Object Orientation – Spring 2018 **Assignment 2**

Learning objectives

After making this assignment, you should be able to

- design and use Java classes and objects properly, especially using encapsulation and model-view separation;
- split a program into a model (the logic) and a view (input-output);
- work with the classes `String` and `StringBuilder` and know their difference.

1 Hangman with classes and objects

In this exercise, you and your partner make your very own implementation of the popular game 'Hangman' to reach the objectives.

The game

For those unfamiliar with the game, we will explain it here briefly. The goal of the game is for the user to guess the word that the other player (or, in our case, the computer) has in mind. As a hint, the user is shown a line of dots, where each dot represents a letter of the word that needs to be guessed (e.g. `........`). The user then guesses a letter. If the letter is in the word, all dots at the positions of that letter are replaced by the letter (e.g. `.a...aa.`). If the letter is not in the word, the other player builds a part of the virtual gallows. This continues until either the player has made enough mistakes to 'complete' the gallows or the player mentioned all letters of the word. The amount of mistakes that can be made differ per version of the game, but a maximum of 10 is a commonly used value.

Producing words for the game

To help you a bit, we have put some files on BlackBoard. Later in the course you will be able to write such a class yourself, but for the moment it contains too many nasty Java details.

First there is the class `WordReader`. When you create an object of this class, it will read all words from a given file. When creating an object, you can specify which file by passing the name of the file to the constructor as a string. The other file is `"words.txt"`, which contains several example words. The method `giveWord` from the class `WordReader` will return a pseudo-random word from the specified file. The file containing the words should be in the root folder of your Java package. You can add or delete words from the word file as you wish.

String and StringBuilder

The word that needs to be guessed and the word as it has been guessed so far need to be remembered. The basic class to store the sequences of characters is `String` in Java. The objects of the `String` class are *immutable*, there is no way to change the content of such an object.

We can make a new `String` object each time we need a slightly different string, but sometimes this is inconvenient. Java has the class `StringBuilder` for mutable versions of strings. See `http://docs.oracle.com/javase/8/docs/api/` or the help of NetBeans for an description of the available methods.

Encapsulation

One of the key design principles in object orientation is *encapsulation*; the bundeling of data inside an object with the methods to manipulate this data. Make sure that all mutable attributes in your classes are `private`. Define getters to make this data available and setters for the attributes that should be mutable.

Model-View

An other key design principle in programming is the separation of concerns. One of the places where this principle is applied is the separation of the stores and logic of your program and the input-output of your program. The input-output, often called *view*, should be separated as much as possible from the logic and stores, the *model*. In this assignment it should be possible to a completely different user interaction without changing the rest of the program.

Different ways to use the model

To make the game a little more interesting, we want to provide two ways to start a new game. One way is to let the word finder come up with a new word. The other is to enter a word yourself. To accommodate this, you will need at least two constructors of the model class. If this class is called `Gallows`, it has the constructors `Gallows (String s)` and `Gallows ()`. The first one uses a word that was provided by the user and the second one goes to the `WordReader` class to get a word there.

Handing in

Hand in your assignment **before Sunday, February 18th, 23:59**. As always, your code should adhere to the layout rules as mentioned on Blackboard. Don't forget your and your partner's names and student numbers!