Java.Lang.Long class in Java

Difficulty Level : Easy Last Updated : 05 Aug, 2021

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

- **Long(long b):** Creates a Long object initialized with the value provided.

```
Syntax : public Long(long b)
Parameters :
b : value with which to initialize
```

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

```
Syntax : public Long(String s)
                    throws NumberFormatException
Parameters :
s : string representation of the long value
Throws :
NumberFormatException : If the string provided does not represent any long val
```

Methods:

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

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

2. toHexString() : Returns the string corresponding to the long value in hexadecimal form, that is it returns a string representing the long value in hex characters-[0-9][a-f]

```
Syntax : public String toHexString(long b)
Parameters :
```

```
b : long value for which hex string representation required.
```

3. toOctalString() : Returns the string corresponding to the long value in octal form, that is it returns a string representing the long value in octal characters-[0-7]

```
Syntax : public String toOctalString(long b)
Parameters :
b : long value for which octal string representation required.
```

4. toBinaryString() : Returns the string corresponding to the long value in binary digits, that is it returns a string representing the long value in hex characters-[0/1]

```
Syntax : public String toBinaryString(long b)
Parameters :
b : long value for which binary string representation required.
```

5. valueOf() : returns the Long object initialized with the value provided.

```
Syntax : public static Long valueOf(long b)
Parameters :
b : a long value
```

Another overloaded function valueOf(String val,long radix) which provides function similar to
new Long(Long.parseLong(val,radix))

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

Another overloaded function valueOf(String val) which provides function similar to
new Long(Long.parseInt(val,10))

```
Syntax : public static Long valueOf(String s)
```

```
            throws NumberFormatException
Parameters :
s : a String object to be parsed as long
Throws :
NumberFormatException : if String cannot be parsed to a long value in given ra
```

6. parseLong() : returns long value by parsing the string in radix provided. Differs from valueOf() as it returns a primitive long value and valueOf() return Long object.

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

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

```
Syntax : public static long parseLong(String val)
             throws NumberFormatException

Parameters :
val : String representation of long
Throws :
NumberFormatException : if String cannot be parsed to a long value in given ra
```

7. getLong() : returns the Long object representing the value associated with the given system property or null if it does not exist.

```
Syntax : public static Long getLong(String prop)
Parameters :
prop : System property
```

Another overloaded method which returns the second argument if the property does not exist, that is it does not return null but a default value supplied by user.

```
Syntax : public static Long getLong(String prop, long val)
Parameters :
prop : System property
val : value to return if property does not exist.
```

Another overloaded method which parses the value according to the value returned, that is if the value returned starts with “#”, than it is parsed as hexadecimal, if starts with “0”, than it is parsed as octal, else decimal.

```
Syntax : public static Long getLong(String prop, Long val)
Parameters :
prop : System property
val : value to return if property does not exist.
```

8. decode() : returns a Long 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 Long decode(String s)
             throws NumberFormatException
Parameters :
s : encoded string to be parsed into long val
Throws :
NumberFormatException : If the string cannot be decoded into a long value
```

9. rotateLeft() : Returns a primitive long by rotating the bits left by given distance in two’s complement form of the value given. When rotating left, the most significant bit is moved to the right hand side, or least significant position i.e. cyclic movement of bits takes place. Negative distance signifies right rotation.

```
Syntax : public static long rotateLeft(long val, int dist)
Parameters :
val : long value to be rotated
dist : distance to rotate
```

10. rotateRight() : Returns a primitive long by rotating the bits right by given distance in the twos complement form of the value given. When rotating right, the least significant bit is moved to the left hand side, or most significant position i.e. cyclic movement of bits takes place. Negative distance signifies left rotation.

Syntax : public static long rotateRight(long val, int dist)
Parameters :
val : long value to be rotated
dist : distance to rotate

```
// Java program to illustrate
// various Long class methods
public class Long_test
{
    public static void main(String args[])
    {
        long b = 55;
        String bb = "45";

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

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

        // toHexString(),toOctalString(),toBinaryString()
        // converts into hexadecimal, octal and binary forms.
        System.out.println("toHexString(b) =" + Long.toHexString(b));
        System.out.println("toOctalString(b) =" + Long.toOctalString(b));
        System.out.println("toBinaryString(b) =" + Long.toBinaryString(b));

