**Strings Java 8**

**Java.util.StringJoiner in Java8**

StringJoiner is a class in [java.util](https://www.geeksforgeeks.org/tag/java-util-package/) package which is used to construct a sequence of characters(strings) separated by a delimiter and optionally starting with a supplied prefix and ending with a supplied suffix. Though this can also be done with the help of StringBuilder class to append delimiter after each string, but StringJoiner provides easy way to do that without much code to write.

**Constructors :**

**1) StringJoiner(CharSequence delimiter)** : Constructs a StringJoiner with no characters in it, with no prefix or suffix, and a copy of the supplied delimiter.

**Syntax** :

public StringJoiner(CharSequence delimiter)

**Parameters :**

delimiter - the sequence of characters to be used between

each element added to the StringJoiner value

**Throws:**

NullPointerException - if delimiter is null

**Example:**

**import** java.util.StringJoiner;

**public** **class** StringJoiner\_CharSequence\_delimiter {

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

// **TODO** Auto-generated method stub

StringJoiner joinNames=**new** StringJoiner(","); // passing comma(,) as delimiter

// Adding values to StringJoiner

joinNames.add("Rahul");

joinNames.add("Raju");

joinNames.add("Peter");

joinNames.add("Raheem");

System.***out***.println(joinNames);

joinNames=**new** StringJoiner("|"); // passing comma(,) as delimiter

// Adding values to StringJoiner

joinNames.add("Rahul");

joinNames.add("Raju");

joinNames.add("Peter");

joinNames.add("Raheem");

System.***out***.println(joinNames);

}

}

**Output:**

Rahul,Raju,Peter,Raheem

Rahul|Raju|Peter|Raheem

**2)StringJoiner(CharSequence delimiter,CharSequence prefix,CharSequence suffix)**: Constructs a StringJoiner with no characters in it using copies of the supplied prefix, delimiter and suffix. If no characters are added to the StringJoiner and methods accessing the string value of it are invoked, it will return the prefix + suffix (or properties thereof) in the result, unless setEmptyValue has first been called.

**Syntax :**

public StringJoiner(CharSequence delimiter,

CharSequence prefix, CharSequence suffix)

**Parameters :**

delimiter - the sequence of characters to be used between

each element added to the StringJoiner value

prefix - the sequence of characters to be used at the beginning

suffix - the sequence of characters to be used at the end

**Throws:**

NullPointerException - if prefix, delimiter, or suffix is null

**Example:**

**import** java.util.StringJoiner;

**public** **class** StringJoiner\_CharSequencedelimiter\_CharSequence\_prefix\_CharSequence\_suffix {

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

// **TODO** Auto-generated method stub

StringJoiner joinNames=**new** StringJoiner(",", "[", "]");

// Adding values to StringJoiner

joinNames.add("Rahul");

joinNames.add("Raju");

joinNames.add("Peter");

joinNames.add("Raheem");

System.***out***.println(joinNames);

}

}

**Output:**

[Rahul,Raju,Peter,Raheem]

**Methods :** There are 5 methods in StringJoiner class.

**StringJoiner toString() method in Java**

The toString() of [StringJoiner](https://www.geeksforgeeks.org/java-util-stringjoiner-java8/) is used to convert StringJoiner to String. It returns the current value, consisting of the prefix, the values added so far separated by the delimiter, and the suffix, unless no elements have been added in which case, the prefix + suffix or the emptyValue characters are returned

**Syntax:**

public String toString()

**Returns:** This method returns the string representation of this StringJoiner

Below programs illustrate the toString() method:

**Example 1:**

**import** java.util.StringJoiner;

**public** **class** StringJoiner\_toString {

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

// **TODO** Auto-generated method stub

StringJoiner joinNames=**new** StringJoiner(",", "[", "]");

// Adding values to StringJoiner

joinNames.add("Rahul");

joinNames.add("Raju");

joinNames.add("Peter");

joinNames.add("Raheem");

String str =joinNames.toString();

System.***out***.println(str);

}

}

**Output:**

[Rahul,Raju,Peter,Raheem]

