Right way to check if String is empty in Java

What do you most of us do while using String in Java? Checking whether String is null or empty right? I am sure you know couple of way to test whether String is empty or not, but do you know the right way to do it? When we talk about [Strings in Java](http://javarevisited.blogspot.sg/2013/07/java-string-tutorial-and-examples-beginners-programming.html), we can imagine them as arrays of characters, and they are, but in Java they are also object. An empty Java String, is considered as the not null String that contains zero characters, meaning its length is 0. However, a Java String that might only contain the white-space character is not considered as empty, it is considered to contain one character and its length is equal to 1. One of the most popular way of *checking whether String is empty or not* is String class' isEmpty() method, this looks perfect right, it's readable and returns Boolean if String is empty otherwise returns false, but problem is you can not call this method without checking whether String is null or not. In another word, this is not null safe and it [will throw NullPointerException if String is null](http://java67.blogspot.sg/2012/09/what-is-nullpointerexception-in-java.html). Another popular and faster way to check if String is empty or not is by checking it's length, e.g. if String.length() = 0 then String is empty, but this is also not null safe. Third common way of checking emptiness of String in Java is comparing it with empty String literal e.g. "".equals(str),this method is not as fast as previous two but it is null safe, you don't need to check for null, in case of null it will return false. So in my opinion, this is the right way to check if String is empty or not.  If you definition of empty String also includes null then you can also use Apache Commons Lang'sStringUtils class. It has methods like isEmpty() which return true for both null and empty String literal. Again this is also null safe and will not throw NullPointerException.

**2 Null Safe way to Check if String is Empty or Not in Java**

Here are couple of examples of testing for String emptiness without worrying about null inputs. These are not the fastest way of checking emptiness but avoids additional null checks.  
  
**1) Using equals Method**  
As discussed in my tips to deal with NullPointerException, I have mentioned a technique to call equals() method on known String object instead of calling on unknown object. This technique can be used to check *emptiness of String* as well. All you need to do is to call equals() method on [empty String literal](http://java67.blogspot.sg/2014/08/difference-between-string-literal-and-new-String-object-Java.html) and pass the object you are testing as shown below :

String nullString = null;

String empty = new String();

boolean test = "".equals(empty); // true

System.out.println(test);

boolean check = "".equals(nullString); // false

System.out.println(check);

There will be no null pointers in this case.  Both of these way confirms that *String is not null and empty*.  
  
  
**2) Using Apache Commons StringUtils class**  
If you are already using Apache commons then you can also use isEmpty() method to check if String is empty or not. Only caveat here is that this method return true in case of null input as well, which may not be correct depending upon your application's definition of empty String. If you treat null as empty String then you can use this wonderful [null-safe method](http://java67.blogspot.sg/2012/09/top-10-tricky-java-interview-questions-answers.html)for quick test, as shown in following example :

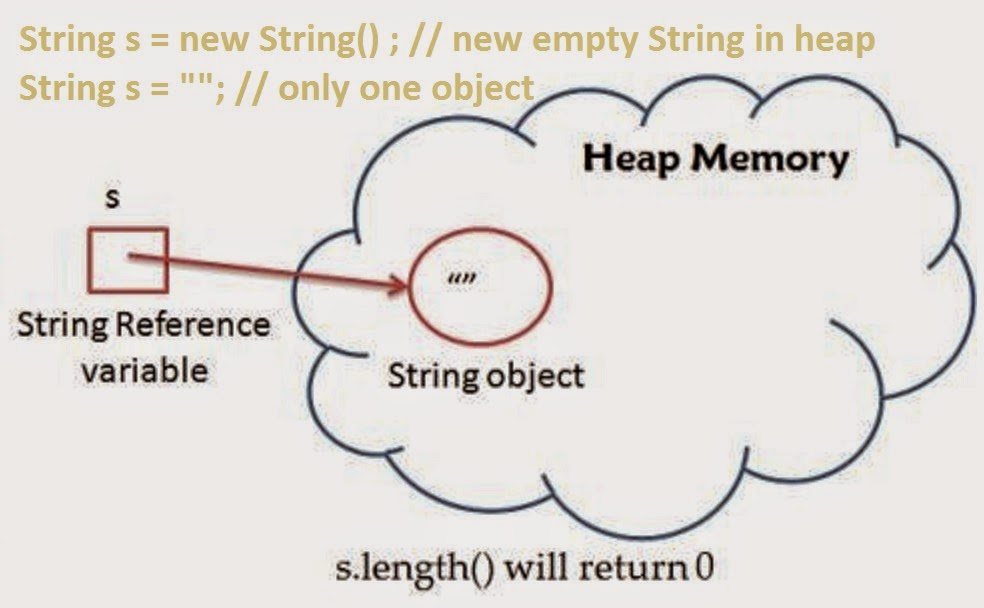
boolean nullCheck = StringUtils.isEmpty(nullString);

boolean emptyCheck = StringUtils.isEmpty("");

