2. semester ST/SE Videregående Objektorienteret Programmering Lecture 2

Subjects for Lecture 2

The main topic for next week lecture is Character-based Streams & Introduction to HTML. But we will quickly revise the following topics and further explain more concepts in them.

1. String and Character
2. Exception Handling

Preparation Lecture 2

Note : If you did not finish the two exercises from lecture 1, you should do them now 😊.

1. Refresh your knowledge about the *java.lang.String*- and *Character*-classes, from *Liang 4.3* and *4.4*.
   Exercise: (A reduced version of an exam exercise). See "CamelWriter" exercise (VOP-2->VOP-2 Preparation (Resources and Activities)) and try to do it by following the "CamelWriter Instructions". A project skeleton with limited functionality is provided as a zipped folder.

2. Study *Savitch&Mock, chp 10.1 – 10.2 (*pages 775 – 788*)*. See VOP-2->VOP-2 Preparation (Resources and Activities)
   Exercise: Small part of an exam exercise: See "DanishIsland" exercise, and try to do it by following the "DanishIsland Instructions". A project skeleton with limited functionality is provided as a zipped folder.

   Note: The solutions will be provided after the lecture

From ReExam F11 CamelWriter

Main topics:

- File I/O

- Operations on text strings and characters

- Exception Handling

Provided files:

- DryLips.txt: lyrics by Dúné

- OhLand.txt: Lyrics by Oh-Land

- MaryAnn.txt: Lyrics by Volbeat

- camelwriter.CamelWriter.java: Code skeleton that can be used as a starting point.

In this assignment, a java program will be developed that takes a text file as input and converts each line into one word, written in camel script following the convention of java methods and variables:

- The first word in the line is written in lower case

- Subsequent words are capitalized

- The words are put together without spaces

When a line is converted to a word in camel script, it is written to the textfile

Example: first verse of Dúné's Dry Lips before and after conversion:

I'm walking down my neighbourhood Where every child is crying You're the only one who sees me You're the only one who'll kiss me On my dry lips	i'mWalkingDownMyNeighbourhood whereEveryChildIsCrying you'reTheOnlyOneWhoSeesMe you'reTheOnlyOneWho'llKissMe onMyDryLips

Hint:

- *Use a Scanner to read the input file one line at a time.*
- *Use split(" ") on the line so that each word is now in a String array*
- *Convert the 1st word to lower case and the first letter of the rest of the words to upper case*
- *Assemble the line words into one long word in camel script and print it.*
- *Write the converted output to a text file.*

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package comparable;

import java.util.*;


public class Person implements Comparable<Person>{
    private String fName;
    private String lName;
    private GregorianCalendar birthDay;
    private double heigth;

    public Person(String fName, String lName, int bYear, int bMonth, int bDate, double heigth) {
        this.fName = fName;
        this.lName = lName;
        this.birthDay = new GregorianCalendar(bYear, bMonth, bDate);
        this.heigth = heigth;
    }

    @Override
    public String toString() {
        return "fName=" + fName + ", lName=" + lName + ", birthDay=" + birthDay.getTime() + ", height_" + heigth+'\n';
    }

    //Opgave 1A:
    // Der skal sorteres på efternavn. Hvis ens, skal der sorteres på fornavn.
    // Hvis det stadig er ens sorteres på fødselsdag.
    @Override
    public int compareTo(Person o) {
        lastNamecompare = this.lastName.compareTo(o.
```

```
34 lastName);
35         if(lastNamecompare != 0){
36             return lastNamecompare;
37         }
38         int firstNameCompare = this.firstName.
   compareTo(o.firstName);
39         if(firstNamecompare !=0){
40             return firstNamecompare
41         }
42         return this.birthDay.compareTo(o.birthDay);
43     }
44
45
46
47
48     public static void main(String[] args) {
49         List<Person> list = new ArrayList();
50         list.add(new Person("A", "BB", 1980, 3, 17, 1
   .87));
51         list.add(new Person("B", "BB", 1980, 3, 8, 1.
   86));
52         list.add(new Person("A", "AA", 1980, 3, 9, 1.
   67));
53         list.add(new Person("A", "BB", 1980, 3, 10, 1
   .67));
54         list.add(new Person("A", "BB", 1980, 3, 1, 1.
   66));
55         list.add(new Person("A", "CC", 1980, 3, 1, 1.
   65));
56         Collections.sort(list);
57         System.out.println("\nsorted:\n" +list);
58         for(Person list : list){
59             System.out.println(list);
60         }
61
62         //Implement the compare method of Comparator
    (anonymous inner class in the main()-method of
   Person), so Persons can be sorted by their height.
63
64         Collections.sort(list, new Comparator<Person
   >() {
```