**Example 2:**

import java.util.StringJoiner;

public class GFG {

public static void main(String[] args)

{

// Creating StringJoiner with delimeter " "

StringJoiner str = new StringJoiner(" ");

// Adding elements in the StringJoiner

str.add("Geeks");

str.add("for");

str.add("Geeks");

// Print the StringJoiner

// using toString() method

System.out.println("StringJoiner: "

+ str.toString());

}

}

**Output:**

StringJoiner: Geeks for Geeks

**Example 3:**

import java.util.StringJoiner;

public class GFG {

public static void main(String[] args)

{

// Creating StringJoiner with delimeter ""

StringJoiner str = new StringJoiner(", ");

// Adding elements in the StringJoiner

str.add("Geeks");

str.add("for");

str.add("Geeks");

str.add("A");

str.add("Computer");

str.add("Portal");

// Print the StringJoiner

// using toString() method

System.out.println("StringJoiner: "

+ str.toString());

}

}

**Output:**

StringJoiner: Geeks, for, Geeks, A, Computer, Portal

**StringJoiner add() method in Java**

The add(CharSequence newElement) of [StringJoiner](https://www.geeksforgeeks.org/java-util-stringjoiner-java8/) adds a copy of the given CharSequence value as the next element of the StringJoiner value. If newElement is null, then “null” is added.

**Syntax:**

public StringJoiner add(CharSequence newElement)

**Parameters:** This method takes a mandatory parameter newElement which is the element to be added.

**Returns:** This method returns a reference to this StringJoiner

Below programs illustrate the add() method:

**Example 1:**

// Java program to demonstrate

// add() method of StringJoiner

import java.util.StringJoiner;

public class GFG1 {

public static void main(String[] args)

{

// Creating StringJoiner with delimeter " "

StringJoiner str = new StringJoiner(" ");

// Adding elements in the StringJoiner

str.add("Geeks");

str.add("for");

str.add("Geeks");

// Print the StringJoiner

System.out.println(str.toString());

}

}

**Output:**

Geeks for Geeks

**Example 2:**

// Java program to demonstrate

// add() method of StringJoiner

import java.util.StringJoiner;

public class GFG1 {

public static void main(String[] args)

{

// Creating StringJoiner with delimeter ","

StringJoiner str = new StringJoiner(",");

// Adding elements in the StringJoiner

str.add("Geeks");

str.add("for");

str.add("Geeks");

// Print the StringJoiner

System.out.println(str.toString());

}

}

**Output:**

Geeks,for,Geeks

**StringJoiner merge() method in Java**

The merge(StringJoiner other) of [StringJoiner](https://www.geeksforgeeks.org/java-util-stringjoiner-java8/) adds the contents of the given StringJoiner without prefix and suffix as the next element if it is non-empty. If the given StringJoiner is empty, the call has no effect.

**Syntax:**

public StringJoiner merge(StringJoiner other)

**Parameters:** This method accepts a mandatory parameter other which is the StringJoiner whose contents should be merged into this one

**Returns:** This method returns this StringJoiner

**Exception:** This method throws NullPointerException if the other StringJoiner is null

Below programs illustrate the merge() method:

**Note: The merge() method merges two StringJoinerobjects excluding of prefix and suffix of the second StringJoiner object.**

**Example 1:**

// Java program to demonstrate

// merge() method of StringJoiner

import java.util.StringJoiner;

public class GFG {

public static void main(String[] args)

{

// Creating StringJoiner with delimeter " "

StringJoiner str1 = new StringJoiner(" ");

// Adding elements in the StringJoiner

str1.add("Geeks");

str1.add("for");

str1.add("Geeks");

// Print the StringJoiner

System.out.println("StringJoiner 1: "

+ str1.toString());

// Creating the second StringJoiner

StringJoiner str2 = new StringJoiner(" ");

str2.add("A");

str2.add("Computer");

str2.add("Portal");

// Print the second StringJoiner

System.out.println("StringJoiner 2: "

+ str2.toString());

// Merging the StringJoiner using merge()

StringJoiner str = str1.merge(str2);