System.out.println(nullCheck); // true

System.out.println(emptyCheck); // true

You an see that check against null String is opposite to how equals method behave. So use this method with caution. Since they return true even for null input, they cannot be trusted for tests like not null and empty. By the way, if you have to create empty String, always use String literal "", it's more memory efficient. Since [String object is Immutable](http://java67.blogspot.sg/2014/01/why-string-class-has-made-immutable-or-final-java.html) and can safely share between threads, there is no need to create multiple separate String object, as String s = new String() will do, as shown in following diagram.

[](http://1.bp.blogspot.com/-0LN0RdV8Hvw/VBBaEA9i3QI/AAAAAAAAB38/sijM8W44BLg/s1600/Empty+String+check+in+Java.jpg)

**Right way to do Empty String check in Java**

There are many ways to check if String is empty in Java, but what is the right way of doing it? right in the sense of robustness, performance and readability.  If robustness is your priority then using [equals() method](http://java67.blogspot.sg/2012/11/difference-between-operator-and-equals-method-in.html) or Apache commons StringUtils is the right way to do this check. If you don't want to use third party library and happy of doing null check by yourself, then checking String's length is the fastest way and using isEmpty() method from String is most readable way. By the way,  don't confuse between empty and null String, if your application treat them same, then you can consider them same otherwise they are different, as null may not be classified as empty.  Here are three *examples of checking String is empty or not* by using JDK library itself.

import java.util.Arrays;

/\*\*

\* Java Program to check if String is empty or not. There are many ways to do

\* this but, but what is the right way?

\*

\* @author Javin Paul

\*/

public class StringEmptyTest {

public static **void** main(**String** args[]) {

**String** nonEmpty = "This is non Empty String";

**String** nullString = null;

**String** emptyString = "";

// Using isEmpty() method to check if String is empty in Java

// need null check to avoid NullPointerException

**String**[] inputs = {nonEmpty, nullString, emptyString};

**System**.out.println("\*\*\*\* isEmpty() method Example \*\*\*\*\*");

for (**String** s : inputs) {

if (s != null) {

**System**.out.println(**String**.format("Does String '%s' is empty? ", s) + s.isEmpty());

}

}

// Using length() method to check if String is empty

// for empty String length() == 0, require null check

**System**.out.println("\*\*\*\* length() method Example \*\*\*\*\*");

for (**String** s : inputs) {

if (s != null) {

**System**.out.println(**String**.format("Does String '%s' is empty? ", s) + (s.length() == 0));

}

}

// Using equals method to check if String is empty or not

// this approach is null-safe and doesn't require null check

// unlike previous examples.

**System**.out.println("\*\*\*\* equals() method Example \*\*\*\*\*");

for (**String** s : inputs) {

**System**.out.println(**String**.format("Does String '%s' is empty? ", s) + "".equals(s));

}

}

}

**Output**:

**Using** isEmpty() method from **java.lang.String** to check emptiness

**Does** **String** 'This is non Empty String' is empty? false

**Does** **String** '' is empty? true

**Does** **String** 'This is non Empty String' is empty? false

**Does** **String** '' is empty? true

**Does** **String** 'This is non Empty String' is empty? false

**Does** **String** 'null' is empty? false

**Does** **String** '' is empty? true

If you are sure that the String you are checking is not null then use length() method, it's the fastest way. If you [give paramount importance to readability](http://javarevisited.blogspot.sg/2012/11/What-is-timeunit-sleep-over-threadsleep.html) without compromising performance then String.isEmpty() is the right way to check String emptiness.  
  
  
That's all about **how to check if a String is empty in Java or not**. As I said, if your preference is towards safety from NullPointerException then null safe methods are best way to check emptiness e.g. calling equals on empty String literal or using StringUtils.isEmpty() method from Apache Commons Lang. If you are already doing null check and absolutely sure that the String you are checking is not null, use String.isEmtpy() method or simply length() method to check if length of String is Zero.