Programming exercise:

Average of Numbers

Implement a program, which reads user input. If the user input is "end", the program stops reading input. The rest of the input is numbers. When the user input is "end", the program prints the average of all of the numbers.

Implement calculating the average using a stream!

Sample output

```
Input numbers, type "end" to stop.
2
4
6
end
average of the numbers: 4.0
```

Sample output

```
Input numbers, type "end" to stop.
-1
1
2
end
average of the numbers: 0.6666666666666666
```

Programming exercise:

Average of selected numbers

Implement a program, which reads user input. If the user input is "end", program stops reading input. The rest of the input is numbers.

Then user is asked if the program should print the average of all the positive numbers, or the average of all the negative numbers (n or p). If the user selects "n", the average of all the negative numbers is printed. Otherwise the average of all the positive numbers is printed.

Use streams to calculate the average and filter the numbers!

Sample output

```
Input numbers, type "end" to stop.
-1
1
2
end

Print the average of the negative numbers or the positive numbers? (n/p)
n
Average of the negative numbers: -1.0
```

Sample output

```
Input numbers, type "end" to stop.
-1
1
2
end

Print the average of the negative numbers or the positive numbers? (n/p)
p
Average of the positive numbers: 1.5
```

Programming exercise:

Positive Numbers

In the exercise template, implement the class method `public static List<Integer> positive(List<Integer> numbers)`, which receives an ArrayList of integers, and returns the positive integers from the list.

Implement the method using stream! For collecting the numbers try the command `Collectors.toList()` in addition to the `Collectors.toCollection(ArrayList::new)` command.

Programming exercise:

Divisible

The exercise template includes a template for the method `public static ArrayList<Integer> divisible(ArrayList<Integer> numbers)`. Implement a functionality there that gathers numbers divisible by two, three or five from the list it receives as a parameter, and returns them as a new list. The list received as a parameter must not be altered.

```
public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(3);
        numbers.add(2);
        numbers.add(-17);
        numbers.add(-5);
        numbers.add(7);

        ArrayList<Integer> divisible = divisible(numbers);

        divisible.stream()
                .forEach(num -> System.out.println(num));
    }
```

Sample output

```
3
2
-5
```

Programming exercise:

Printing User Input

Write a program that reads the user's input as strings. When the user inputs an empty string (only presses enter), the input reading will be stopped and the program will print all the user inputs.

Sample output

first
second
war is peace: 1984

first
second
war is peace: 1984

Programming exercise:

Limited numbers

Write a program that reads user input. When the user gives a negative number as an input, the input reading will be stopped. After this, print all the numbers the user has given as input that are between 1 and 5.

Sample output

```
7
14
4
5
4
-1
4
5
4
```

Programming exercise:

Unique last names

The exercise template contains a sketch of a program that reads user-provided information about people. Expand the program so that it will print all the unique last names of the user-provided people in alphabetical order.

Sample output

```
Continue personal information input? "quit" ends:

Input first name: Ada
Input last name: Lovelace
Input the year of birth: 1815

Continue personal information input? "quit" ends:

Input first name: Grace
Input last name: Hopper
Input the year of birth: 1906

Continue personal information input? "quit" ends:

Input first name: Alan
Input last name: Turing
Input the year of birth: 1912

Continue personal information input? "quit" ends:
quit

Unique last names in alphabetical order:
Hopper
Lovelace
Turing
```