// Printing the merged StringJoiner

System.out.println("Merged StringJoiner : "

+ str);

}

}

**Output:**

StringJoiner 1: Geeks for Geeks

StringJoiner 2: A Computer Portal

Merged StringJoiner : Geeks for Geeks A Computer Portal

**Example 2:**

To demonstrate merge() with delimeter “, ”

// Java program to demonstrate

// merge() method of StringJoiner

import java.util.StringJoiner;

public class GFG1 {

public static void main(String[] args)

{

// Creating StringJoiner with delimeter ", "

StringJoiner str1 = new StringJoiner(", ");

// Adding elements in the StringJoiner

str1.add("Geeks");

str1.add("for");

str1.add("Geeks");

// Print the StringJoiner

System.out.println("StringJoiner 1: "

+ str1.toString());

// Creating the second StringJoiner

StringJoiner str2 = new StringJoiner(", ");

str2.add("A");

str2.add("Computer");

str2.add("Portal");

// Print the StringJoiner

System.out.println("StringJoiner 2: "

+ str2.toString());

// Merging the StringJoiner using merge()

StringJoiner str = str1.merge(str2);

// Printing the merged StringJoiner

System.out.println("Merged StringJoiner : "

+ str);

}

}

**Output:**

StringJoiner 1: Geeks, for, Geeks

StringJoiner 2: A, Computer, Portal

Merged StringJoiner : Geeks, for, Geeks, A, Computer, Portal

**Example 3:**

 To demonstrate NullPointerException

// Java program to demonstrate

// merge() method of StringJoiner

import java.util.StringJoiner;

public class GFG1 {

public static void main(String[] args)

{

// Creating StringJoiner with delimeter ", "

StringJoiner str1 = new StringJoiner(", ");

// Adding elements in the StringJoiner

str1.add("Geeks");

str1.add("for");

str1.add("Geeks");

// Print the StringJoiner

System.out.println("StringJoiner 1: "

+ str1.toString());

// Creating the second StringJoiner

// to be merged as null

StringJoiner str2 = null;

// Print the StringJoiner

System.out.println("StringJoiner 2: "

+ str2);

try {

// Merging the StringJoiner using merge()

StringJoiner str = str1.merge(str2);

}

catch (Exception e) {

System.out.println("Exception during merge: "

+ e);

}

}

}

**Output:**

StringJoiner 1: Geeks, for, Geeks

StringJoiner 2: null

Exception during merge: java.lang.NullPointerException

**Example 4 :**

**import** java.util.StringJoiner;

**public** **class** StringJoiner\_merge {

StringJoiner joinNames=**new** StringJoiner(",", "[", "]");

// Adding values to StringJoiner

joinNames.add("Rahul");

joinNames.add("Raju");

// Creating StringJoiner with :(colon) delimiter

StringJoiner joinNames2 =**new** StringJoiner(":", "[", "]"); // passing colon(:) and square-brackets as delimiter

// Adding values to StringJoiner

joinNames2.add("Peter");

joinNames2.add("Raheem");

// Merging two StringJoiner

StringJoiner merge =joinNames.merge(joinNames2);

System.***out***.println(merge);

}

}

**Output:**

[Rahul,Raju,Peter:Raheem]

**StringJoiner length() method in Java**

The length() of [StringJoiner](https://www.geeksforgeeks.org/java-util-stringjoiner-java8/) is used to find out the length of the StringJoiner in characters. It returns the length of the String representation of this StringJoiner. Note that if no add methods have been called, then the length of the String representation (either prefix + suffix or emptyValue) will be returned. The value should be equivalent to toString().length().

