**Primitive Classes**

As Java is an Object Oriented language sometimes we may face some situations where we have to use objects instead of primitive data types, so Java provides wrapper classes for every primitive data type. In this topics we will see how to create objects for Byte, Short, Integer, Long and also their methods.

* The abstract class **Number** is the superclass of classes BigDecimal, BigInteger, Byte, Double, Float, Integer, Long and Short. Subclasses of Number must provide methods to convert the represented numeric value to byte, double, float, int, long and short.

**Double**

Double wrapper class has two methods isInfinite and isNaN. isInfinite method checks whether a value is infinite and isNaN checks whether a value is NaN(Not a Number).

class isInfiniteandisNanDemo  
{  
    public static void main(String arg[])  
    {  
        Double d1 = new Double(2.0 / 0.0);  
        Double d2 = new Double(0 / 0.0);  
        System.out.println(d1 + ": " + d1.isInfinite());  
        System.out.println(d2 + ": " + d2.isNaN());      
    }  
}

**Creating Objects for primitive data types:**The following are the constructors used in creating objects for primitive data type.  
For Byte :  
Byte(byte num)  
Byte(String str) throws NumberFormatException  
  
For Short :  
Short(short num)  
Short(String str) throws NumberFormatException  
  
For Long :  
Long(long numb)  
Long(String str) throws NumberFormtatException

class ConstructorsDemo  
{  
    public static void main(String[] args)  
    {  
        Byte b = new Byte((byte) 5); // LINE A  
        Byte b\_str = new Byte("5" + 1);  
        System.out.println("Value of b : " + b + " Value of b\_str " + b\_str);  
        Short s = new Short((short) 10); // LINE B  
        Short s\_str = new Short("10" + 2);  
        System.out.println("Value of s : " + s + " Value of b\_str " + s\_str);  
        Integer i = new Integer(15); // LINE C  
        Integer i\_str = new Integer("15" + 3);  
        System.out.println("Value of i : " + i + " Value of b\_str " + i\_str);  
        Long l = new Long(20); // LINE D  
        Long l\_str = new Long("20" + 4);  
        System.out.println("Value of i : " + l + " Value of b\_str " + l\_str);  
    }  
}

**Converting Numbers to and from Strings**

**Converting Numbers to String:**There are many ways of converting numbers into string object,

* By concatenating number with empty string i.e. " + " operator
* The valueOf() method converts the given data type to String.
* Each of the Number subclass includes a class method, toString, that will convert its primitive type in to a String.

class NumberstoStringTest  
{  
    public static void main(String arg[])  
    {  
        int i = 32;  
        String s1 = "" + i; // LINE A  
        System.out.println("s1 = " + s1);  
          
        double d = 876.54;  
        String s2 = String.valueOf(d); // LINE B  
        System.out.println("s2 = " + s2);  
          
        float f = 156.8f;  
        String s3 = Float.toString(f); // LINE C  
        System.out.println("s3 = " + s3);  
          
        String s4 = "" + i + d; // LINE D  
          
        System.out.println("s4 = " + s4);  
        int n = s4.indexOf('.');  
        System.out.println(n + " Digits before decimal point of s4 string.");  
        System.out.println(s4.length() - n - 1  
                + " Digits after decimal point of s4 string.");      
    }

**Converting Strings to Number:**The two ways of converting Strings into specified data type are,

* The valueOf() method converts the String to specified data type.
* The parsing methods are widely used to convert strings into numbers.

class StringstoNumberTest  
{  
    public static void main(String arg[])  
    {  
        String s1 = "25";  
        int i1 = Integer.valueOf(s1); // LINE E  
        int i2 = Integer.parseInt(s1); // LINE F  
        System.out.println(i1 + " " + i2);  
          
        float f1 = Float.valueOf(s1);  
        float f2 = Float.parseFloat(s1);  
        System.out.println(f1 + " " + f2);  
          
        double d1 = Double.valueOf(s1);  
        double d2 = Double.parseDouble(s1);  
        System.out.println(d1 + " " + d2);  
          
