Software Development I – Exercises **Winter Term 2017/2018**

Übungen zu Softwareentwicklung 1 **Assignment 8**

Name: Daniel Bauer Teaching Assistant: _____

Student ID (Matr.Nr.): 0857178 Points (max. 24): _____

Group: ☑ G1 ☐ G2 ☐ G3 ☐ G4 ☐ G5 ☐ G6 Deadline: **Tue., January 09, 2018 22:00**

Instructor: ☐ M. Haslgrübler ☐ C. Wirth ☑ T. Forstner Editing time (hours): 8

Preferred language for comments, proposals for improvements from TA's: ☑ DE ☐ EN

Problem 1: Message Compressor **12 points**

In order to shrink messages you decided to implement a simple message compression on your own. The compression method you want to implement works by just deleting all the multiples of any character (so only the first occurrence of every character remains in the message).

Your program should read in a message, compress it and output the compressed message. Furthermore, your program should output an occurrence count for all characters and determine the most frequent character. If there are several characters with the same maximum amount of occurrences, only the one with the lowest ASCII number should be printed.

Example

```
Please insert message: Max Mustermann
' ' 1
'M' 2
'a' 2
'e' 1
'm' 1
'n' 2
'r' 1
's' 1
't' 1
'u' 1
'x' 1

Compressed Message: Max ustermn
Character 'M' occured most often with 2 times
```

Problem 2: Polyalphabetic Cipher **12 points**

In this task, you should develop a program to encrypt and decrypt messages. This technique uses a key and applies it cyclically to the original message. The key contains only of capital letters and space characters. The encryption is realized by adding the code value of the key characters to the code values of the characters of the original message. The code values for the key characters are assigned as follows: 1 for A, 2 for B, 3 for C, … 26 for Z and 27 for the space character. Suppose the key is "I LOVE ME" and the original message is "Max Mustermann." Then the result of the encryption looks like this:

Original Message	M	a	x		M	u	s	t	e	r	m	a	n	n
Key	I		L	O	V	E		M	E	I		L	O	V
Encrypted Message	V	|	%	/	c	z	/	"	j	{	)	m	}	%

To decrypt a message the encryption process is reverted.

To develop this program you should create an instantiable class `CipherKey`. This class contains a private String attribute `key`, which has a default value (e.g. “I LOVE ME”). Your class should have the following method:

- `public void setKey(String newKey)`
  This method allows the user to set a new key for encryption. The key is only overwritten if the user passes a valid key as argument. A key is valid if it has at least one character and consists only of capital letters and spaces.
- `public String encrypt(String message)`
  This method encrypts the parameter and returns the encrypted message. If the message is not valid, the method should return `null`. The message is valid, if all containing characters lie between 32 and 126. To encrypt the message the code values (e.g. A=1, B=2, …,Z=26 and ‘ ‘=27) of the key are cyclically added to the ASCII values of the characters of the original message. All ASCII values of the encrypted message are between 32 and 126. If the calculated new character lies outside of the ranges (<32 or >126) it has to be corrected (e.g. set code value 127 to 32).
- `public String decrypt(String message)`
  This method decrypts the parameter and returns the decrypted message. Use the same approach as you used in encrypting a message, only subtract instead of add the code values of the key.

Finally, create a test program, which instantiate your `CipherKey` class and uses a menu. This menu allows the user to encrypt as well as decrypt messages and set new keys.

No more static methods or fields allowed – except main function and constants

Example

```
1) Change key
2) Encrypt
3) Decrypt
0) Quit
2
Enter the message: Max Mustermann
Encrypted Message: V|%/cz/"j{)m}%
1) Change key
2) Encrypt
3) Decrypt
0) Quit
3
Enter the message: V|%/cz/"j{)m}%
Decrypted Message: Max Mustermann
1) Change key
2) Encrypt
3) Decrypt
0) Quit
0
```

Requested material for all programming problems:

For each exercise, hand in the following:

a) Approach to solving the problem (textual representation)

b) Source code (Java classes ideally with English(!) comments); in addition to the source code Java source files have to be converted to PDF and are to be included in the ZIP file,

c) Test plan for analyzing boundary values (e.g., minimal value allowed, maximum number of input, etc.) and exceptional cases (e.g., textual input when a number is required, etc.). State the expected behavior of the program for each input and make sure there is no "undefined" behavior leading to runtime exceptions. List all your test cases in a table (test case #, description, user input, expected (return) values).

d) The output of your java program for all test cases in your test plan

Pay attention to using adequate and reasonable data types and meaningful English variable names for your implementation, check the user input carefully and print out meaningful error messages.