**Syntax:**

public int length()

**Returns:** This method returns the length of the current value of StringJoiner

Below programs illustrate the length() method:

**Example 1:**

To demonstrate length() with delimeter ” ”

// Java program to demonstrate

// length() method of StringJoiner

import java.util.StringJoiner;

public class GFG {

public static void main(String[] args)

{

// Creating StringJoiner with delimeter " "

StringJoiner str = new StringJoiner(" ");

// Adding elements in the StringJoiner

str.add("Geeks");

str.add("for");

str.add("Geeks");

// Print the StringJoiner

System.out.println("StringJoiner: "

+ str.toString());

// Printing the length of str

// using length() method

System.out.println(str.length());

}

}

**Output:**

StringJoiner: Geeks for Geeks

15

**Example 2:**

To demonstrate length() with delimeter “, ”

// Java program to demonstrate

// length() method of StringJoiner

import java.util.StringJoiner;

public class GFG {

public static void main(String[] args)

{

// Creating StringJoiner with delimeter ""

StringJoiner str = new StringJoiner(", ");

// Adding elements in the StringJoiner

str.add("Geeks");

str.add("for");

str.add("Geeks");

str.add("A");

str.add("Computer");

str.add("Portal");

// Print the StringJoiner

System.out.println("StringJoiner: "

+ str.toString());

// Printing the length of str

// using length() method

System.out.println(str.length());

}

}

**Output:**

StringJoiner: Geeks, for, Geeks, A, Computer, Portal

38

**StringJoiner setEmptyValue() method in Java**

The setEmptyValue(CharSequence emptyValue) of [StringJoiner](https://www.geeksforgeeks.org/java-util-stringjoiner-java8/) sets the sequence of characters to be used when determining the string representation of this StringJoiner and no elements have been added yet, that is, when it is empty. A copy of the emptyValue parameter is made for this purpose. Note that once an add method has been called, the StringJoiner is no longer considered empty, even if the element(s) added correspond to the empty String.

**Syntax:**

public StringJoiner setEmptyValue(CharSequence emptyValue)

**Parameters:** This method accepts a mandatory parameter emptyValue which is the characters to return as the value of an empty StringJoiner

**Returns:** This method returns this StringJoiner itself, so the calls may be chained

**Exception:** This method throws NullPointerException when the emptyValue parameter is null

Below examples illustrates the setEmptyValue() method:

**Example 1:**

// Java program to demonstrate

// setEmptyValue() method of StringJoiner

import java.util.StringJoiner;

public class GFG {

public static void main(String[] args)

{

// Create a StringJoiner

StringJoiner str = new StringJoiner(" ");

// Print the empty StringJoiner

System.out.println("Initial StringJoiner: "

+ str);

// Add an emptyValue

// using setEmptyValue() method

str.setEmptyValue("StrigJoiner is empty");

// Print the StringJoiner

System.out.println("After setEmptyValue(): "

+ str);

// Add elements to StringJoiner

str.add("Geeks");

str.add("forGeeks");

// Print the StringJoiner

System.out.println("Final StringJoiner: "

+ str);

}

}

**Output:**

Initial StringJoiner:

After setEmptyValue(): StrigJoiner is empty

Final StringJoiner: Geeks forGeeks

**Example 2:**

To demonstrate NullPointerException

// Java program to demonstrate

// setEmptyValue() method of StringJoiner

import java.util.StringJoiner;

public class GFG {

public static void main(String[] args)

{

// Create a StringJoiner

StringJoiner str = new StringJoiner(" ");

// Print the empty StringJoiner

System.out.println("Initial StringJoiner: "

+ str);

try {

// Add a null emptyValue

// using setEmptyValue() method

str.setEmptyValue(null);

}

catch (Exception e) {

System.out.println("Exception when adding null"

+ " in setEmptyValue(): " + e);

}

}

}

**Output:**

Initial StringJoiner:

Exception when adding null in setEmptyValue(): java.lang.NullPointerException: The empty value must not be null

Below is the java program to demonstrate all methods.

