**STREAMS**

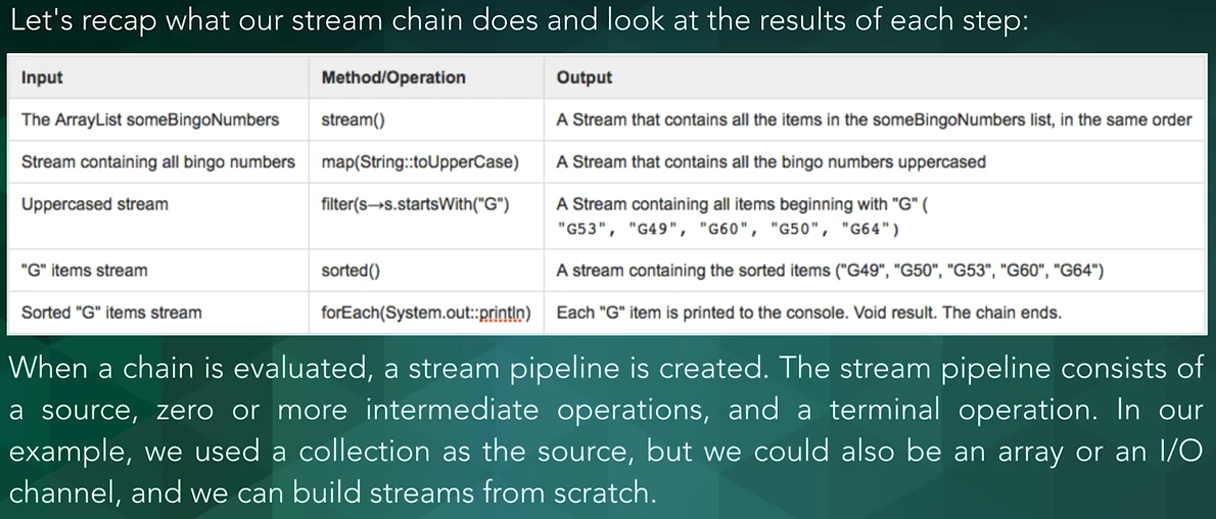
**Streams** are objects that represent sources and destinations of data. **Streams** that are sources of data can be read from, and **streams** that are destinations of data can be written to. A **stream** in **Java** is an ordered sequence of bytes of undetermined length.

## Details on Interface Stream<T>

[**https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html**](https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html)

|  |  |
| --- | --- |
| package com.company;  import java.util.ArrayList; import java.util.Arrays; import java.util.List;  public class Main {   public static void main(String[] args) {  List<String> strings = Arrays.*asList*(  "a01", "a02",  "b01", "b04", "b02", "B00",  "c01", "c02",  "d01", "d02", "d03", "d04"  );   ArrayList<String> snow = new ArrayList<>();  *//case 2* System.*out*.println("case 2");  strings.forEach(number -> {  if (number.toUpperCase().startsWith("B")){  snow.add(number);  }  });  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  *//arraylist without shorting* snow.forEach(s -> System.*out*.println(s));   System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  snow.sort((s1, s2) -> s1.compareTo(s2));  snow.forEach((s) -> System.*out*.println(s));  } } | case 2  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  b01  b04  b02  B00  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  B00  b01  b02  b04 |
| **WITH USING STREAM** | |
| **package** com.company; **import** java.util.ArrayList; **import** java.util.Arrays; **import** java.util.List; **public class** Main {  **public static void** main(String[] args) {  List<String> strings = Arrays.*asList*(  **"a01"**, **"a02"**,  **"b01"**, **"b04"**, **"b02"**, **"B00"**,  **"c01"**, **"c02"**,  **"d01"**, **"d02"**, **"d03"**, **"d04"** );   ArrayList<String> snow = **new** ArrayList<>();  strings  .stream()  .map(String::toUpperCase) *//s -> s.toUpperCase()* .filter(s -> s.startsWith(**"B"**)) //PREDICATE  .sorted()  .forEach(System.***out*** :: println);  } } | B00  B01  B02  B04 |

**FOREACH IS CALLED TERMINAL OPERATION METHOD,**

****

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.ArrayList; **import** java.util.Arrays; **import** java.util.List; **import** java.util.stream.Stream;  **public class** Main {   **public static void** main(String[] args) {  List<String> strings = Arrays.*asList*(  **"a01"**, **"a02"**,  **"b01"**, **"b04"**, **"b02"**, **"B00"**,  **"c01"**, **"c02"**,  **"d01"**, **"d02"**, **"d03"**, **"d04"** );   ArrayList<String> snow = **new** ArrayList<>();  strings  .stream()  .map(String::toUpperCase) *//s -> s.toUpperCase()* .filter(s -> s.startsWith(**"B"**))  .sorted()  .forEach(System.***out*** :: println);   System.***out***.println(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**);   Stream<String> s1 = Stream.*of*(**"I26"**, **"b20"**, **"C12"**, **"K44"**);  Stream<String> s2 = Stream.*of*(**"I20"**, **"I26"**, **"I99"**, **"I9"**, **"I5"**);  Stream<String> concats = Stream.*concat*(s1, s2);  *//System.out.println(concats.count()); ------------🡪 9* System.***out***.println(concats.distinct().count()); | **B00**  **B01**  **B02**  **B04**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **8** |

**Checking the distinct values**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.ArrayList; **import** java.util.Arrays; **import** java.util.List; **import** java.util.stream.Stream;  **public class** Main {   **public static void** main(String[] args) {  List<String> strings = Arrays.*asList*(  **"a01"**, **"a02"**,  **"b01"**, **"b04"**, **"b02"**, **"B00"**,  **"c01"**, **"c02"**,  **"d01"**, **"d02"**, **"d03"**, **"d04"** );   ArrayList<String> snow = **new** ArrayList<>();  strings  .stream()  .map(String::toUpperCase) *//s -> s.toUpperCase()* .filter(s -> s.startsWith(**"B"**))  .sorted()  .forEach(System.***out*** :: println);   System.***out***.println(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**);   Stream<String> s1 = Stream.*of*(**"I26"**, **"b20"**, **"C12"**, **"K44"**);  Stream<String> s2 = Stream.*of*(**"I20"**, **"I26"**, **"I99"**, **"I9"**, **"I5"**);  Stream<String> concats = Stream.*concat*(s1, s2);  *//System.out.println(concats.count());* System.***out***.println(concats  .distinct()  .peek(System.***out***::println)  .count());   } } | B00  B01  B02  B04  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  I26  b20  C12  K44  I20  I99  I9  I5  8 |

**As it is a terminal operation, we cant use foreach method, when we want to pull the values without completing the chain as well;**