# Exercise: Stream API and Recursion in Java

This document defines in-class exercise problems from the [“Java Fundamentals“ Course @ Software University](https://softuni.bg/trainings/1232/java-fundamentals-october-2015). You are presented with some problems and certain steps you need to take in order to accomplish the tasks.

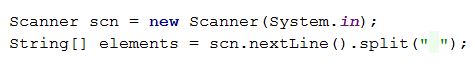
## Problem 1. Filter Array

Write a program that filters an array of strings, finds the numbers in it, and maps them to a list of integers. Sum all the integers and print the result.

|  |  |
| --- | --- |
| **Input** | **Output** |
| Banica 12 gosho j1k 33 kl bozichka 81 88kl 5562 abcd 8 I te taka | 5696 |

### Step 1. Read and Process the Input

To prepare our program, we need a Scanner and a string array, where we will keep the elements before transforming them to stream.



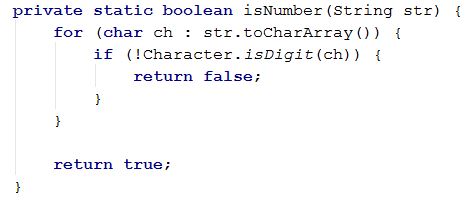
### Step 2. Filter the Numbers

Java arrays cannot be directly transformed to streams. To do so, we need to use the static method **Arrays.stream()**

**C:\Users\Edu\Pictures\stream.JPG**

To remove the elements that are not numbers, we can use the .filter method. It works with a lambda that evaluates a Boolean expression. If the result is true, the element is passed down the stream, else – it is filtered. First we need to create the method to check if a string is a number. It is pretty straightforward – we iterate through al l the characters and check if every one of them is digit.

Note that this method will not work for negative numbers!



Now that we have the method, we can use it in the filter:

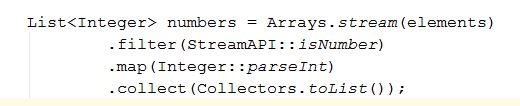
C:\Users\Edu\Pictures\filter.JPG

**StreamAPI** is the class name that holds the static isNumber methods. In your program, it may be different

Then, we have to transform the filtered strings in to numbers. We can use the .map method. It works similarly to .Select in C#:

C:\Users\Edu\Pictures\map.JPG

Finally, we must collect the results of the stream. This is done via the .collect method. We will keep the numbers in a List of Integers. The complete statement should look something like this.



### Step 3. Sum the Resulting List

There are several ways to obtain the total sum of a collection of numbers in Java. In C#, you must be familiar with the .**Sum**() method, which works on any type of collection, as long as that collection is **Enumerable**. In Java, the .**sum**() method can be called on an **IntStream**. So, although you have a collection of integers, you will have to transform them one more time. The required method is **mapToInt(),** which returns an **IntStream**.C:\Users\Edu\Pictures\maptoint.JPG

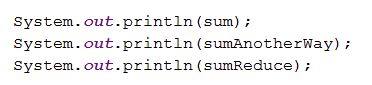
Another way is to use the **Collectors’** **summingInt**() method:

C:\Users\Edu\Pictures\sumanotherway.JPG

A third way, maybe the most intuitive one, is to use the .reduce() method, which works on any stream

C:\Users\Edu\Pictures\sumreduce.JPG

Print the three obtained numbers and see if the results are equal.



## Print Numbers from 1 to N Using Recursion

Printing numbers is quite easy. You must be familiar with the usual way of using a for/while loop. Recursion may be a little difficult to grasp at first. It is actually a method that calls itself with updated arguments. The method will continue calling itself until the argument reaches its base case. At that point, the method starts returning the accumulated values. In our case, those values will be simple integers.

### Step 1. Read and Process the Input

First, we need to take int N from the console. This is done with a simple Scanner:

C:\Users\Edu\Pictures\inputrecursive.JPG

### Step 2. Create the Recursive Method

To use recursion, we need a method. Create a simple method printRecursive(int n) that calls itself with (n – 1) until the number **n** is smaller than 1. For instance, if **n** is equal to 5, in the stack we will have:

printRecursive(1)

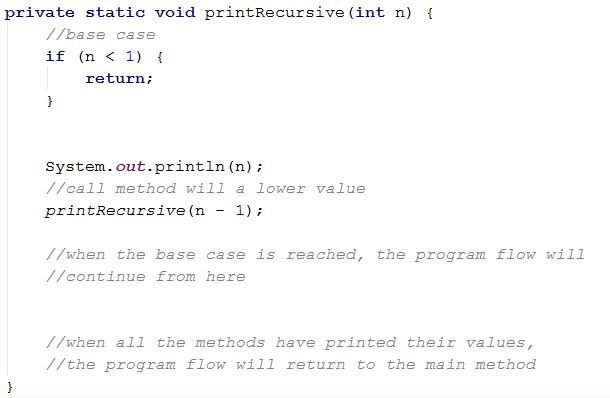
printRecursive(2)

printRecursive(3)

printRecursive(4)

printRecursive(5)

When we call printRecursive(0), the program flow will enter the base case, which will stop it. At that point, the methods that are pushed into the stack will start returning their values. Since a stack always returns the element at the top, the first returned value will be 1, the next will be 2, and so on. Print those values:



All you need to do is call the method from main() with the initial value.

*If you move the println() call before the recursive call, you will notice that the numbers are printed from top to bottom (5, 4, 3, 2, 1), instead of (1, 2, 3, 4, 5).* ***Think why.***

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 | 1  2  3  4  5 |