation
{
     publicstaticvoid main(String[] args)
           {
            String s1="Welcome";
           String s2=" to java programming";
            String s3="";
```

```
        s3=s1.concat(s2);
        System.out.println("After connect two strings is =" +s3);
        }
}
```

Output:

```
After connect two strings is =Welcome to java programming
```

(D) Convert string to upper case:

In java, to change the lower case to upper case using toUpperrCase () method.

Example:

```
publicclass stringoperation
{
        publicstaticvoid main(String[] args)
        {
                String s1="welcome to java programming";
                String s2="";
                s2=s1.toUpperCase();
                System.out.println("After convert upper case  = " +s2);
        }
}
```

Output:

```
After convert upper case  =WELCOME TO JAVA PROGRAMMING
```

(E) Convert string to lower case:

In java, to change the Upper case to Lower case using toLowerCase () method.

Example:

```
publicclass stringoperation
{
        publicstaticvoid main(String[] args)
        {
                String s1="WELCOME TO JAVA PROGRAMMING";
                String s2="";
                s2=s1.toLowerCase();
                System.out.println("After convert Lower case  = " +s2);
        }
}
```

Output:

```
After convert Lower case  = welcome to java programming
```

(F) Search sub string:

We can extract substring from given string using **substring ()** method. This method has two form.

1. String substring(int startIndex)
2. String substring(int startindex, int endindex)

Example:

```
publicclass stringoperation
```

```
{
        publicstaticvoid main(String[] args)
        {
        String s1="WELCOME TO JAVA PROGRAMMING";
        System.out.println("substring from index 5:  "+ s1.substring(5));
        System.out.println("substring from index 11 to 14 :  "+ s1.substring(11, 15));
   }
}
```

Output:

```
substring from index 5:  ME TO JAVA PROGRAMMING
substring from index 11 to 14 :  JAVA
```

(G) String trim:

The trim () is return the string without leading as well as trailing white space. For example our string is like "   welcome   ". So here we have apply trim () on this string so we will get like "welcome".

Example:

```
publicclassstringoperation
{
        publicstaticvoid main(String[] args)
        {
                String s1="   WELCOME TO JAVA PROGRAMMING  ";
                String s2="   hello how are you  ";

                System.out.println("before: s1=" +s1);
                System.out.println("before: s2=" +s2);

                s2.trim();
                System.out.println("After trim: s1=" +s1.trim());
                System.out.println("after trim: s2=" +s2.trim());
        }
}
```

Output:

```
before: s1=   WELCOME TO JAVA PROGRAMMING
before: s2=   hello how are you
After trim: s1=WELCOME TO JAVA PROGRAMMING
after trim: s2=hello how are you
```

2.6 command Line Arguments:

The java command-line argument is an argument i.e. passed at the time of running the java program. The arguments passed from the console can be received in the java program and it can be used as an input. So, it provides a convenient way to check the behaviour of the program for the different values. You can pass **N** (1, 2, 3 and so on) numbers of arguments from the command prompt.

Here you have passed number in command line argument that number is always consider as a string.

Example 1:
We have receiving only one argument and printing it. you must pass at least one argument from the command prompt.

```
class DemoCommandLine
{
public static void main(String args[])
{
    System.out.println("Your first argument is: "+args[0]);
}
}
```

Output:

```
Your first argument is:10
```

Example 2: addition of two interger numbers which are passed as command line arguments
Here assumed that the command line arguments are 20 and 30. So the args[0] location takes 20 value and args[1] location takes 30 value. Those are considering as a strings so we need **parse** the value into integer you can see in below example.

