Joiner class

Joiner class is used to concatenate strings together with a given delimiter.

Let us understand it with example.

Let’s say we have

Example 1:

List<String> as ["Monday", "Tuesday", "Wednesday"]

Delimiter as “|”

Output should be “Monday| Tuesday| Wednesday”

Example 2:

List<String> as [“Mon”, ”Tues”, null, ”Wed”]

Delimiter as “|”

Output should be “Mon| Tues| Wed”

Let us first try to write the method for this functionality. We won’t use Joiner class right now. Let us code our method and see what problem it faces.

/\*\*

\* In this method we write the code to join the list of Strings

\* or List<String> with a delimiter.

\* So, Example 1

\* List is ["Monday","Tuesday","Wednesday"]

\* delimiter is "|"

\* Output

\* Monday|Tuesday|Wednesday

\*

\* Example 2

\* List is ["Mon", null,"Tues", "Wed"]

\* delimiter is "|"

\* Output

\* Mon||Tues|Wed

\* null are to be avoided.

\* \*/

**public** **static** String joinDelimiter(**final** List<String> strList,**final** String delimiter) {

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

Iterator<String> iter = strList.iterator();

**while** (iter.hasNext()) {

String string = iter.next();

**if** (string != **null**) {

sb.append(string).append(delimiter);

}

}

sb.setLength(sb.length()-delimiter.length());

**return** sb.toString();

}

Now we have the above method and it works fine. It ignores the null too. But here the problem is. What if I want to pass List<Integer> or List<Float> or List<> of some custom type. What if I want to pass map?

Now this is the problem. Mostly we tend to build this method as need comes by. But it is still not the solution.

Here Google Guava comes into picture.

Now let us learn Joiner class and its method that we can use to make our job easier.

Joiner class

This class is used to join the pieces of text specified as array or Iterable<E> or varargs or Map with a separator. It either appends the result to Appendable or returns it as String.

One thing notable over here is that it can use Iterable<E>. The reason behind this is that Collection<E> interface extends Iterable<E> so every collection can be used in Joiner class.

All the objects in the argument are converted to string using Object.toString() before appended.

If argument holds null values then we can filter them using skipNulls() or useForNull(String). If this methods are not used and nulls are passed then NullPointerException is thrown.

Joiner is Immutable. So make sure that you use it properly.

Joiner join = Joiner.on(“,”);

join.skipNulls();

return joiner.join(“worong”, null, ”wrong”);

Let us now understand how to use API Joiner.

List<String> list1=**new** ArrayList<String>(Arrays.*asList*("Monday", "Tuesday", "Wednesday"));

We have the above list with us and we want to use separator “|” between all the elements in List.

So we want to join on “|” skipNulls() and then join(list)

In guava we write like this

Joiner.*on*("|").skipNulls().join(strList);

Joiner class is explained above.

on() method – it is public static method that returns new Joiner instance. Remember Joiner is Immutable.

There are 2 variants of on() method.

on(String seperator)- returns new Joiner instance with separator in it.

**public** **static** Joiner on(String separator) {

**return** **new** Joiner(separator);

}

on(char seperator) – character separator is converted to String (using String.valueOf(char)) separator and returns new joiner instance with String separator.

**public** **static** Joiner on(**char** separator) {

**return** **new** Joiner(String.valueOf(separator));

}

skipNulls() – is used to skip all the null entries and returns new Joiner instance that will skip nulls.

join(list) – join method is used to join the entries of the argument. Let us understand it using without List example. So List<E> extends Collection<E> and Collection<E> extends Iterable<E> so

join(Iterable<?> parts) will be called.

**public** **final** String join(Iterable<?> parts) {

**return** join(parts.iterator());

}

Above method calls below method as we now get iterator()

**public** **final** String join(Iterator<?> parts) {

**return** appendTo(**new** StringBuilder(), parts).toString();

}

Now append the entries and the separator. The entries are converted to string using Object.toString() method.

Below is the code for using Joiner class.

**package** com.google.guava.learning.chapter1;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Iterator;

**import** java.util.List;

**import** com.google.common.base.Joiner;

/\*\*

\* Joiner class - object which joins pieces of text specified as an

\* array, Iterable, varargs or Map with a separator.

