

# Java.Lang.Byte class in Java

Difficulty Level : Basic Last Updated : 29 Oct, 2021

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

- **Byte(byte b):** Creates a Byte object initialized with the value provided.

**Syntax:** `public Byte(byte b)`

**Parameters :**

`b` : value with which to initialize

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

**Syntax :** `public Byte(String s)`

`throws NumberFormatException`

**Parameters :**

`s` : string representation of the byte value

**Throws :**

`NumberFormatException` : If the string provided does not represent any byte va

- 

## Fields in Byte class:

1. **static int BYTES:** The number of bytes used to represent a byte value in two's complement binary form.

2. **static byte MAX\_VALUE**: A constant holding the maximum value a byte can have, 27-1.
3. **static byte MIN\_VALUE**: A constant holding the minimum value a byte can have, -27.
4. **static int SIZE**: The number of bits used to represent a byte value in two's complement binary form.
5. **static Class<Byte> TYPE**: The Class instance representing the primitive type byte.

## Methods:

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

Syntax : `public String toString(byte b)`

Parameters :

b : byte value for which string representation required.

1. **valueOf()** : returns the Byte object initialized with the value provided.

Syntax : `public static Byte valueOf(byte b)`

Parameters :

b : a byte value

1. Another overloaded function `valueOf(String val,int radix)` which provides function similar to `new Byte(Byte.parseByte(val,radix))`

Syntax : `public static Byte valueOf(String val, int radix)`

throws `NumberFormatException`

Parameters :

val : String to be parsed into byte value

radix : radix to be used while parsing

Throws :

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

- 
1. Another overloaded function `valueOf(String val)` which provides function similar to `new Byte(Byte.parseByte(val,10))`

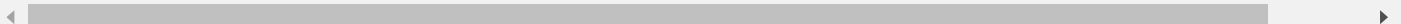
**Syntax :** `public static Byte valueOf(String s)`  
throws `NumberFormatException`

**Parameters :**

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

**Throws :**

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



1. **`parseByte()`** : returns byte value by parsing the string in radix provided. Differs from `valueOf()` as it returns a primitive byte value and `valueOf()` return `Byte` object.

**Syntax :** `public static byte parseByte(String val, int radix)`  
throws `NumberFormatException`


**Parameters :**

`val` : `String` representation of byte

`radix` : radix to be used while parsing

**Throws :**

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



1. Another overloaded method containing only `String` as a parameter, radix is by default set to 10.

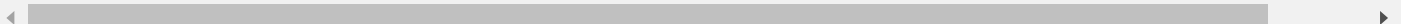
**Syntax :** `public static byte parseByte(String val)`  
throws `NumberFormatException`

**Parameters :**

`val` : `String` representation of byte

**Throws :**

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



1. **decode()** : returns a Byte object holding the decoded value of string provided. String provided must be of the following form else NumberFormatException will be thrown-

Decimal- (Sign)Decimal\_Number

Hex- (Sign)"0x"Hex\_Digits

Hex- (Sign)"0X"Hex\_Digits

Octal- (Sign)"0"Octal\_Digits

**Syntax** : public static Byte decode(String s)  
throws NumberFormatException

**Parameters** :

s : encoded string to be parsed into byte val

**Throws** :

NumberFormatException : If the string cannot be decoded into a byte value

1. **byteValue()** : returns a byte value corresponding to this Byte Object.

**Syntax** : public byte byteValue()

1. **shortValue()** : returns a short value corresponding to this Byte Object.

**Syntax** : public short shortValue()

1. **intValue()** : returns a int value corresponding to this Byte Object.

**Syntax** : public int intValue()

1. **longValue()** : returns a long value corresponding to this Byte Object.

**Syntax** : public long longValue()

1. **doubleValue()** : returns a double value corresponding to this Byte Object.

**Syntax :** `public double doubleValue()`

1. **floatValue()** : returns a float value corresponding to this Byte Object.

**Syntax :** `public float floatValue()`

1. **hashCode()** : returns the hashCode corresponding to this Byte Object.

**Syntax :** `public int hashCode()`

1. **equals()** : Used to compare the equality of two Byte objects. This methods returns true if both the objects contains same byte 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

1. **compareTo()** : Used to compare two Byte objects for numerical equality. This should be used when comparing two Byte 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(Byte b)`

**Parameters :**

b : Byte object to compare with

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

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

**Parameters :**

x : byte value  
y : another byte value

---

```
// Java program to illustrate
// various methods of Byte class
public class Byte_test
{
    public static void main(String[] args)
    {

        byte b = 55;
        String bb = "45";

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

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

        // valueOf()
        // return Byte object
        Byte z = Byte.valueOf(b);
        System.out.println("valueOf(b) = " + z);
        z = Byte.valueOf(bb);
        System.out.println("ValueOf(bb) = " + z);
        z = Byte.valueOf(bb, 6);
        System.out.println("ValueOf(bb,6) = " + z);

        // parseByte()
        // return primitive byte value
        byte zz = Byte.parseByte(bb);
        System.out.println("parseByte(bb) = " + zz);
        zz = Byte.parseByte(bb, 6);
        System.out.println("parseByte(bb,6) = " + zz);

        //decode()
        String decimal = "45";
        String octal = "005";
        String hex = "0x0f";

        Byte dec=Byte.decode(decimal);
        System.out.println("decode(45) = " + dec);
        dec=Byte.decode(octal);
        System.out.println("decode(005) = " + dec);
        dec=Byte.decode(hex);
```

```

        System.out.println("decode(0x0f) = " + dec);

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

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

```

## Output:

```

toString(b) = 55
valueOf(b) = 55
ValueOf(bb) = 45
ValueOf(bb,6) = 29
parseByte(bb) = 45
parseByte(bb,6) = 29
decode(45) = 45
decode(005) = 5
decode(0x0f) = 15
bytevalue(x) = 55
shortvalue(x) = 55
intvalue(x) = 55
longvalue(x) = 55
doublevalue(x) = 55.0
floatvalue(x) = 55.0
hashCode(x) = 55

```

```
x.equals(y) = false  
compare(x,y) = 10  
x.compareTo(y) = 10
```

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](https://write.geeksforgeeks.org) or mail your article to [review-team@geeksforgeeks.org](mailto: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.