Seminar 6
week 6 (4-8 November 2019)

Announcement: Seminar written test is in week 8 (18-22 November 2019). The test requires all the labs and the seminars materials.

A. **Discussion of the implementation for the lab assignment A3.** Regarding the View part we discuss how it would be possible to call many times the execution of the same example.
B. **Discussion of the following IO classess usage:** FileReader, FileWriter, BufferedReader, BufferedWriter, StreamTokenizer, Scanner and PrintStream. Some code templates of using these classes are given below:

- **FileReader class example:**

```
try(FileReader fileReader = new FileReader("c:\\data\\text.txt")){
        int data = fileReader.read();
    while(data != -1) { // read a char
     System.out.print((char) data));
     data = fileReader.read();
    }
}
```

- **FileWriter class example:**

```
try(FileWriter fileWriter = new FileWriter("data\\filewriter.txt",true)){
       //true –appends, false or nothing-overwrites
        fileWriter.write("data 1");
        fileWriter.write("data 2");
        fileWriter.write("data 3");
}
```

- **BufferedReader class example:**

```
Reader reader = new FileReader("data.bin");
try(BufferedReader bufferedReader =new BufferedReader(reader)){
   String line = bufferedReader.readLine();
   while(line != null) {
      //do something with line

      line = bufferedReader.readLine();
   }
}
 or

br=new BufferedReader(new FileReader(numefis));
     String linie;
     while((linie=br.readLine())!=null){
        String[] elems=linie.split("[|]");
        if (elems.length<2){
           System.err.println("Linie invalida "+linie);
        continue;}
     //do something with the line
   }
```

- **BuferredWriter class example:**

```
FileWriter output = new FileWriter("data.bin");
try(BufferedWriter bufferedWriter = new BufferedWriter(output)){
   for(i=0;i<100;i++){
         bufferedWriter.write("Hello World");
         bufferedWriter.newLine();
         if(i%5==0)
            bufferedWriter.flush();
   }
}
```

- **StreamTokenizer class example:**

```
Reader reader = new FileReader("data.bin");
try(StreamTokenizer streamTokenizer = new StreamTokenizer(reader)){

   while(streamTokenizer.nextToken() != StreamTokenizer.TT_EOF){

      if(streamTokenizer.ttype == StreamTokenizer.TT_WORD) {
         System.out.println(streamTokenizer.sval);
      } else if(streamTokenizer.ttype == StreamTokenizer.TT_NUMBER) {
         System.out.println(streamTokenizer.nval);
      } else if(streamTokenizer.ttype == StreamTokenizer.TT_EOL) {
         System.out.println();
      }
   }
}
```

- **PrintWriter class example:**

```
         FileWriter  writer      = new FileWriter("report.txt");
         PrintWriter printWriter = new PrintWriter(writer);
         printWriter.print(true);
         printWriter.print((int) 123);
         printWriter.print((float) 123.456);
         intVar i=200;
         printWriter.printf("Text + data: %d", intVar);
         printWriter.close();
```

- **Scanner class examples:**

```
Scanner sc = new Scanner(new File("myNumbers"));
     while (sc.hasNextLong()) {
        long aLong = sc.nextLong();
     }
```

C. **Please 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. Please use forEach()

P2. Repeat the previous problem, but without the two spaces in front. This is trivial if you use the same approach as in P1, so the point is to use a method reference here, as opposed to an explicit lambda as in P1.

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.