Prof. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Live Assignment 03

Submission of solutions until 6:00 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!

Exercise 1 (Evaluation: Numbers/Text) (10 Points)

Your program should first read in a `int` number a (where you can assume that $a > 0$ is). Then it should preset a `double` variable x with the value 2 and then with a `for` loop ten times (a) first output the current value of x and (b) then redefine x each time via the formula $x = (x + a/x)/2$ (similar to the Lunar Land Game without protocol). Exactly 10 `double` numbers are printed.

Example:

```
Input: 16
Output: 2.0 5.0 4.1 4.001219512195122 4.0000001858445895
4.000000000000004 4.0 4.0 4.0 4.0
```

Note: This method is the so-called "Heron" method for calculating the square root.

Exercise 2 (Evaluation: Numbers) (10 Points)

The given progam reads in two arrays. In this task the arrays should be interpreted as vectors. Your task is to determine the scalar product of both vectors. Use the following formula:

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \text{ and } \vec{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \text{ lead to } \vec{a} \circ \vec{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$$

An example of the application of the scalar product is:

$$\vec{a} = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix} \text{ and } \vec{b} = \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix} \text{ lead to } \vec{a} \circ \vec{b} = 2 \times 10 + 3 \times 20 + 4 \times 30 = 200$$

You can assume, that both read in arrays have the same size.

Exercise 3 (Evaluation: Text) (15 Points)

Write a program that reads a **`int`** number as input and generates a pattern (from $n \times n$ characters) of the following type as output (similar to the example `K3B14E_Flag`):

```
Input: 6
XXXXXX
XXXXXo
XXXXoo
XXXooo
XXoooo
Xooooo
```

```
Input: 7
XXXXXXX
XXXXXXo
XXXXXoo
XXXXooo
XXXoooo
XXooooo
Xoooooo
```

Also make sure that no line ends are missing and that you do not create unnecessary blank lines, otherwise the evaluation will not work. The letters used in the pattern are the capital `X` and the lower `o`.

Exercise 4 (No Evaluation) (15 Points)

Use your program to find the smallest positive `int` number n that is so large that the value $n \cdot n$ can no longer be represented as a `int` value (called overflow).

Hint: Which values would $(n \cdot n)/n$ and $(n \cdot n)\%n$ have to have, if calculated correctly? An input is not needed, the output should have the following form:

```
First overflow during squaring occurs at 50000
```

Watch out: The value **50000** is not the correct result. *During the exercise* there is no correct value for the evaluation given, i.e. you cannot test your program for correctness by the evaluation.

Prof. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Live Assignment 04

Submission of solutions until 6:00 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!

Exercise 1 (Evaluation: predefined `main` method) (10 Points)

Implement a method

```
public static String rearrange(String word, int[] sequence)
```

that is able to rearrange the characters in a given word `word` according to the given index values in the sequence array `sequence`. In case `sequence` contains numbers < 0 or $>= word.length()$ return the String "Invalid Sequence".

Examples:

Word: juiceApple

Sequence: 5 6 7 8 9 0 1 2 3 4

Result: Applejuice

Word: juiceApple

Sequence: 5 6 7 8 9 0 1 2 3 10

Result: Invalid Sequence (since the highest index in the given word is 9)

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

Implement a **recursive method**

```
public static String reverse(String string)
```

that is able to reverse the String (`string`). For example the call `reverse("1234")` should lead to `"4321"` as output. Note that it is important that you implement the method in a recursive way. Other solutions are not accepted.

Hint: Use the methods charAt() and substring() of the `String` class.

