Prof. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 23/24

Exercises for the Class

Elements of Computer Science: Programming

Live Assignment 4

Submission of solutions for group 1 until 2:00 p.m. and for group 2 until 3:00 p.m.
at `moodle.uni-trier.de`

- Submission that can't be compiled are rated with **0** points!
- Please comment your solutions, otherwise you can lose points!
- If you try to cheat, you will lose your points and the classroom exercise will be over!

Exercise 1 (Evaluation: Numbers) (10 Points)

Create a method

```
static int minSum(int[][] a)
```

that has a 2-dimensional array as parameter and returns a **`int`** value. The method should return the line (i.e. the index value) with the lowest sum over all components in the passed array `a` In the example below the return value would be 0 because the 0th line has the lowest sum.
In case that more than one row has the lowest sum return only the lowest index.

Example:

$$\begin{array}{c|c} \mathbf{0} & \begin{pmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ 28 & 22 & 17 & 19 \\ 18 & 27 & 20 & 43 \\ 10 & 10 & 11 & 18 \end{pmatrix} \\ 1 & \\ 2 & \\ 3 & \end{array} \Rightarrow \begin{array}{c|c} \mathbf{1+2+3+4} & \mathbf{10} \\ 28+22+17+19 & 86 \\ 18+27+20+43 & 108 \\ 10+10+11+18 & 49 \end{array}$$

Exercise 2 (Evaluation: predefined `main` method) (10 Points)

Implement the *digit sum* as **recursive method** (for numbers $n \geq 0$; this condition does not need to be tested):

```
static int ds(int n)
```

The cross sum $\texttt{ds}(n)$ can be calculated as follows:

1. For numbers n with $0 \leq n < 10$, $\texttt{ds}(n) = n$.
2. For numbers n with $n \geq 10$ is $\texttt{ds}(n) = n\%10 + \texttt{ds}(n/10)$.

Non-recursive solutions do not correspond to the task and are therefore completely wrong!

Exercise 3 (Evaluation: predefined `main` method) (15 Points)

Create the following methods

- `static int addDigits(String str)`
- `static boolean containsSum(String str).`

The method `addDigits(String str)` should add up all digits that appear in `str`. Then the method `containsSum(string str)` should check if the sum determined with the method `addDigits(string str)` is contained in `str`.
The output looks as follows:

```
Input: a13bc9dgc
Output: The value 13 is contained in the string

Input: a1b2c3d4e
Output: The value 10 is not contained in the string
```

Hint: Use the method `charAt(index)` provided by object of type `String` to iterate over the characters in `str`. You can use the method `Character.isDigit(char)` to check if `char` is a digit.

Exercise 4 (Evaluation: predefined `main` method) (15 Points)

Create a class named `Product` that stores the name of a product (`String name`), the brand (`String brandName`) and its price (`double price`). Additionally, implement the following aspects:

- Implement a constructor that takes as input the name, brand and price of the products and sets the corresponding information to the given values.

- Implement a second constructor that takes as input the name and the price of a product. Again, set the corresponding information to the given values. The brand name (`brandName`) should be set to "`Unknown`".
- Implement all needed getter methods.
- Implement the following method

  ```
  Product search(String productName, Product[] products)
  ```

  that searchs in the the given product array (`Product[]`) for the first product with the same(!) name as `productName`. In case no product with the name `productName` is found, return the **`null`** reference. Think about if this method needs to be **`static`** or not!