

Java.Lang.Double class in Java

Difficulty Level : Basic Last Updated : 03 Aug, 2021

Double class is a wrapper class for the primitive type double which contains several methods to effectively deal with a double value like converting it to a string representation, and vice-versa. An object of Double class can hold a single double value. There are mainly two constructors to initialize a Double object-

- **Double(double b):** Creates a Double object initialized with the value provided.

Syntax : `public Double(Double d)`

Parameters :

`d` : value with which to initialize

- **Double(String s):** Creates a Double object initialized with the parsed double value provided by string representation. Default radix is taken to be 10.

Syntax : `public Double(String s)`

`throws NumberFormatException`

Parameters :

`s` : string representation of the byte value

Throws :

`NumberFormatException` : If the string provided does not represent any double



Methods:

1. **toString():** Returns the string corresponding to the double value.

Syntax : `public String toString(double b)`

Parameters :

`b` : double value for which string representation required.

2. **valueOf():** returns the Double object initialized with the value provided.

Syntax : `public static Double valueOf(double b)`

Parameters :

`b` : a double value

Another overloaded function `valueOf(String val)` which provides function similar to `new Double(Double.parseDouble(val,10))`


Syntax : `public static Double valueOf(String s)`
throws `NumberFormatException`

Parameters :

`s` : a `String` object to be parsed as double

Throws :

`NumberFormatException` : if `String` cannot be parsed to a double value in given



3. `parseDouble()`: returns double value by parsing the string. Differs from `valueOf()` as it returns a primitive double value and `valueOf()` return `Double` object.


Syntax : `public static double parseDouble(String val)`
throws `NumberFormatException`

Parameters :

`val` : `String` representation of double

Throws :

`NumberFormatException` : if `String` cannot be parsed to a double value in given



4. `byteValue()`: returns a byte value corresponding to this `Double` Object.

Syntax : `public byte byteValue()`

5. `shortValue()`: returns a short value corresponding to this `Double` Object.

Syntax : `public short shortValue()`

6. `intValue()`: returns a int value corresponding to this `Double` Object.

Syntax : `public int intValue()`

7. `longValue()`: returns a long value corresponding to this `Double` Object.

Syntax : `public long longValue()`

8. `doubleValue()`: returns a double value corresponding to this `Double` Object.

Syntax : `public double doubleValue()`

9. floatValue(): returns a float value corresponding to this Double Object.

Syntax : `public float floatValue()`

10. hashCode(): returns the hashcode corresponding to this Double Object.

Syntax : `public int hashCode()`

11. isNaN(): returns true if the double object in consideration is not a number, otherwise false.

Syntax : `public boolean isNaN()`

Another static method `isNaN(double val)` can be used if we don't need any object of double to be created. It provides similar functionality as the above version.

Syntax : `public static boolean isNaN(double val)`

Parameters :

`val` : double value to check for

12. isInfinite(): returns true if the double object in consideration is very large, otherwise false. Specifically, any number beyond `0x7ff0000000000000L` on the positive side and below `0xfff0000000000000L` on negative side is the infinity values.

Syntax : `public boolean isInfinite()`

Another static method `isInfinite(double val)` can be used if we dont need any object of double to be created. It provides similar functionality as the above version.

Syntax : `public static boolean isInfinte(double val)`

Parameters :

`val` : double value to check for

13. toHexString(): Returns the hexadecimal representation of the argument double value.

Syntax : `public static String toHexString(double val)`

Parameters :

`val` : double value to be represented as hex string

14. doubleToLongBits(): returns the IEEE 754 floating-point “double format” bit layout of the given double argument.

Syntax : `public static long doubleToLongBits(double val)`

Parameters :

`val` : double value to convert

15. doubleToRawLongBits(): returns the IEEE 754 floating-point “double format” bit layout of the given double argument. It differs from the previous method as it preserves the Nan values.

Syntax : `public static long doubleToRawLongBits(double val)`

Parameters :

`val` : double value to convert

16. LongBitsToDouble(): Returns the double value corresponding to the long bit pattern of the argument. It does reverse work of the previous two methods.