        System.out.print("The sum of all numbers is : ");  
        System.out.println(i1 + i2 + f1 + f2 + d1 + d2); // LINE G  
          
        String s2 = "ABCD";  
        char c1[] = new char[s2.length()];  
        for (int i = 0; i < s2.length(); i++) {  
            c1[i] = Character.valueOf(s2.charAt(i)); // LINE H  
        }  
        for (int j = 0; j < c1.length; j++) {  
            System.out.print(c1[j] + " ");  
        }      
    }  
}

**Character**

Character class wraps a value of the primitive type  char in an object. An object of type Character contains a single field whose type is char.

char ch = 'a'; // Char

char[] charArray = { 'a', 'b', 'c', 'd', 'e' }; // Array of chars

class CharacterTest  
{  
    public static void main(String arg[])  
    {  
        System.out.println("Checks for letter : " + Character.isLetter('M'));  
        System.out.println("Checks for digit : " + Character.isDigit('5'));  
        System.out.println("Checks for white space : " + Character.isWhitespace(' '));  
        System.out.println("Checks for uppercase : " + Character.isUpperCase('C'));  
        System.out.println("Checks for lowercase : " + Character.isLowerCase('c'));  
        System.out.println("Converts to uppercase : " + Character.toUpperCase('m'));  
        System.out.println("Converts to lowercase : " + Character.toLowerCase('Z'));  
        System.out.println("Converts to string : " + Character.toString('w'));      
    }  
}

**Boolean:**

Boolean class wraps a value of the primitive type boolean in an object. An object of type Boolean contains a single field whose type is boolean.

class BooleanTest  
{  
    public static void main(String arg[])  
    {  
        Boolean a = new Boolean(true);  
        Boolean b = new Boolean(false);  
        Boolean c = new Boolean(false);  
          
        System.out.println("Boolean value of a : " + a.booleanValue());  
        System.out.println("Comparison of b and c using compare : " + Boolean.compare(b, c));  
        System.out.println("CompareTo of a and c : " + a.compareTo(c)); // LINE A  
        System.out.println("Equals of b and c : " + b.equals(c));  
          
        System.setProperty("Isboolean", "true"); // LINE B  
        System.out.println("Get Boolean from Isboolean : " + Boolean.getBoolean("Isboolean"));  
                System.out.println("ParseBoolean converts String toboolean : " + Boolean.parseBoolean("true"));  
        System.out.println("Boolean is converted into String : " + b.toString());  
        System.out.println("Boolean is converted into String : " + Boolean.toString(a));  
        System.out.println("Converts Boolean into Boolean : " + Boolean.valueOf(a));  
        System.out.println("Converts String into Boolean : " + Boolean.valueOf("true"));      
    }  
}

**System**

**currentTimeMillis()**

The currentTimeMillis method returns the current time in terms of milliseconds since midnight, January 1, 1970. The currentTimeMillis method is to time how long various parts of your program take to execute .

import java.lang.\*;  
  
class Elapsed  
{  
    public static void main(String arg[])  
    {  
        long start, end;  
        System.out.println("Timing a for loop from 0 to 1000");  
        // time a for loop from o to 1000  
        start = System.currentTimeMillis(); // get time  
        System.out.println("Starting time in milliseconds : " + start);  
          
        for (int i = 0; i < 1000; i++)  // LINE A  
        {  
              
            for(int j =0; j < 2000; j++)  
            {  
                for( int k =0; k < 3000; k++)  
                {  
                      
                }  
            }  
        }  
        end = System.currentTimeMillis(); // get ending time  
        System.out.println("Ending time in milliseconds : " + end);  
        System.out.println("Elapsed time: " + (end - start)+ " ms"); // LINE B      
    }  
}

