Java.lang.Boolean Class in Java

Difficulty Level : Easy Last Updated : 23 Aug, 2018

Java provides a wrapper class **Boolean** in java.lang package. The 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.

In addition, this class provides useful methods like to convert a boolean to a String and a String to a boolean, while dealing with a boolean variable.

Creating a Boolean object

Boolean class provides **two** constructors for creating Boolean object.

- The below statement creates a Boolean object which contain the value argument.

  ```
  Boolean b = new Boolean(boolean value);
  ```

- The below statement creates a Boolean object which contain the value true if the string argument is not null and is equal, ignoring case, to the string “true”, otherwise Boolean object with value false is created.

  ```
  Boolean b = new Boolean(String s);
  ```

Fields:

- **static Boolean FALSE :** The Boolean object corresponding to the primitive value false.
- **static Boolean TRUE :** The Boolean object corresponding to the primitive value true.
- **static Class :** The Class object representing the primitive type boolean.

Methods:

1. **static boolean parseBoolean(String s)** : This method parses the string argument as a boolean. The boolean returned represents the value true if the string argument is not null and is equal, ignoring case, to the string “true”, otherwise return false.

   ```
   Syntax :
   public static boolean parseBoolean(String s)
   Parameters :
   s - the String containing the boolean representation to be parsed
   Returns :
   the boolean represented by the string argument
   ```

```java
// Java program to demonstrate parseBoolean() method
public class Test
{
    public static void main(String[] args)
    {
        // parsing different Strings
        boolean b1 = Boolean.parseBoolean("True");
        boolean b2 = Boolean.parseBoolean("TruE");
        boolean b3 = Boolean.parseBoolean("False");
        boolean b4 = Boolean.parseBoolean("FALSE");
        boolean b5 = Boolean.parseBoolean("GeeksForGeeks");

        System.out.println(b1);
        System.out.println(b2);
        System.out.println(b3);
        System.out.println(b4);
        System.out.println(b5);

    }
}
```

Output:

```
true
true
false
false
false
```

2. **boolean booleanValue()** : This method returns the value of this Boolean object as a boolean primitive.

```
Syntax :
public boolean booleanValue()
Parameters :
NA
Returns :
the primitive boolean value of this object.
```

```java
// Java program to demonstrate booleanValue() method
public class Test
{
    public static void main(String[] args)
    {
        // creating different Boolean objects
        Boolean b1 = new Boolean("True");
        Boolean b2 = new Boolean("False");
        Boolean b3 = new Boolean("GeeksForGeeks");

        // getting primitive boolean value
        boolean b4 = b1.booleanValue();
        boolean b5 = b2.booleanValue();
        boolean b6 = b3.booleanValue();

        System.out.println(b4);
        System.out.println(b5);
        System.out.println(b6);

    }
}
```

Output:

```
true
false
false
```

3. **static Boolean valueOf(boolean b)** : This method returns a Boolean instance representing the specified boolean value. If the specified boolean value is true, it returns Boolean.TRUE or if it is false, then this method returns Boolean.FALSE. The other variant of this method is discussed next.

```
Syntax :
public static boolean valueOf(boolean b)
Parameters :
b - a boolean value.
Returns :
a Boolean object representing b.
```

```java
// Java program to demonstrate valueOf() method
public class Test
{
    public static void main(String[] args)
    {
        // creating boolean variable
        boolean b1 = true;
        boolean b2 = false;

        // getting Boolean objects from boolean variables
        Boolean b3 = Boolean.valueOf(b1);
        Boolean b4 = Boolean.valueOf(b2);

        System.out.println(b3);
        System.out.println(b4);

    }
}
```

Output:

```
true
false
```

4. **static Boolean valueOf(String s)** : This method returns a Boolean with a value represented by the specified string 's'. The Boolean returned represents a true value if the string argument is not null and is equal, ignoring case, to the string "true".

```
Syntax :
public static boolean valueOf(String s)
Parameters :
s - a string
Returns :
a Boolean value represented by the string
```

```
// Java program to demonstrate valueOf() method
public class Test
{
    public static void main(String[] args)
    {
        // creating boolean variable using different Strings
        Boolean b1 = Boolean.valueOf("true");
        Boolean b2 = Boolean.valueOf("TRue");
        Boolean b3 = Boolean.valueOf("False");
        Boolean b4 = Boolean.valueOf("GeeksForGeeks");
        Boolean b5 = Boolean.valueOf(null);
        System.out.println(b1);
        System.out.println(b2);
        System.out.println(b3);
        System.out.println(b4);
        System.out.println(b5);

    }
}
```

Output:

```
true
true
false
false
false
```

5. **static String toString(boolean b)** : This method returns a String object representing the specified boolean. If the specified boolean is true, then the string “true” will be returned, otherwise the string “false” will be returned.The other variant of this method is discussed next.

```
Syntax :
public static String toString(boolean b)
Parameters :
b - the boolean to be converted
Returns :
the string representation of the specified boolean
```

```
// Java program to demonstrate toString() method
public class Test
{
    public static void main(String[] args)
    {
        // creating boolean variable
        boolean b1 = true;
        boolean b2 = false;

        // getting String value of the primitives boolean
        String str1 = Boolean.toString(b1);
        String str2 = Boolean.toString(b2);
        System.out.println(str1);
        System.out.println(str2);
    }
}
```

