Prof. Dr.-Ing. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Assignment 08

Submission of solutions until 3:00 p.m. at 05.01.2022
at `moodle.uni-trier.de`

- Every task needs to be edited in a meaningful way!
- Please comment your solutions, so that we can easy understand your ideas!
- If you have questions about programming or the homeworks, just ask you teachers!

Exercise 1 (Book Class, Evaluation: predefined `main`)

Create a class `Book` with the following structure:

- Every book has as instance variables `author`, `title` (both as `String`), a `year of publication`, a `page number` (both as **`int`**) and of course a `content` (again as `String`).
- Write a constructor that sets the first four values at once (in the above order). In addition, the `content` is preset with the empty string.
- The content can then be written with a method `append(String str)` and extended later.
- The class also contains a class variable `String publishing` that can be set with a static `setPublishing` method.
- All variables are supposed to be **`private`**
- Write a `toString` method that wraps all of these values except the content in a string following format:

  ```
  The book "xx" with "xx" pages by the author "xx"has been
  published by the publishing company "xx"in the year xx.
  ```

- An object method `string quote( int start,int length)` should return exactly `length` characters, starting at position `start`. If the book has fewer characters than necessary, the return value should be correspondingly shorter.

A suitable test class with a `main` method is specified for the evaluation.

Exercise 2 (Dice and Mia, Evaluation: predefined `main`)

First implement a class `Dice` with the following methods analogous to the coin toss (example `K5B01E_CoinToss` from the lecture):

- A method `throwDice()` randomly assigns a new number (equally distributed) between 1 and 6 to the dice.
- A method `pips()` should return the value thrown by the dice as **`int`**.

  The function `pips()` of a finished `Dice` object must return **always** a number between 1 and 6!
- Both **`throw`**`()` and `pips()` should be accessible for users.
- In addition implement a function in `Dice`

  `public static int` `pipSum(Dice d1, Dice d2),`

  that returns the total value of both dice.

Write another class `Mia` that internally stores two `Dice` objects. These should be created as follows in the constructor of `Mia`

```
1  public class Mia {
2          private Dice d1, d2;
3          public Mia() {
4                  d1 = new Dice();
5                  d2 = new Dice();
6          }
7  }
```

The class should also provide the following functions:

- **`public int`** `valueAndThrowDice()` interprets the current numbers of the two dice as numbers according to Mia rules[1]. In addition, the function should throw both dice again, and then return the interpreted value *of the old throw*.

[1] See `https://en.wikipedia.org/wiki/Mia_(game)`

- **public static boolean** `isMia(int value)` checks whether `value` is the maximum "'Mia"' result.
- **public static boolean** `isDouble(int value)` tests `value` to see if it's a doublet.
- **public static boolean** `isLess(int a, int b)` compares the two values `a` and `b` according to Mia rules and returns **true** exactly when `a` represents a smaller value than `b` in this sense.

  Hint: Implement this method by using `isMia(int value)` and `isDouble(int value)`

Watch out: The evaluation only shows you whether you have completed all subtasks; it does not help you to be correct!

Exercise 3 (Evaluation: predefined `main`)

Write a class `UserManagement` to simulate the management of user names in a computer, in particular password management.

The class shall contain two *private* arrays `id` and `pw` of 100 strings each to store the data.

The following methods are to be implemented:

- A constructor `UserManagement()`
  Here the fields `id` and `pw` are associated. Pay attention to what initials values the individual components of `id[i]` are set!
- **boolean** `newUser(String name, String password)`
  This can be used to create a new user with the specified pair of name and initial password.

  If the user already exists, no change should be made in the data, but the value `false` should be returned. The same applies to the case that more than 100 users should exist at the same time together with the new user or that the username consists of less than four characters. Otherwise the user will be saved accordingly and `true` will be returned. The password may be any non-empty string here.
- **boolean** `checkPassword(String name, String password)`
  This checks if the specified `password` matches the user `name`.
- **boolean** `changePassword(String name, String oldPw, String newPw)`
  This can be used to change the user's `name` password if `oldPw` was correct and the new password `newPw` has a length of at least eight characters.

  If the change is successful, the value `true` is returned, otherwise `false`.

- There is an administrator password that applies to all objects of this class. This password can be set once by **`void`** `setAdmin( String adminPw)`, again requiring at least eight characters.

  Any attempt to change a previously set administrator password should be logged with a warning `"unauthorized access "` on the console.

  Attempts to set an unset administrator password to a password that is too short, on the other hand, are completely ignored.

- The administrator password can be used in objects of the class `UserManagement`. Passwords can be set to any non-empty values, using the following method:
  `void` `setPassword( String adminPw,`
  `String name, String newPw)`

You have to think for yourself where to put the modifiers `public`, `private` or `static` (matching the given `test` class). For the sake of simplicity, deletion of users is not foreseen.

Attention: An associated class `test` with some calls to test the user management is given. However, this is supposed to be replaced by another class when correcting your solution, which among other things will try to crash your solution (which you should avoid...)