Reverse a String in 4 ways.

In this blog I will discuss how to reverse a String in 4 different ways.

Let us understand those ways one by one.

**Use the reverse API.**

This is most easiest solution to reverse a String. It uses reverse(str) to reverse the String. In interview we do not use this method as it does not represent any skills to interviewer. But it is good to know this API.

StringBuilder sb = new StringBuilder(str);

sb.reverse();

**Reverse by Appending character from Back to Front.**

In this method we iterate String from back to front. Now for every character iterated we append it to StringBuilder.

StringBuilder sb = new StringBuilder();

for (int i = str.length() - 1; i >= 0; i--) {

sb.append(str.charAt(i));

}

return sb.toString();

**Reverse by using Stack.**

This is one of the interview question. This question demostrate that does the candidate knows about the Stack's push and pop operations and How the stack works.

String "abc" would be be pushed onto Stack like this.

| c |

| b |

| a |

**Reverse by Recursion.**

This method is asked to know whether candiddate knows about the recursion or not.

This is simple method. In this we take subString of unprocessed string and append it with first character of string.

Here is code snippet for same.

reverseByRecursion(str.substring(1))+str.charAt(0);

Below are the test cases designed for all the methods.

**package** strings;

**import** **static** org.junit.Assert.\*;

**import** junit.framework.Assert;

**import** org.junit.Test;

**public** **class** StringReverseTest {

**final** String SINGLE\_LENGTH = "a";

**final** String STR1 = "abc";

@Test

**public** **void** testReverseByStringBuilderSingleLength() {

Assert.*assertEquals*("a",

StringReverse.*reverseByStringBuilder*(SINGLE\_LENGTH));

}

@Test

**public** **void** testReverseByStringBuilder() {

Assert.*assertEquals*("cba", StringReverse.*reverseByStringBuilder*(STR1));

}

@Test

**public** **void** testReverseByAppendingSingleLength() {

Assert.*assertEquals*("a", StringReverse.*reverseByAppend*(SINGLE\_LENGTH));

}

@Test

**public** **void** testReverseByAppending() {

Assert.*assertEquals*("cba", StringReverse.*reverseByAppend*(STR1));

}

@Test

**public** **void** testReverseByStackSingleLength() {

Assert.*assertEquals*("a", StringReverse.*reverseByStack*(SINGLE\_LENGTH));

}

@Test

**public** **void** testReverseByStack() {

Assert.*assertEquals*("cba", StringReverse.*reverseByStack*(STR1));

}

@Test

**public** **void** testReverseByRecursionSingleLength() {

Assert.*assertEquals*("a", StringReverse.*reverseByRecursion*(SINGLE\_LENGTH));

}

@Test

**public** **void** testReverseByRecursion() {

Assert.*assertEquals*("cba", StringReverse.*reverseByRecursion*(STR1));

}

}

Below is the code for the 4 methods.

**package** strings;

**import** java.util.Stack;

**public** **class** StringReverse {

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

}

**public** **static** String reverseByStringBuilder(String str) {

/\*\*

\* If input string is null then throw NullPointerException.

\* \*/

**if** (str == **null**) {

**throw** **new** NullPointerException("Input Should not be null");

}

//Below is one line code to reverse a String.

// return new StringBuffer().append(str).reverse().toString();

StringBuilder sb = **new** StringBuilder(str);

//Use reverse() method API to reverse a String

sb.reverse();

**return** sb.toString();

}

**public** **static** String reverseByAppend(String str) {

/\*\*

\* If input string is null then throw NullPointerException.

\* \*/

**if** (str == **null**) {

**throw** **new** NullPointerException("Input Should not be null");

}

/\*\*

\* Iterate through the string in reverse order

\* and append the character in StringBuilder.

\* \*/

StringBuilder sb = **new** StringBuilder();

**for** (**int** i = str.length() - 1; i >= 0; i--) {

sb.append(str.charAt(i));

}

**return** sb.toString();

}

**public** **static** String reverseByStack(String str){

/\*\*

\* If input string is null then throw NullPointerException.

\* \*/

**if** (str == **null**) {

**throw** **new** NullPointerException("Input Should not be null");

}

/\*\*

\* Push characters in Stack.

\* \*/

Stack<Character> stack=**new** Stack<Character>();

**for**(**char** ch:str.toCharArray()){

stack.push(ch);

}

StringBuilder sb=**new** StringBuilder();

/\*\*

\* As Stack is Last In First Out last character will be popped first.

\* Do it till Stack is empty.

\* \*/

**while**(!stack.isEmpty()){

sb.append(stack.pop());

}

//return the reversed String

**return** sb.toString();

}

**public** **static** String reverseByRecursion(String str){

/\*\*

\* If input string is null then throw NullPointerException.

\* \*/

**if**(str==**null**){

**throw** **new** NullPointerException("Input should not be null");

}

/\*\*

\* Base condition to end the recursion.

\* \*/

**if**(str.length()<2){

**return** str;

}

**return** *reverseByRecursion*(str.substring(1))+str.charAt(0);

}

}