**Math Class**

Methods in Math class are used to perform basic operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

import java.lang.Math;                      
  
class MathDemo  
{  
    public static void main(String arg[])  
    {  
        int num = 9;  
        if (Math.sqrt(num) \* Math.sqrt(num) == num) // LINE A  
            System.out.println(num + " is a perfect square");  
        else  
            System.out.println(num + " is not a perfect square");  
        num = -2;  
        System.out.println("Value of num : " + Math.abs(num)); // LINE B  
        System.out.println("value when Math.ceil is used : " + Math.ceil(Math.PI)); // LINE C  
        System.out.println("Value when Math.floor is used : " + Math.floor(Math.PI)); // LINE D  
        num = (int) (Math.random() \* 10); // LINE E  
        System.out.println("Random number between 0 and 10 : " + num);      
    }  
}

class NumberMethodTest2  
{  
    public static void main(String arg[])  
    {  
        Float b = 5.6f;  
        System.out.println("Converts to Positive Number : " + Math.abs(-7));  
        System.out.println("Smallest integer greater than b : " + Math.ceil(b));  
        System.out.println("Greatest integer less than b : " + Math.floor(b));  
        System.out.println("Closest integer as double : " + Math.rint(b));  
        System.out.println("Closest integer : " + Math.round(b));  
        System.out.println("Minimum of two numbers : " + Math.min(12.3, 9.6));  
        System.out.println("Maximum of two numbers : " + Math.max(8, 6));  
        System.out.println("e power 100 : " + Math.exp(100));  
        System.out.println("Log value : " + Math.log(2.7183));  
        System.out.println("2 Power 5 : " + Math.pow(2, 5));  
        System.out.println("Square root : " + Math.sqrt(81));      
    }  
}

class NumberMethodTest3  
{  
    public static void main(String arg[])  
    {  
        Double degree = 30.0;  
              
        System.out.println("sin(30) : " + Math.sin(Math.toRadians(degree)));  
        System.out.println("cos(30) : " + Math.cos(Math.toRadians(degree)));  
        System.out.println("tan(30) : " + Math.tan(Math.toRadians(degree)));  
        System.out.println("sec(30) : " + Math.asin(Math.toRadians(degree)));  
        System.out.println("cosec(30) : " + Math.acos(Math.toRadians(degree)));  
        System.out.println("cot(30) : " + Math.atan(Math.toRadians(degree)));  
        System.out.println("Theta of rectangle co-ordinate : " + Math.atan2(45,30));      
    }  
}

**Enum in Java**

Enumerations, in general, are a set of **named constants.** They have been in other programming languages from the beginning (e.g., C, C++). Prior to Java 1.5, Java lacked language support for enumerations. This concept was added in 1.5v and the enum in Java is more powerful than many other languages.

enum Move  
{  
    LEFT,  
    RIGHT,  
    UP,  
    DOWN;    // ; is optional  
}

enum Move  
{  
    public static final Move LEFT = new Move();  
    public static final Move RIGHT = new Move();  
    public static final Move UP = new Move();  
    public static final Move DOWN = new Move();  
}

Enums are often used in switch statements to check for corresponding values:

enum Level {

LOW,

MEDIUM,

HIGH

}

public class Main {

public static void main(String[] args) {

Level myVar = Level.MEDIUM;

switch(myVar) {

case LOW:

System.out.println("Low level");

break;

case MEDIUM:

System.out.println("Medium level");

break;

case HIGH:

System.out.println("High level");

break;

}

}

}

**Java Comparable Interface**

Comparable interface in Java is used to compare two objects in some meaningful manner.

Comparable is generic and is declared as shown below.

public interface Comparable<T>

Here, T represents the type of objects being compared. If any class implement Comparable interface in Java then collection of that object either List or Array can be sorted automatically by using Collections.sort() or Arrays.sort() method and object will be sorted based on there natural order defined by CompareTo method

**Note:**

We will see example when we work on collections