// Java program to demonstrate methods

// of StringJoiner class

import java.util.ArrayList;

import java.util.StringJoiner;

public class Test2

{

public static void main(String[] args)

{

ArrayList<String> al = new ArrayList<>();

al.add("Ram");

al.add("Shyam");

al.add("Alice");

al.add("Bob");

StringJoiner sj1 = new StringJoiner(",");

// setEmptyValue() method

sj1.setEmptyValue("sj1 is empty");

System.out.println(sj1);

// add() method

sj1.add(al.get(0)).add(al.get(1));

System.out.println(sj1);

// length() method

System.out.println("Length of sj1 : " + sj1.length());

StringJoiner sj2 = new StringJoiner(":");

sj2.add(al.get(2)).add(al.get(3));

//merge() method

sj1.merge(sj2);

// toString() method

System.out.println(sj1.toString());

System.out.println("Length of new sj1 : " + sj1.length());

}

}

**Output:**

sj1 is empty

Ram,Shyam

Length of sj1 : 9

Ram,Shyam,Alice:Bob

Length of new sj1 : 19

Examples from java doc: The String "[George:Sally:Fred]" may be constructed as follows:

StringJoinersj=newStringJoiner(":", "[", "]");

sj.add("George").add("Sally").add("Fred");

String desiredString=sj.toString();

**Java 8 Streams | Collectors.joining() method with Examples**

The joining() method of Collectors Class, in Java, is used to join various elements of a character or string array into a single string object. This method uses stream to do so. There are various overloads of joining method present in the Collector class.

Class Heirarchy:

java.lang.Object

↳ java.util.stream.Collectors

joining()

java.util.stream.Collectors.joining() is the most simple joining method which do not takes any parameter. It returns a Collector which joins or concatenates the input streams into String in the order of their appearance.

**Syntax:**

public static Collector<CharSequence, ?, String> joining()

Below are the illustration for how to use joining() method:

**Program 1: Using joining() with an array of characters:**

In the below program, a character array is created in ‘ch’. Then this array is fed to be converted into Stream using Stream.of(). Then the resulted stream is mapped for a sequential series using map(). At last the sequential stream containing the character array is joined into a String using Collectors.joining() method. It is stored in ‘chString’ variable.

|  |
| --- |
| // Java Program to demonstrate the working  // of the Collectors.joining() method  import java.util.stream.Collectors;  import java.util.stream.Stream;  public class GFG {  public static void main(String[] args)  {  // Create a character array  char[] ch = { 'G', 'e', 'e', 'k', 's',  'f', 'o', 'r',  'G', 'e', 'e', 'k', 's' };  // Convert the character array into String  // using Collectors.joining() method  String chString = Stream.of(ch)  .map(arr -> new String(arr))  .collect(Collectors.joining());  // Print the concatenated String  System.out.println(chString);  }  } |

**Output:**

GeeksforGeeks

**Program 2: Using joining() with a list of characters:**

In the below program, a character list is created in ‘ch’. Then this list is fed to be converted into Stream using ch.stream() method. Then the resulted stream is mapped for a sequential series using map(). At last the sequential stream containing the character list is joined into a String using Collectors.joining() method. It is stored in ‘chString’ variable.

|  |
| --- |
| // Java Program to demonstrate the working  // of the Collectors.joining() method  import java.util.stream.Collectors;  import java.util.stream.Stream;  import java.util.Arrays;  import java.util.List;  public class GFG {  public static void main(String[] args)  {  // Create a character list  List<Character> ch = Arrays.asList('G', 'e', 'e', 'k', 's',  'f', 'o', 'r',  'G', 'e', 'e', 'k', 's');  // Convert the character list into String  // using Collectors.joining() method  String chString = ch.stream()  .map(String::valueOf)  .collect(Collectors.joining());  // Print the concatenated String  System.out.println(chString);  }  } |

**Output:**

GeeksforGeeks

**Program 3: Using joining() with n list of string:**

In the below program, a String list is created in ‘str’. Then this list is fed to be converted into Stream using str.stream() method. Then the resulted stream is mapped for a sequential series using map(). At last the sequential stream containing the character list is joined into a String using Collectors.joining() method. It is stored in ‘chString’ variable.

|  |
| --- |
| // Java Program to demonstrate the working  // of the Collectors.joining() method  import java.util.stream.Collectors;  import java.util.stream.Stream;  import java.util.Arrays;  import java.util.List;  public class GFG {  public static void main(String[] args)  {  // Create a character list  List<String> str = Arrays.asList("Geeks", "for", "Geeks");  // Convert the character list into String  // using Collectors.joining() method  String chString = str.stream()  .map(String::valueOf)  .collect(Collectors.joining());  // Print the concatenated String  System.out.println(chString);  }  } |

