Java.Lang.Short class in Java

Difficulty Level : Basic Last Updated : 29 Jul, 2021

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

- **Short(short b):** Creates a Short object initialized with the value provided.

```
Syntax : public Short(short b)
Parameters :
b : value with which to initialize
```

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

```
Syntax : public Short(String s) 
                    throws NumberFormatException
Parameters :
s : string representation of the short value 
Throws :
NumberFormatException : If the string provided does not represent any short v
```

-

Methods:

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

```
Syntax : public String toString(short b)
Parameters :
b : short value for which string representation required.
```

1. **valueOf() :** returns the Short object initialiszed with the value provided.

```
Syntax : public static Short valueOf(short b)
Parameters :
b : a short value
```

1. Another overloaded function valueOf(String val,int radix) which provides function similar to new Short(Short.parseShort(val,radix))

```
Syntax : public static Short valueOf(String val, int radix)
            throws NumberFormatException
Parameters :
val : String to be parsed into short value
radix : radix to be used while parsing
Throws :
NumberFormatException : if String cannot be parsed to a short value in given
```

1. Another overloaded function valueOf(String val) which provides function similar to new Short(Short.parseShort(val,10))

```
Syntax : public static Short valueOf(String s)
           throws NumberFormatException
Parameters :
s : a String object to be parsed as short
Throws :
NumberFormatException : if String cannot be parsed to a short value in given
```

1. **parseShort() :** returns short value by parsing the string in radix provided. Differs from valueOf() as it returns a primitive short value and valueOf() return Short object.

```
Syntax : public static short parseShort(String val, int radix)
             throws NumberFormatException
Parameters :
val : String representation of short
radix : radix to be used while parsing
Throws :
NumberFormatException : if String cannot be parsed to a short value in given
```

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

```
Syntax : public static short parseShort(String val)
             throws NumberFormatException
Parameters :
val : String representation of short
Throws :
NumberFormatException : if String cannot be parsed to a short value in given
```

1. **decode() :** returns a Short 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 Short decode(String s)
             throws NumberFormatException
Parameters :
s : encoded string to be parsed into short val
Throws :
NumberFormatException : If the string cannot be decoded into a short value
```

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

```
Syntax : public byte byteValue()
```

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

```
Syntax : public short shortValue()
```

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

```
Syntax : public int intValue()
```

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

```
Syntax : public long longValue()
```

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

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

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

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

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

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

1. **equals() :** Used to compare the equality of two Short objects. This methods returns true if both the objects contains same short 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 Short objects for numerical equality. This should be used when comparing two Short 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(Short b)
Parameters :
b : Short object to compare with
```

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

```
Syntax : public static int compare(short x,short y)
Parameters :
x : short value
y : another short value
```

1. **reverseBytes() :** returns a primitive short value reversing the the order of bits in two's complement form of the given short value.

```
Syntax : public static short reverseBytes(short val)
Parameters :
val : short value whose bits to reverse in order.
```

1.

```java
// Java program to illustrate
// various methods of Short class
public class Short_test
{

    public static void main(String[] args)
    {

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

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

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

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

        // parseShort()
        // return primitive short value
        short zz = Short.parseShort(bb);
        System.out.println("parseShort(bb) = " + zz);
        zz = Short.parseShort(bb, 6);
        System.out.println("parseShort(bb,6) = " + zz);

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

        Short dec = Short.decode(decimal);
        System.out.println("decode(45) = " + dec);
        dec = Short.decode(octal);
        System.out.println("decode(005) = " + dec);
        dec = Short.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 = Short.compare(x, y);
        System.out.println("compare(x,y) = " + e);

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


        short to_rev = 45;
        System.out.println("Short.reverseBytes(to_rev) = " + Short.reverseByte
    }
}
```

Output :

```
toString(b) = 55
valueOf(b) = 55
ValueOf(bb) = 45
ValueOf(bb,6) = 29
parseShort(bb) = 45
parseShort(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
Short.reverseBytes(to_rev) = 11520
```

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.