        // valueOf(): return Long object
        // an overloaded method takes radix as well.
        Long z = Long.valueOf(b);
        System.out.println("valueOf(b) = " + z);
        z = Long.valueOf(bb);
        System.out.println("ValueOf(bb) = " + z);
        z = Long.valueOf(bb, 6);
        System.out.println("ValueOf(bb,6) = " + z);

        // parseLong(): return primitive long value
        // an overloaded method takes radix as well
        long zz = Long.parseLong(bb);
        System.out.println("parseLong(bb) = " + zz);
```

```
        zz = Long.parseLong(bb, 6);
        System.out.println("parseLong(bb,6) = " + zz);

        // getLong(): can be used to retrieve
        // long value of system property
        long prop = Long.getLong("sun.arch.data.model");
        System.out.println("getLong(sun.arch.data.model) = " + prop);
        System.out.println("getLong(abcd) =" + Long.getLong("abcd"));

        // an overloaded getLong() method
        // which return default value if property not found.
        System.out.println("getLong(abcd,10) =" + Long.getLong("abcd", 10));

        // decode() : decodes the hex,octal and decimal
        // string to corresponding long values.
        String decimal = "45";
        String octal = "005";
        String hex = "0x0f";

        Long dec = Long.decode(decimal);
        System.out.println("decode(45) = " + dec);
        dec = Long.decode(octal);
        System.out.println("decode(005) = " + dec);
        dec = Long.decode(hex);
        System.out.println("decode(0x0f) = " + dec);

        // rotateLeft and rotateRight can be used
        // to rotate bits by specified distance
        long valrot = 2;
        System.out.println("rotateLeft(0000 0000 0000 0010 , 2) =" +
                                    Long.rotateLeft(valrot, 2));
        System.out.println("rotateRight(0000 0000 0000 0010,3) =" +
                                    Long.rotateRight(valrot, 3));
    }
}
```

Output:

```
toString(b) = 55
toHexString(b) =37
toOctalString(b) =67
toBinaryString(b) =110111
valueOf(b) = 55
ValueOf(bb) = 45
ValueOf(bb,6) = 29
parseInt(bb) = 45
parseInt(bb,6) = 29
```

```
getLong(sun.arch.data.model) = 64
getLong(abcd) =null
getLong(abcd,10) =10
decode(45) = 45
decode(005) = 5
decode(0x0f) = 15
rotateLeft(0000 0000 0000 0010 , 2) =8
rotateRight(0000 0000 0000 0010,3) =1073741824
```

Some more Long class methods are –

11. byteValue() : returns a byte value corresponding to this Long Object.

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

12. shortValue() : returns a short value corresponding to this Long Object.

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

13. intValue() : returns a int value corresponding to this Long Object.

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

14. longValue() : returns a long value corresponding to this Long Object.

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

15. doubleValue() : returns a double value corresponding to this Long Object.

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

16. floatValue() : returns a float value corresponding to this Long Object.

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

17. hashCode() : returns the hashcode corresponding to this Long Object.

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

18. bitcount() : Returns number of set bits in twos complement of the long given.

```
Syntax : public static int bitCount(long i)
Parameters :
i : long value whose set bits to count
```

19. numberOfLeadingZeroes() : Returns number of 0 bits preceding the highest 1 bit in twos complement form of the value, i.e. if the number in twos complement form is 0000 1010 0000 0000, then this function would return 4.

```
Syntax : public static int numberofLeadingZeroes(long i)
Parameters :
i : long value whose leading zeroes to count in twos complement form
```

20. numberOfTrailingZeroes() : Returns number of 0 bits following the last 1 bit in twos complement form of the value, i.e. if the number in twos complement form is 0000 1010 0000 0000, then this function would return 9.