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

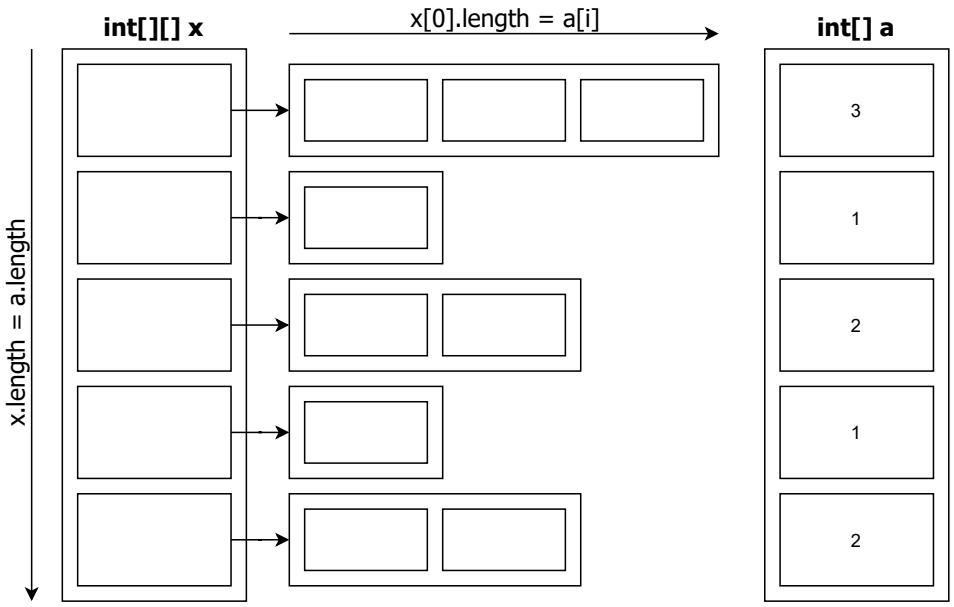
Implement the method

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

A call of the form `int[][] x = createArray(...)` shall generate in array `x` a two-dimensional array. The size of the "first" dimension is defined by `a.length`. The size of each individual array, that will be contained in each entry of x (e.g. x[i] for $i = 0, 1, ...,$`a.length`), is defined by a[i].

Hint: The concrete contents of `m` are unimportant, it is here only about the allocation in correct form and size.

Visual Example:



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

Implement the method

```
static int countUniqueWords(String s)
```

The method receives as String parameter `s` a sentence and shall count the contained unique words. Words in a String are always split by a blank space. The method should be case insensitive, which means `"this"` and `"This"` are the same word.

Example:
`countUniqueWords("This␣is␣an␣example")` ⇝ 4

	0	1	2	3
words	*This*	*is*	*an*	*example*

`countUniqueWords("This␣this␣is␣is␣a␣Test")` ⇝ 4

	0	1	2	3	4	5
words	*This*	*this*	*is*	*is*	*a*	*Test*

`countUniqueWords("Computer␣Science␣is␣Is␣iS␣fun")` ⇝ 4

	0	1	2	3	4	5
words	*Computer*	*Science*	*is*	*Is*	*iS*	*fun*

Prof. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 23/24

Exercises for the Class

Elements of Computer Science: Programming

Live Assignment 5

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: predefined `main` method) (10 Points)

Given is the class `Product`, which stores the name (`String name`), a product category (`String category`) and the price (`double price`) of a product. All instance variables of the class are **`public`**, which is why getter methods are not necessary.

In the class `Evaluation`, implement the method

```
double categoryValue(String category)
```

which calculates and returns the value of all products with the category `category`.

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

Implement the class `Restaurant`. An object of this class stores the name (`String name`) and a rating (`double rating`) of a restaurant. All instance variables of the class should only be accessable to the class itself. Implement next a constructor of the following form:

```
public Restaurant(String name, double rating)
```

Make sure that, as usual, the rating is in the interval between 0 and 5 inclusive. If a value is less than 0, set the rating to 0 and if the rating is greater than 5, set the rating to 5. Furthermore, implement all required getter methods.

Next, implement the following method in the `RestaurantFinder` class:

```
public Restaurant findBest(Restaurant[] restaurants)
```

The aim of the method is to find the best restaurant from the restaurant array. The best restaurant is the restaurant with the highest rating.