\*

\* It either appends the results to an Appendable or returns them

\* as a String.

\* \*/

**public** **class** JoinerDemo {

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

List<String> list = **new** ArrayList<String>(

Arrays.*asList*("Mon", **null**, "Tues", "Wed"));

System.***out***.println("Skip nulls "+*joinUsingGuavaJoiner*(list, "|"));

System.***out***.println("Use for null "+*joinUsingGuavaJoinerUseForNull*(list,"|"));

}

/\*\*

\* In this method we write the code to join the list of Strings

\* or List<String> with a delimiter.

\* So, Example 1

\* List is ["Monday","Tuesday","Wednesday"]

\* delimiter is "|"

\* Output

\* Monday|Tuesday|Wednesday

\*

\* Example 2

\* List is ["Mon", null,"Tues", "Wed"]

\* delimiter is "|"

\* Output

\* Mon||Tues|Wed

\* null are to be avoided.

\* \*/

**public** **static** String joinDelimiter(**final** List<String> strList,**final** String delimiter) {

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

Iterator<String> iter = strList.iterator();

**while** (iter.hasNext()) {

String string = iter.next();

**if** (string != **null**) {

sb.append(string).append(delimiter);

}

}

sb.setLength(sb.length()-delimiter.length());

**return** sb.toString();

}

/\*\*

\* Now let us solve this using Google Guava.

\*

\* Joiner class's on method returns new Joiner(separator);

\*

\* Then it used skipNulls() to skip null elements in List.

\*

\* Then join method is used to join delimiter between

\* two elements of list.

\* \*/

**public** **static** String joinUsingGuavaJoiner(List<String> strList, String delimiter){

**return** Joiner.*on*("|").skipNulls().join(strList);

}

/\*\*

\* This method is similar to above except one difference.

\*

\* In above method we skipped the null elements in list.

\*

\* But in this method we will use some nullText to denote

\* the null element in List.

\* \*/

**public** **static** String joinUsingGuavaJoinerUseForNull(List<String> list, String delimiter){

**return** Joiner.*on*("|").useForNull("#").join(list);

}

}

Test cases for the above class.

**package** com.google.guava.learning.test.chapter1;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**import** com.google.guava.learning.chapter1.JoinerDemo;

**import** junit.framework.Assert;

**import** junit.framework.TestCase;

**public** **class** JoinerDemoTest **extends** TestCase{

List<String> list1=**new** ArrayList<String>(

Arrays.*asList*("Monday", "Tuesday", "Wednesday"));

String output1 = "Monday|Tuesday|Wednesday";

List<String> list2 = **new** ArrayList<String>(

Arrays.*asList*("Mon", **null**, "Tues", "Wed"));

String output2="Mon|Tues|Wed";

String output3="Mon|#|Tues|Wed";

**public** **static** **final** String ***DELIMITER***="|";

/\*\*

\* Method written without Joiner class API

\* List does not contain null

\* \*/

**public** **void** testJoinDelimiter() {

Assert.*assertEquals*(output1, JoinerDemo.*joinDelimiter*(list1, ***DELIMITER***));

}

/\*\*

\* Method written without Joiner class API

\* List does contains null

\* \*/

**public** **void** testJoinDelimiterWithNull() {

Assert.*assertEquals*(output2, JoinerDemo.*joinDelimiter*(list2, ***DELIMITER***));

}

/\*\*

\* Method written using Joiner class API

\* List does not contain null

\* \*/

**public** **void** testjoinUsingGuavaJoiner() {

Assert.*assertEquals*(output1, JoinerDemo.*joinUsingGuavaJoiner*(list1, ***DELIMITER***));

}

/\*\*

\* Method written using Joiner class API

\* List does contains null

\* \*/

**public** **void** testjoinUsingGuavaJoinerNull() {

Assert.*assertEquals*(output2, JoinerDemo.*joinUsingGuavaJoiner*(list2, ***DELIMITER***));

}

/\*\*

\* Method written using Joiner class API

\* List does contains null

\* We represent null using null

\* \*/

**public** **void** testjoinUsingGuavaJoinerNullText() {

Assert.*assertEquals*(output3, JoinerDemo.*joinUsingGuavaJoinerUseForNull*(list2, ***DELIMITER***));

}

}

Test cases execution.