Syntax : `public static double LongBitsToDouble(long b)`

Parameters :

`b` : long bit pattern

17. equals(): Used to compare the equality of two Double objects. This method returns true if both the objects contain same double value. Should be used only if checking for equality. In all other cases, compareTo method should be preferred.

Syntax : `public boolean equals(Object obj)`

Parameters :

`obj` : object to compare with

18. compareTo(): Used to compare two Double objects for numerical equality. This should be used when comparing two Double values for numerical equality as it would differentiate between less and greater values. Returns a value less than 0, 0, value greater than 0 for less than, equal to and greater than.

Syntax : `public int compareTo(Double b)`

Parameters :

`b` : Double object to compare with

19. compare(): Used to compare two primitive double values for numerical equality. As it is a static method therefore it can be used without creating any object of Double.

Syntax : `public static int compare(double x,double y)`

Parameters :

x : double value

y : another double value

```
// Java program to illustrate
// various Double class methods
// of java.lang class
public class Double_test
{

    public static void main(String[] args)
    {

        double b = 55.05;
        String bb = "45";

        // Construct two Double objects
        Double x = new Double(b);
        Double y = new Double(bb);

        // toString()
        System.out.println("toString(b) = " + Double.toString(b));

        // valueOf()
        // return Double object
        Double z = Double.valueOf(b);
        System.out.println("valueOf(b) = " + z);
        z = Double.valueOf(bb);
        System.out.println("ValueOf(bb) = " + z);

        // parseDouble()
        // return primitive double value
        double zz = Double.parseDouble(bb);
        System.out.println("parseDouble(bb) = " + zz);

        System.out.println("bytevalue(x) = " + x.byteValue());
        System.out.println("shortvalue(x) = " + x.shortValue());
        System.out.println("intvalue(x) = " + x.intValue());
        System.out.println("longvalue(x) = " + x.longValue());
        System.out.println("doublevalue(x) = " + x.doubleValue());
        System.out.println("floatvalue(x) = " + x.floatValue());

        int hash = x.hashCode();
        System.out.println("hashCode(x) = " + hash);
```

```

    boolean eq = x.equals(y);
    System.out.println("x.equals(y) = " + eq);

    int e = Double.compare(x, y);
    System.out.println("compare(x,y) = " + e);

    int f = x.compareTo(y);
    System.out.println("x.compareTo(y) = " + f);

    Double d = Double.valueOf("1010.54789654123654");
    System.out.println("isNaN(d) = " + d.isNaN());

    System.out.println("Double.isNaN(45.12452) = " + Double.isNaN(45.12452));

    // Double.POSITIVE_INFINITY stores
    // the positive infinite value
    d = Double.valueOf(Double.POSITIVE_INFINITY + 1);
    System.out.println("Double.isInfinite(d) = " +
        Double.isInfinite(d.doubleValue()));

    double dd = 10245.21452;
    System.out.println("Double.toString(dd) = " + Double.toHexString(dd));

    long double_to_long = Double.doubleToLongBits(dd);
    System.out.println("Double.doubleToLongBits(dd) = " + double_to_long);

    double long_to_double = Double.longBitsToDouble(double_to_long);
    System.out.println("Double.LongBitsToDouble(double_to_long) = " +
        long_to_double);
}
}

```

Output :

```

toString(b) = 55.05
valueOf(b) = 55.05
ValueOf(bb) = 45.0
parseDouble(bb) = 45.0
bytevalue(x) = 55
shortvalue(x) = 55
intvalue(x) = 55
longvalue(x) = 55
doublevalue(x) = 55.05
floatvalue(x) = 55.05

```

```
hashCode(x) = 640540672
x.equals(y) = false
compare(x,y) = 1
x.compareTo(y) = 1
isNaN(d) = false
Double.isNaN(45.12452) = false
Double.isInfinite(d) = true
Double.toString(dd) = 0x1.4029b7564302bp13
Double.doubleToLongBits(dd) = 4666857980575363115
Double.LongBitsToDouble(double_to_long) = 10245.21452
```

References: [Official Java Documentation](#)

This article is contributed by **Rishabh Mahrsee**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.