```
publicclass SecondjavaClass
{
        publicstaticvoid main(String[] args)
        {
        int a, b, add;
         a=Integer.parseInt(args[0]);
         b=Integer.parseInt(args[1]);
         add=a+b;
         System.out.println("addition value is =" + add);
         }
}
```

Output:

```
addition value is=50
```

2.7 wrapper class:

Wrapper class is defined as a class in which a primitive value is wrapped up. These primitive wrapper classes are used to represent primitive data type values as objects.

For example, Integer wrapper class holds primitive ‘int’ data type value. Similarly, Float wrapper class holds ‘float’ primitive values, Character wrapper class holds a ‘char’ type value, and Boolean wrapper class holds also ‘boolean’ value. Wrapper classes are immutable.

See figure for illustration about wrapper class

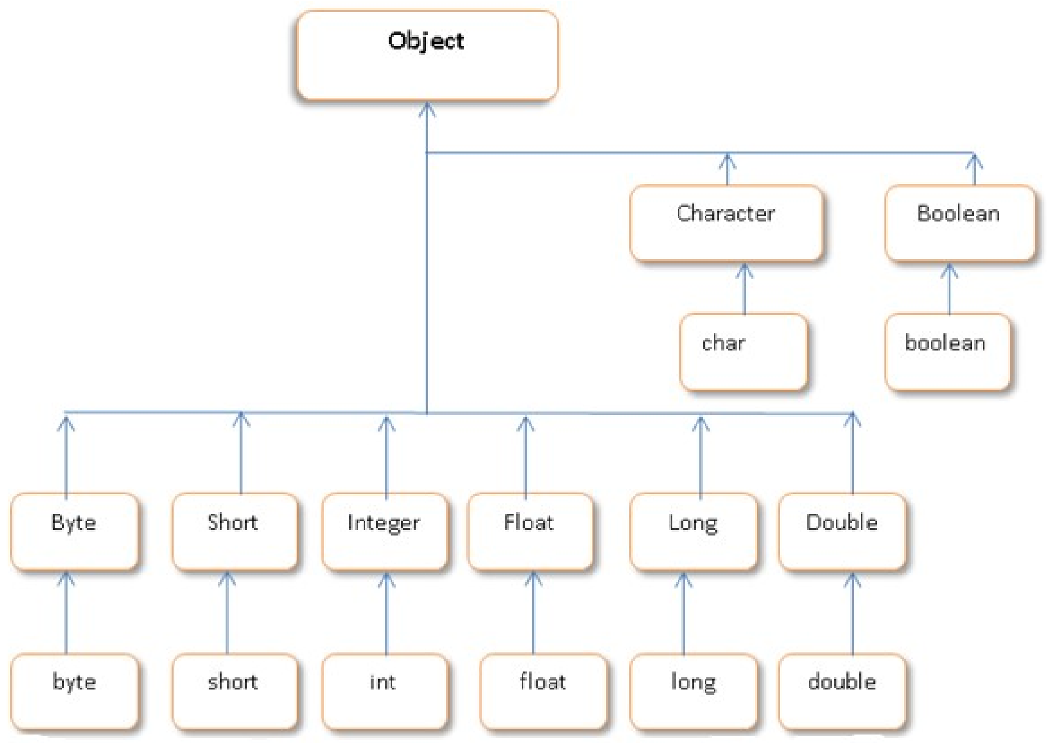


Figure. Illustration about wrapper class and object

List of wrapper classes in java

Wrapper class	**Primitive data type**
Character	char
Boolean	boolean
Byte	byte
Short	short
Integer	int
Float	float
Long	long
Double	double

Example:

```
publicclass FinalizedMethodDemo
{
publicstaticvoid main(String[] args)
  {
        // Primitive data type 'int'
        int i = 20;
        char ch='A';
        float f=12.12f;
        double d=15.11d;

        //  Integer Wrapper class instantiation
            Integer int_Obj = newInteger(i);
```

```
                Character ch_obj = newCharacter(ch);
                Float f_obj = newFloat(f);
                Double d_obj = newDouble(d);

// Unwrapping primitive data 'int' from wrapper object
int i1 = int_Obj.intValue();
char ch1 = ch_obj.charValue();
float f1 = f_obj.floatValue();
double d1 = d_obj.doubleValue();

              System.out.println("int value ="+i1);
              System.out.println("char value="+ch1);
              System.out.println("float value=="+f1);
              System.out.println("double value="+d1);
    }
}
```

Output:

```
int value =20
char value=A
float value==12.12
double value=15.11
```

2.8 Math class in java:

The class Math contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

Return type	**Method name**	**description**
static double	E	E base of the natural logarithms
static double	PI	Ratio of PI constant
static double	abs(double a)	Return absolute value of given arguments double a.
static float	abs(float a)	Return absolute value of given argument float a.
static long	abs(long a)	Return absolute value of given argument long a.
static double	acos(double a)	Return cosine value
static double	asin(double a)	Return sign value.
static double	atan(double a)	Return tangent value
static double	cbrt(double a)	Return the cube root of double value
static double	ceil(double a)	Return smallest double value
static double	cos(double a)	Return trigonometric cosine of a
static double	cosh(double x)	Return the hyperbolic cosine of double value
static double	exp(double a)	Return e raised to the power of double value
static double	floor(double a)	Return the largest double value that is less than or equal to the argument a
static double	log (double a)	Return natural logarithm of a double value base e
static double	log10(double a)	Return base 10 logarithm of double value
static double	max(double a ,double b)	Return largest of two double values.
static float	max(float a ,float b)	Return largest of two float values.
static int	max(int a ,int b)	Return largest of two int values.

static long	max(long a ,long b)	Return largest of two long values.
static double	min(double a ,double b)	Return smallest of two double values.
static float	min(float a ,float b)	Return smallest of two float values.
static int	min(int a ,int b)	Return smallest of two int values.
static long	min(long a ,long b)	Return smallest of two long values.
static double	pow(double a, double b)	Return value of the first arguments raised to the power of the second arguments.
static double	random()	Return double value within 0.0 to 1.0
static double	rint(double a)	Return double value that is closest in value to the argument and is equal to a mathematical integer
static double	round(double a)	Return the closest long to the argument, with ties rounding up
static float	round(float a)	Return the closest int to the argument with ties rounding up.
static double	sin(double a)	Return the trigonometric sine of an angle
static double	sinh(double x)	Return the hyperbolic sine of a double value
static double	sqrt(double a)	Return correctly rounded positive square root of a double value.
static double	tan(double a)	Return trigonometric tangent of an angle
static double	tanh(double x)	Return the hyperbolic tangent of double value.

Example:

```
package ABC;
publicclass DemoMathClass
{
        publicstaticvoid main(String[] args)
        {
                double a = 10.7;
                double b=5.5;
                System.out.println("E="+Math.E);
                System.out.println("PI="+Math.PI);
                System.out.println("abs(a)="+Math.abs(a));
                System.out.println("acos(a)="+Math.acos(a));
                System.out.println("asin(a)="+Math.asin(a));
                System.out.println("atan(a)="+Math.atan(a));
                System.out.println("cbrt(a)="+Math.cbrt(a));
                System.out.println("ceil(a)="+Math.ceil(a));
                System.out.println("cos(a)=" +Math.cos(a));
                System.out.println("cosh(a)="+Math.cosh(a));
                System.out.println("exp(a)="+Math.exp(a));
                System.out.println("floor(a)="+Math.floor(a));
                System.out.println("log(a)="+Math.log(a));
                System.out.println("log10(a)="+Math.log10(a));
                System.out.println("max(a,b)="+Math.max(a,b));
                System.out.println("min(a,b)="+Math.min(a,b));
                System.out.println("pow(a,b)="+Math.pow(a, b));
                System.out.println("random()="+Math.random());
                System.out.println("rint(a)="+Math.rint(a));
                System.out.println("round(a)="+Math.round(a));
```

```
                    System.out.println("sin(a)"+Math.sin(a));
                    System.out.println("sinh(a)"+Math.sinh(a));
                    System.out.println("sqrt(a)"+Math.sqrt(a));
                    System.out.println("tan(a)"+Math.tan(a));
                    System.out.println("tanh(a)"+Math.tanh(a));
    }
}
```

Output:

