Prof. Dr.-Ing. Ralf Schenkel
Martin Blum
Trier University

Wintersemester 2024/2025

Exercises for the Class

Elements of Computer Science: Programming

Assignment 9

Submission of solutions until 16.01.2025 at 10:00
at `moodle.uni-trier.de`

- Every task needs to be edited in a meaningful way in order to get a point!
- Please comment your solutions, so that we can easily understand your ideas!
- If you have questions about programming or the assignments, just ask you teachers!
- **Submission that can't be compiled are graded with 0 points!**

Exercise 1 (Evaluation: predefined `main` method)

Familiarize yourself with the methods of the `String` class[1]:

1. Create a method

   ```
   static int countOccurrence(String haystack, String needle)
   ```

   that counts at how many positions the string `needle` occurs as a substring of `haystack` For example, the call `countOccurrence("abbbba", "bb")` should return the value 3.

2. Create another method

   ```
   static String extractTag(String s)
   ```

   with the following property: For an argument `s` of the form

   prefix '`[`' *infix* '`]`' *suffix*

   the method `extractTag` should return the substring *Infix*. You can assume that '`[`' and '`]`' both occur exactly once in the correct order in `s`.

   For `extractTag("1234[5678]9")` the string `5678` should be found and returned.

Exercise 2 (Evaluation: Predefined `main` method)

This task deals with two-dimensional (**`double`**) arrays and how they can be utilized as matrices. You should implement two methods that make it possible to multiply two matrices [2] together:

[1] `https://docs.oracle.com/javase/8/docs/api/java/lang/String.html`
[2] `https://en.wikipedia.org/wiki/Matrix_multiplication`

```
boolean isValid(double[][] matrix1, double[][] matrix2)
double[][] mulMatrices(double[][] matrix1, double[][] matrix2)
```

The method `isValid()` is used to determine if a matrix multiplication is possible. Remember, that you can only multyply two matrices if the number of collumns of `matrix1` is equal to the number of rows of `matrix2`.

The method `mulMatrices()` is used to implement the actual matrix multiplication. It returns a two-dimensional **`double`** array, containing the result of the multiplication. In case that both matrices cannot be multiplied because `isValid()` returns **`false`**, simply return the **`null`** reference.

Exercise 3 (Evaluation: predefined `main` method)

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`**. It must always return a number between 1 and 6!
- Both `throwDice()` and `pips()` should be accessible for users.
- In addition, implement a method 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 methods:

- **`public int`** `valueAndThrowDice()` interprets the current values of the two dice according to Mia rules[3]. In addition, the method should throw both dice again, and then return the interpreted value *of the old throw*.
- **`public static boolean`** `isMia(int value)` checks whether `value` is the maximum Mia result.

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

- **public static boolean** `isDouble(int value)` tests `value` to see if it is 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 check if your program's logic is correct!

Exercise 4 (Evaluation: predefined `main` method)

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()`
  `id` and `pw` are associated. Pay attention to what initial values the individual array components `id[i]` and `pw[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 and the value `false` should be returned. The same applies to the case that more than 100 users would exist at the same time together with the new user or that the name 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 `name`'s password if `oldPw` is 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.

  On the other hand, attempts to set an unset administrator password to a value that is too short should be ignored completely.

- The administrator password can be used in objects of the class `UserManagement`. Passwords can be set to any non-empty value, 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 intended.