Seminar 7
week 7 (14-20 November 2017)

1. Questions from **Lab-Assignment 4 from Laboratory 5**.
2. Discussion of the **Lab-Assignment 5 from Laboratory 7**.
3.Solve the following problems using the functional programming style (**using Java Streams):**

please start with a List of Strings similar to this:
• List<String> words = Arrays.asList("hi", "hello", ...);

P1. Loop down the words and print each on a separate line, with two spaces in front of each word. Don't use map.

P2. Repeat the previous problem, but without the two spaces in front. This is trivial if you use the same approach as in #1, so the point is to use a method reference here, as opposed to an explicit lambda in problem 1.

P3. We assume that we have a method StringUtils.transformedList(List<String>, Function1<String>)
where interface Function1<T> { T app(T);}
and we produced transformed lists like this:
• List<String> excitingWords = StringUtils.transformedList(words, s -> s + "!");
• List<String> eyeWords = StringUtils.transformedList(words, s -> s.replace("i", "eye"));
• List<String> upperCaseWords = StringUtils.transformedList(words, String::toUpperCase);
Produce the same lists as above, but this time use streams and the builtin "map" method.

P4. We assume that we have the method StringUtils.allMatches(List<String>, Predicate1<String>)
where interface Predicate1<T> { boolean check(T);}
and we produced filtered lists like this:
• List<String> shortWords = StringUtils.allMatches(words, s -> s.length() < 4);
• List<String> wordsWithB = StringUtils.allMatches(words, s -> s.contains("b"));
• List<String> evenLengthWords = StringUtils.allMatches(words, s -> (s.length() % 2) == 0);
Produce the same lists as above, but this time use "filter".

P5. Turn the strings into uppercase, keep only the ones that are shorter than 4 characters, of what is remaining, keep only the ones that contain "E", and print the first result. Repeat the process, except checking for a "Q" instead of an "E". When checking for the "Q", try to avoid repeating all the code from when you checked for an "E".

P6. Produce a single String that is the result of concatenating the uppercase versions of all of the Strings. Use a single reduce operation, without using map.

P7. Produce the same String as above, but this time via a map operation that turns the words into uppercase, followed by a reduce operation that concatenates them.

P8. Produce a String that is all the words concatenated together, but with commas in between. E.g., the result should be "hi,hello,...". Note that there is no comma at the beginning, before "hi", and also no comma at the end, after the last word. Major hint: there are two versions of reduce discussed in the notes.

P9. Find the total number of characters (i.e., sum of the lengths) of the strings in the List.

P10. Find the number of words that contain an “h”.

4. Discuss some of Java 9 additions to Stream:

TakeWhile

```
Stream.of("a", "b", "c", "", "e").takeWhile(s -> !String.isEmpty(s));forEach(System.out::print);
//Result: abd
```

DropWhile

```
Stream.of("a", "b", "c", "de", "f").dropWhile(s -> s.length <= 1);.forEach(System.out::print);
//Result: def
```

Iterate from Java 8

```
Stream.iterate(1, i -> 2 * i).forEach(System.out::println);
// output: 1 2 4 8 ...
```

Iterate from Java 9

```
Stream.iterate(1, i -> i <= 10, i -> 2 * i).forEach(System.out::println);
// output: 1 2 4 8
```