```
65              @Override
66              public int compare(Person o1, Person o2
    ) {
67                  return Double.compare(o1.height, o2.
    height);
68              }
69
70          });
71
72          // Fjern udkommenteringen, når Comparatoren
    er programmeret:
73
74          System.out.println("\nsorted:\n" +list);
75
76      }
77  }
78
```

```java
package textanalyzer;


import java.io.File;
import java.util.Map;

public class TextAnalyzer {

    private File file;

    public TextAnalyzer(String fileName) {
        file = new File(fileName);
    }



    // Opgave 2:   Nearly as Listing 21.9 from Liang
    //
    public Map<String, Integer> countWords() {
        Map<String, Integer> map = new TextAnalyzer<
Strng, Integer>();
        file = "alice30.txt";
        try(Scanner scanner = new Scanner(new File(
file));){
            while(file.hasNext){
                word = clean(word);
                if(word == null){
                    return false;
                }else if (word = true){
                    countWords();
                }
            }

        }
            //map = //create hashmap instance

        // gennemlæs filen "alice30.txt" et ord ad
gangen
        // kald clean() metoden på hvert ord
        // benyt mappen til at tælle forekomsten af
hvert ord
```

```
38
39          return map;
40      }
41
42
43
44      // Denne metode forsøger at fjerne alt 'snavs'
    fra en String,
45      // så kun bogstaver bevares og store gøres til
    små
46      private String clean(String s) {
47          String r = "";
48          for (int i = 0; i < s.length(); i++) {
49              char c = s.charAt(i);
50              if (Character.isLetter(c)) {
51                  r = r + c;
52              }
53          }
54          return r.toLowerCase();
55      }
56
57      /**
58       * @param args
59       */
60      public static void main(String[] args) {
61
62          TextAnalyzer ta = new TextAnalyzer("alice30.
    txt");
63
64          // Opgave 2. Tæl forekomster af ord
65          Map<String, Integer> map = ta.countWords();
66          System.out.println(map);
67          System.out.println("\n
    ----------------------------------------------------
    ------------\n");
68      }
69  }
70
```

From ReExam Danish Island

Unzip DanishIslandExercise.zip. You will find:

- Islands komma.txt
- Islands punktum.txt

The files contain various information about Danish inhabited islands. The information in the files is the same, except for whether the decimal point is a comma or a full stop. Each line represents one island with the following data (separated by spaces):

Navn Omkreds Areal Addresser Adr/km2 af datatyperne *(String, double, double, int, int).*

- A java package danishisland with the classes:

  - DanishIsland: objects of this class represent one island. Study the code, it is fully implemented.

  - DanishIslandFileReader: Code skeleton for reading the island text file. Study the code and implement what is missing

    - Load the file one line at a time. Split the line on the space character.
    - Convert the individual values to the types to be used in DanishIsland.
    - Create an object for each line and add it to the list.
    - Be careful to catch exceptions and close the scanner and file.
    - In the main() method you can choose whether to use comma or period as decimal character for double values

```
C:\Users\vivek\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2.1\lib\idea_rt.jar=56476:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2.1\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\vivek\Downloads\DanishIslandExercise\DanishIslandExercise\target\classes org.example.danishisland.DanishIslandFileReader
C:\Users\vivek\Downloads\DanishIslandExercise\.
Result:
[]
java.io.FileNotFoundException: Islands punktum.txt (Den angivne fil blev ikke fundet)
    at java.base/java.io.FileInputStream.open0(Native Method)
    at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:158)
    at java.base/java.util.Scanner.<init>(Scanner.java:641)
    at org.example.danishisland.DanishIslandFileReader.readFile(DanishIslandFileReader.java:25)
    at org.example.danishisland.DanishIslandFileReader.main(DanishIslandFileReader.java:68)

Process finished with exit code 0

```