**Output:**

GeeksforGeeks

**joining(delimiter)**

java.util.stream.Collectors.joining(CharSequence delimiter) is an overload of joining() method which takes delimiter as parameter, of the type CharSequence. A delimiter is a symbol or a CharSequence that are used to seperate words from each other. For example, in every sentence, space ‘ ‘ is used as the by default delimiter for the words in it. It returns a Collector which joins or concatenates the input elements into String in the order of there appearance, seperated by the delimiter.

**Syntax:**

public static Collector<CharSequence, ?, String> joining(CharSequence delimiter)

Below are the illustration for how to use joining(delimiter) method:

**Program 1: Using joining(delimiter) with a list of characters:**

In the below program, a character list is created in ‘ch’. Then this list is fed to be converted into Stream using ch.stream() method. Then the resulted stream is mapped for a sequential series using map(). At last the sequential stream containing the character list is joined into a String using Collectors.joining() method with “, ” passed as the delimiter. It is stored in ‘chString’ variable.

|  |
| --- |
| // Java Program to demonstrate the working  // of the Collectors.joining() method  import java.util.stream.Collectors;  import java.util.stream.Stream;  import java.util.Arrays;  import java.util.List;  public class GFG {  public static void main(String[] args)  {  // Create a character list  List<Character> ch = Arrays.asList('G', 'e', 'e', 'k', 's',  'f', 'o', 'r',  'G', 'e', 'e', 'k', 's');  // Convert the character list into String  // using Collectors.joining() method  // with, as the delimiter  String chString = ch.stream()  .map(String::valueOf)  .collect(Collectors.joining(", "));  // Print the concatenated String  System.out.println(chString);  }  } |

**Output:**

G, e, e, k, s, f, o, r, G, e, e, k, s

**Program 2: Using joining(delimiter) with a list of string:**

In the below program, a String list is created in ‘str’. Then this list is fed to be converted into Stream using str.stream() method. Then the resulted stream is mapped for a sequential series using map(). At last the sequential stream containing the character list is joined into a String using Collectors.joining() method with “, ” passed as the delimiter. It is stored in ‘chString’ variable.

|  |
| --- |
| // Java Program to demonstrate the working  // of the Collectors.joining() method  import java.util.stream.Collectors;  import java.util.stream.Stream;  import java.util.Arrays;  import java.util.List;  public class GFG {  public static void main(String[] args)  {  // Create a character list  List<String> str = Arrays.asList("Geeks", "for", "Geeks");  // Convert the character list into String  // using Collectors.joining() method  String chString = str.stream()  .map(String::valueOf)  .collect(Collectors.joining(", "));  // Print the concatenated String  System.out.println(chString);  }  } |

**Output:**

Geeks, for, Geeks

**joining(delimiter, prefix, suffix)**

java.util.stream.Collectors.joining(CharSequence delimiter, CharSequence prefix, CharSequence suffix) is an overload of joining() method which takes delimiter, prefix and suffix as parameter, of the type CharSequence.

A delimiter is a symbol or a CharSequence that are used to seperate words from each other. A prefix is a symbol or a CharSequence that is joined at the starting of the 1st element of the String. Then suffix is also a CharSequence parameter but this is joined after the last element of the string. i.e. at the end.

For example, in every {Geeks, for, Geeks}, space ‘ ‘ is used as the by default delimiter for the words in it. The ‘{‘ is the prefix and ‘}’ is the suffix. It returns a Collector which joins or concatenates the input elements into String in the order of there appearance, seperated by the delimiter.

**Syntax:**

public static Collector<CharSequence, ?, String> joining(CharSequence delimiter.