```
E=2.718281828459045
PI=3.141592653589793
abs(a)=10.7
acos(a)=NaN
asin(a)=NaN
atan(a)=1.4776090650260174
cbrt(a)=2.2035754532216254
ceil(a)=11.0
cos(a)=-0.2912892817213455
cosh(a)=22177.92757642139
exp(a)=44355.85513029784
floor(a)=10.0
log(a)=2.3702437414678603
log10(a)=1.0293837776852097
max(a,b)=10.7
min(a,b)=5.5
pow(a,b)=458786.6554619732
random()=0.34806010016308064
rint(a)=11.0
round(a)=11
sin(a)-0.9566350162701879
sinh(a)22177.92755387645
sqrt(a)3.271085446759225
tan(a)3.2841408053775507
tanh(a)0.9999999989834516
```

2.9 String class vs StringBuffer class [summer 2014, winter 2013, winter 2012]

String Class	**StringBuffer Class**
String is immutable (once created cannot be changed)object	StringBuffer is mutable means one can change the value of the object
The object created as a String is stored in the Constant String Pool.	The object created through StringBuffer is stored in the heap
String cannot be used by two threads simultaneously.	each method in StringBuffer is synchronized that is StringBuffer is thread safe .
String once assigned cannot be changed.	StringBuffer value can be changed , it means it

	can be assigned to the new value
Example: String demo = “hello”; // The above object is stored in constant string pool and its value cannot be modified.	StringBuffer demo1 = new StringBuffer ("Hello"); // The above object stored in heap and its value can be changed .

Difference Between String , StringBuilder And StringBuffer Classes

String

String is immutable (once created can not be changed)object . The object created as a String is stored in the Constant String Pool.
Every immutable object in Java is thread safe ,that implies String is also thread safe . String can not be used by two threads simultaneously.
String once assigned can not be changed.

StringBuffer

StringBuffer is mutable means one can change the value of the object . The object created through StringBuffer is stored in the heap. StringBuffer has the same methods as the StringBuilder , but each method in StringBuffer is synchronized that is StringBuffer is thread safe .

Due to this it does not allow two threads to simultaneously access the same method . Each method can be accessed by one thread at a time .

But being thread safe has disadvantages too as the performance of the StringBuffer hits due to thread safe property . Thus StringBuilder is faster than the StringBuffer when calling the same methods of each class.
String Buffer can be converted to the string by using
toString() method.

StringBuffer demo1 = new StringBuffer("Hello") ;
// The above object stored in heap and its value can be changed .
demo1=new StringBuffer("Bye");
// Above statement is right as it modifies the value which is allowed in the StringBuffer

StringBuilder

StringBuilder is same as the StringBuffer , that is it stores the object in heap and it can also be modified . The main difference between the StringBuffer and StringBuilder is
that StringBuilder is also not thread safe.
StringBuilder is fast as it is not thread safe .

StringBuilder demo2= new StringBuilder("Hello");
// The above object too is stored in the heap and its value can be modified
demo2=new StringBuilder("Bye");
// Above statement is right as it modifies the value which is allowed in the StringBuilder

2.10 Ragged array [Nov dec 2011]

Ragged array: is an array with more than one dimension each dimension has different size. There is no requirement that all rows in a two-dimensional array have the same length—an array with rows of non-uniform length is known as a *ragged array*.

```
public class  Rarray
{
        public static void  main(String s[])
        {
        int a[][]=newint[3][];
        a[0]= newint[3];
        a[1]= new int[2];
        a[2]= new int[1];

        a[0][0]=1;
        a[0][1]=2;
        a[0][2]=3;

        a[1][0]=4;
        a[1][1]=5;

        a[2][0]=6;

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

Output:

```
1
2
3
4
5
6
```

2.11GTU Question paper Program:

Program 1:

Ragged array:[Nov dec 2011]
See in 2.10 section

2.12 Asked Question in GTU Papers:

Q.1 state whether the following statements are true or false. [Dec 2011]

(i) The statements in an array must be of primitive data types: TRUE

(ii) a method can change the length of an array passes as a parameter: TRUE

Q.2 answer of following question: Ragged array.[Dec 2011]

Q.3 Different between String Class and StringBuffer class.[Summer 2014, winter 2013, winter 2012]