```
Syntax : public static int numberofTrailingZeroes(long i)
Parameters :
i : long value whose trailing zeroes to count in twos complement form
```

21. highestOneBit() : Returns a value with at most a single one bit, in the position of highest one bit in the value given. Returns 0 if the value given is 0, that is if the number is 0000 0000 0000 1111, than this function return 0000 0000 0000 1000 (one at highest one bit in the given number)

```
Syntax : public static long highestOneBit(long i)
Parameters :
i : long value
```

22. LowestOneBit() : Returns a value with at most a single one bit, in the position of lowest one bit in the value given. Returns 0 if the value given is 0, that is if the number is 0000 0000 0000 1111, than this function return 0000 0000 0000 0001 (one at highest one bit in the given number)

```
Syntax : public static long LowestOneBit(long i)
Parameters :
i : long value
```

23. equals() : Used to compare the equality of two Long objects. This methods returns true if both the objects contains same long 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
```

24. compareTo() : Used to compare two Long objects for numerical equality. This should be used when comparing two Long 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(Long b)
Parameters :
b : Long object to compare with
```

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

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

26. signum() : returns -1 for negative values, 0 for 0 and +1 for values greater than 0.

```
Syntax : public static int signum(long val)
Parameters :
val : long value for which signum is required.
```

27. reverse() : returns a primitive long value reversing the order of bits in two's complement form of the given long value.

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

28. reverseBytes() : returns a primitive long value reversing the order of bytes in two's complement form of the given long value.

Syntax : public static long reverseBytes(long val)

Parameters :

val : long value whose bits to reverse in order.

```
// Java program to illustrate
// various Long methods
public class Long_test
{
    public static void main(String args[])
    {
        long b = 55;
        String bb = "45";

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

        // xxxValue can be used to retrieve
        // xxx type value from long value.
        // xxx can be int,byte,short,long,double,float
        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());

        long value = 45;

        // bitcount() : can be used to count set bits
        // in twos complement form of the number
        System.out.println("Long.bitcount(value)=" + Long.bitCount(value));

        // numberOfTrailingZeroes and numberOfLeaadingZeroes
        // can be used to count prefix and postfix sequence of 0
        System.out.println("Long.numberOfTrailingZeros(value)=" +
                           Long.numberOfTrailingZeros(value));
        System.out.println("Long.numberOfLeadingZeros(value)=" +
                           Long.numberOfLeadingZeros(value));

        // highestOneBit returns a value with one on highest
        // set bit position
        System.out.println("Long.highestOneBit(value)=" +
```

```
                                Long.highestOneBit(value));

        // highestOneBit returns a value with one on lowest
        // set bit position
        System.out.println("Long.lowestOneBit(value)=" +
                                Long.lowestOneBit(value));

        // reverse() can be used to reverse order of bits
        // reverseytes() can be used to reverse order of bytes
        System.out.println("Long.reverse(value)=" + Long.reverse(value));
        System.out.println("Long.reverseBytes(value)=" +
                                 Long.reverseBytes(value));

        // signum() returns -1,0,1 for negative,0 and positive
        // values
        System.out.println("Long.signum(value)=" + Long.signum(value));

        // hashcode() returns hashcode of the object
        int hash = x.hashCode();
        System.out.println("hashcode(x) = " + hash);

        // equals returns boolean value representing equality
        boolean eq = x.equals(y);
        System.out.println("x.equals(y) = " + eq);

        // compare() used for comparing two int values
        int e = Long.compare(x, y);
        System.out.println("compare(x,y) = " + e);

        // compareTo() used for comparing this value with some
        // other value
        int f = x.compareTo(y);
        System.out.println("x.compareTo(y) = " + f);
    }
}
```

Output :

```
bytevalue(x) = 55
shortvalue(x) = 55
intvalue(x) = 55
longvalue(x) = 55
doublevalue(x) = 55.0
floatvalue(x) = 55.0
Long.bitcount(value)=4
Long.numberOfTrailingZeros(value)=0
```

```
Long.numberOfLeadingZeros(value)=58
Long.highestOneBit(value)=32
Long.lowestOneBit(value)=1
Long.reverse(value)=-5476377146882523136
Long.reverseBytes(value)=3242591731706757120
Long.signum(value)=1
hashcode(x) = 55
x.equals(y) = false
compare(x,y) = 1
x.compareTo(y) = 1
```

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.