CharSequence prefix,

CharSequence suffix))

Below are the illustration for how to use joining(delimiter, prefix, suffix) method:

**Program 1: Using joining() with a list of characters:**

In the below program, a character list is created in ‘ch’. Then this list is fed to be converted into Stream using ch.stream() method. Then the resulted stream is mapped for a sequential series using map(). At last the sequential stream containing the character list is joined into a String using Collectors.joining() method with “, ” passed as the delimiter, “[” as the prefix and “]” as the suffix. It is stored in ‘chString’ variable.

|  |
| --- |
| // Java Program to demonstrate the working  // of the Collectors.joining() method  import java.util.stream.Collectors;  import java.util.stream.Stream;  import java.util.Arrays;  import java.util.List;  public class GFG {  public static void main(String[] args)  {  // Create a character list  List<Character> ch = Arrays.asList('G', 'e', 'e', 'k', 's',  'f', 'o', 'r',  'G', 'e', 'e', 'k', 's');  // Convert the character list into String  // using Collectors.joining() method  // with, as the delimiter  String chString = ch.stream()  .map(String::valueOf)  .collect(Collectors.joining(", ", "[", "]"));  // Print the concatenated String  System.out.println(chString);  }  } |

**Output:**

[G, e, e, k, s, f, o, r, G, e, e, k, s]

**Program 2: Using joining() with a list of string:**

In the below program, a String list is created in ‘str’. Then this list is fed to be converted into Stream using str.stream() method. Then the resulted stream is mapped for a sequential series using map(). At last the sequential stream containing the character list is joined into a String using Collectors.joining() method with “, ” passed as the delimiter, “{” as the prefix and “}” as the suffix. It is stored in ‘chString’ variable.

|  |
| --- |
| // Java Program to demonstrate the working  // of the Collectors.joining() method  import java.util.stream.Collectors;  import java.util.stream.Stream;  import java.util.Arrays;  import java.util.List;  public class GFG {  public static void main(String[] args)  {  // Create a character list  List<String> str = Arrays.asList("Geeks", "for", "Geeks");  // Convert the character list into String  // using Collectors.joining() method  String chString = str.stream()  .map(String::valueOf)  .collect(Collectors.joining(", ", "{", "}"));  // Print the concatenated String  System.out.println(chString);  }  } |

**Output:**

{Geeks, for, Geeks}

**When to use StringJoiner over StringBuilder?**

Prerequisite: [StringJoiner](https://www.geeksforgeeks.org/java-util-stringjoiner-java8/)

StringJoiner is very useful, when you need to join Strings in a Stream.

**Task :**Suppose we want the string “[George:Sally:Fred]”, where we have given a string array that contains “George”, “Sally” and “Fred”.

StringJoiner provide add(String str) method to concatenate the strings based on supplied delimiter,prefix and suffix in the constructor, but if we use [StringBuilder](https://www.geeksforgeeks.org/g-fact-27-string-vs-stringbuilder-vs-stringbuffer/) to perform our task then first we have to append prefix and then iterate through string array and append the required delimiter after each element and finally append the suffix. Below is the java program to demonstrate both ways.

|  |
| --- |
| // Java program to demonstrate use of  // StringJoiner class over StringBuilder class  import java.util.StringJoiner;  public class Test  {  public static void main(String[] args)  {  // given string array  String str[] = {"George","Sally","Fred"};  // By using StringJoiner class  // initializing StringJoiner instance with  // required delimiter, prefix and suffix  StringJoiner sj = new StringJoiner(":", "[", "]");  // concatenating strings  sj.add("George").add("Sally").add("Fred");  // converting StringJoiner to String  String desiredString = sj.toString();  System.out.println(desiredString);  // By using StringBuilder class  // declaring empty stringbuilder  StringBuilder sb = new StringBuilder();  // appending prefix  sb.append("[");  // cheking for empty string array  if(str.length>0)  {  // appending first element  sb.append(str[0]);  // iterating through string array  // and appending required delimiter  for (int i = 1; i < str.length; i++)  {  sb.append(":").append(str[i]);  }  }  // finally appending sufix  sb.append("]");  // converting StringBuilder to String  String desiredString1 = sb.toString();  System.out.println(desiredString1);  }  } |

**Output:**

[George:Sally:Fred]

[George:Sally:Fred]