Output:

```
true
false
```

6. **String toString()** : This method returns a String object representing this Boolean's value. If this object represents the value true, a string equal to "true" is returned. Otherwise, the string "false" is returned.

```
Syntax :
public String toString()
Parameters :
NA
Returns :
a string representation of this object
Overrides :
toString in class Object
```

```java
// Java program to demonstrate toString() method
public class Test
{
    public static void main(String[] args)
    {
        // creating different Boolean objects
        Boolean b1 = new Boolean("True");
        Boolean b2 = new Boolean("False");
        Boolean b3 = new Boolean("GeeksForGeeks");
        Boolean b4 = new Boolean(null);


        // getting String value of Boolean objects
        String str1 = b1.toString();
        String str2 = b2.toString();
        String str3 = b3.toString();
        String str4 = b4.toString();

        System.out.println(str1);
        System.out.println(str2);
        System.out.println(str3);
        System.out.println(str4);
    }
}
```

Output:

```
true
false
false
false
```

7. **int hashCode()** : This method returns a hash code for this Boolean object. Note that hashcode for true is 1231 and for false is 1237. To find reason for choosing this integers as hashcode, refer here.

```
Syntax :
public int hashCode()
Parameters :
NA
Returns :
the integer 1231 if this object represents true;
returns the integer 1237 if this object represents false
Overrides :
hashCode in class Object
```

```
// Java program to demonstrate hashCode() method
public class Test
{
    public static void main(String[] args)
    {
        // creating different Boolean objects
        Boolean b1 = new Boolean("True");
        Boolean b2 = new Boolean("False");
        Boolean b3 = new Boolean("TRue");
        Boolean b4 = new Boolean(null);

        System.out.println(b1.hashCode());
        System.out.println(b2.hashCode());
        System.out.println(b3.hashCode());
        System.out.println(b4.hashCode());
    }
}
```

Output:

```
1231
1237
1231
1237
```

8. **boolean equals(Object obj)** : This method returns true iff the argument is not null and is a Boolean object that represents the same boolean value as this object.

```
Syntax :
public boolean equals(Object obj)
Parameters :
obj - the object to compare with.
Returns :
true if the Boolean objects represent the same value;
false otherwise
Overrides :
equals in class Object
```

```
// Java program to demonstrate equals() method
public class Test
{
    public static void main(String[] args)
    {
        // creating different Boolean objects
        Boolean b1 = new Boolean("True");
        Boolean b2 = new Boolean("False");
        Boolean b3 = new Boolean("TrUe");
        Boolean b4 = new Boolean(null);


        // checking equality of Boolean objects
        System.out.println(b1.equals(b2));
        System.out.println(b2.equals(b4));
        System.out.println(b1.equals(b3));
        System.out.println(b1.equals(b4));
    }
}
```

Output:

```
false
true
true
false
```

9. **int compareTo(Boolean b)** : This method “compares” this Boolean instance with passed argument ‘b’.

```
Syntax :
public int compareTo(Boolean b)
Parameters :
b - the Boolean instance to be compared
Returns :
zero if this object represents the same boolean value as the argument;
a positive value if this object represents true and the argument represents
a negative value if this object represents false and the argument represent
Throws :
NullPointerException - if the argument is null
```

```java
// Java program to demonstrate compareTo() method
public class Test
{
    public static void main(String[] args)
    {
        // creating different Boolean objects
        Boolean b1 = new Boolean("True");
        Boolean b2 = new Boolean("False");
        Boolean b3 = new Boolean("TRue");
        Boolean b4 = new Boolean(null);

        //comparing b1,b2,b3,b4
        System.out.println(b1.compareTo(b2));
        System.out.println(b1.compareTo(b3));
        System.out.println(b2.compareTo(b1));
        System.out.println(b1.compareTo(b4));
        System.out.println(b2.compareTo(b4));

        // The following statement throws NullPointerExcetion
        //  System.out.println(b1.compareTo(null));
    }
}
```

Output:

```
1
0
-1
1
0
```

10. **int compare(boolean x, boolean y)** : This method is used to “compares” primitives boolean variables.

```
Syntax :
public static int compare(boolean x, boolean y)
Parameters :
x - the first boolean to compare
y - the second boolean to compare
Returns :
zero if x is same boolean value as y;
a positive value x is true and y is false;
a negative value if x is false and y is true;
Throws :
NullPointerException - if the argument is null
```

```java
// Java program to demonstrate compare() method
public class Test
{
    public static void main(String[] args)
    {
        // creating boolean variable
        boolean b1 = true;
        boolean b2 = false;
        boolean b3 = true;
        boolean b4 = false;

        //comparing b1,b2,b3,b4
        System.out.println(Boolean.compare(b1, b2));
        System.out.println(Boolean.compare(b1, b3));
        System.out.println(Boolean.compare(b2, b1));
        System.out.println(Boolean.compare(b2, b4));
        // The following statement throws NullPointerExcetion
        //  System.out.println(Boolean.compare(b1, null));
    }
}
```

Output:

```
1
0
-1
0
```

This article is contributed by **Gaurav Miglani**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@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.