Prof. Dr. Christoph Bockisch
MSc Steffen Dick
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Companion exercises
Objektorientierte Programmierung: Wintersemester 2021/2022

No. 7, due until 13.12.2021

Attention: You are able to achieve more than the usual 12 points, though only 12 points count towards your debit. Any achieved points above 12 will be granted as bonus points.

Task 7.1: Playing fair 10 Points

You are supposed to implement the Playfair-encryption within this task. The Playfair-encryption is a method that was created in 1854 by Charles Wheatstone. To encode a word, a 5 by 5 matrix containing letters is used.

This matrix consists of a codeword and the rest of the letters of the alphabet that were not contained within the codeword. Should the codeword contain more than one of a given letter, every following occurrence of that latter is ignored. Every J in the codeword is removed. As soon as the codeword is cleaned up, it is added to the matrix and the rest of the space is filled with the remaining alphabet-letters (excluding J).

Example: APFELSTRUDEL

```
A P F E L
S T R U D
B C G H I
K M N O Q
V W X Y Z
```

The word to be encrypted is cleaned if spaces and any special characters and then separated into pairs of two. Should any pair contain two of the same letter, a X is added between both and the second letter is added to the next pair. If the last pair only contains one letter, an A is added to that pair. MITTWOCH, for example, becomes MI TX TW OC HA, but OTTO stays OT TO.

Every pair can now be translated by using the matrix. For that translation, the following rules apply:

1. If both letters of the pair are within the same row within the matrix, the letter to their respective rights is used to encode each one. The first letter of a row is used if the original letter is the last one in that row.

   Example: Codeword *Apfelstrudel*

   AP $\rightarrow$ PF, EL $\rightarrow$ LA
2. If both letters of the pair are within the same column of the matrix, the letter below each one will be used. Is the original letter the last one in a column, the first letter of that column is used instead.

   Example: Codeword *Apfelstrudel*

PT → TC, MW → WP

3. Should neither rule apply, the letter that is in the same row as the original but in the same column as the other letter of the pair will be used.

   Example: Codeword *Apfelstrudel*

   AU → ES, HK → BO

a) Implement a class called *Playfair*. Create a private final field *playfairSquare* of *Character[][]*-type. Furthermore, create a private static final field of type String with the value *"ABCDEFGHIKLMNOPQRSTUVWXYZ"*. Use a name corresponding to the code-conventions for this field.

b) We need a helper method to populate *playfairSquare*. Implement `boolean characterInString(String s, Character character)` that returns `true` if a *String* s contains a *Character* c.

c) Implement the constructor `public Playfair(String codeword)` that populates the square as follows:
   1. Transform the codeword into upper case.
   2. Remove further occurrences of other letters from the codeword.
   3. Add the missing letters from the alphabet to the codeword so that it has exactly 25 characters.
   4. Populate *playfairSquare* with the finished codeword.

d) Write a method `public void printSquare()` that prints the square to the console.

e) We need some help to encode words. Implement the following:
   1. A class *Position* that contains a x-coordinate and a y-coordinate as public final fields.
   2. A method `private Position findInSquare(Character c)` that returns the *Position* of a *Character* within *playfairSquare*.
   3. A method `private String cleanWord(String word)` that cleans the to be encoded word like described. Firstly, strip the word of any special characters. Then build the pairs as described. Lastly, transform the word to upper case.

f) Implement the method `public String encode(String word)` that returns the encoded word. Use your helper methods from the last sub-task.

g) Test your encryption by calling your methods within a main-method.

   Attention: You may use the codeword "Apfelstrudel".

Task 7.2: The Invisible Library

6 Points

You are supposed to implement a DVD-Collection in this task.

a) Firstly, we need to implement a few class structures (make sure that fields have the correct visibility and are `final` if necessary):
   - *Library*
     A *Library* has a name, a date (last time updated) and an Array of *DVD*.
   - *DVD*
     A *DVD* has a title, a ISBN-number, a director and an Array of the most important actors (headliners).
   - *Actor*
     An *Actor* has a name, a surname and a date of birth.
   - *Director*
     A *Director* has a name, a surname and a date of birth.

   Attention: Use *java.util.Date* for dates.

b) Implement useful constructors for every of the above classes.

c) Implement methods to have read access to private fields.

d) Add a method `void addDVD(DVD dvd)` to *Library* that adds a new DVD to your library.

   Furthermore, add a method `void removeDVD(String title)` to *Library* that removes any DVD with the name *title* from your collection. Be aware that you need to change the date of your *Library*.

   Attention: Arrays can not be increased after their initialisation. You need to create another array with room for one more DVD and copy old values over to the new array.

e) Implement a method `boolean doIOwn(DVD dvd)` in *Library* that returns `true` if you a DVD with the same title.

f) Write a main-method in which you:
   - Create a new collection of DVDs
   - Add at least 5 different DVDs to your collection with the *addDVD*-method
   - Remove 2 DVDs from your collection
   - Check if you own one of the removed and one of the note removed DVDs