Exercise 3 (Evaluation: predefined `main` method) (10 Points)

Your task is to implement a class `ArrayDictionary` and `DictionaryElement`.

A `DictionaryElement` stores an `Object key` and an `Object value`. These variables should only be accessable by the class itself. Therefore, implement all necessary getter methods of the class.

In addition, you have to implement a constructor

```
public DictionaryElement(Object key, Object value)
```

that initialises both parameters accordingly.

The class `ArrayDictionary` stores an array `DictionaryElement[] dictionary`. The size of the array is specified via the constructor and the dictionary is initialised accordingly in the constructor. Additionally, implement the methods of the interface as follows:

- **`boolean`** `add(Object key, Object value)`:
  The methods searchs in the `dictionary` for the next free position and then stores a `DictionaryElement` there with the values `key` and `value`. If an entry with the `key` already exists, overwrite the saved value `vlaue` with the new value.

  If a `DictionaryElement` is stored successfully, **`true`** should be returned. If the array is already full, **`false`** should be returned.

- `Object get(Object key)`:
  The method searches in the `dictionary` array for an `DictionaryElement` with the same `key`. If no `DictionaryElement` with the searched `key` exists, return the **`null`** reference.

Exercise 4 (Evaluation: predefined `main` method) (10 Points)

Given is the class `Product` which stores an ID (**`int`** `id`), the name (`String name`) and the price (**`double`** `price`) of a product.

Implement next in the class `ProductParser` a method

```
Product parse(String productString).
```

The method is passed a string containing information about the ID, the name and the price. Examples of such strings can be found in the following example:

```
12345;Smartphone;399.99
12;Smartphone;399.99
```

```
;Smartphone;399.99
12345;;399.99
;;399.99
```

The aim of the method is to process the given string and create and return an object of the class `Product`.

The individual pieces of information are always separated by a semicolon (`;`) and always in the same order. This means that the ID is given first, followed by the product name and finally the price. However, as you can see from the examples, it can happen that an ID is incorrect (i.e. shorter than 5 characters and simply does not exist). In this case, the ID should be set to -1. It can also happen that the product name is missing. In this case, you should set the product name to "Unknown".

Prof. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Live Assignment 06

Submission of solutions until 6:00 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!

Exercise 1 (Evaluation: predefined `main` method) (15 Points)

Implement an abstract class `Person` that can store two pieces of information: The first and last name of a person (each as an `String`). Additionally, implement a suitable constructor.

The class consists of a method **`public`** `String toString()` and shall return the first and last name of a person.

The class should also contain two abstract methods:

- **`void`** `learn(String newWord)`
- **`int`** `getNumberOfWords()`

Derive from the `Person` class the two subclasses `PersonA` and `PersonB`. Implement appropriate constructors as used in the `Evaluation` class.

Since `PersonA` and `PersonB` in this example learn words of a new language in different ways, the method `learn(...)` must be implemented for both classes. It is important that in the class `PersonA` the learned words are stored in an array (initialized with 100 fields). In the class `PersonB` all learned words are stored in a `List<String>`.

Since `PersonA` and `PersonB` use different data structures to store the words, the method `getNumberOfWords()` must also be implemented separately for each class.

Implement appropriate constructors for the classes `PersonA` and `PersonB`

Exercise 2 (Evaluation: predefined test class `Test`) (15 Points)

Given is the class `PhoneBookEntry`, which stores a name (`String name`) and a phone number (`String number`). Your taks is to create a `PhoneBook` class that contains only **`static`** class members:

- A `PhoneBook` stores a `List` contianing objects of type `PhoneBookEntry`.
- Implement a method **`void`** `change(name,number)`, which can change an entry in the phone book (if already contained by the book). That means in case that the name is not known to the phonebook (in this example names are unique) create a new entry containing the name and the phone number. But if the name is known to the phone book you have to replace the old number with the new one.
- Implement a method `String searchNumber(name)` that returns the phone number for a name. If the search method searches for a name that is still unknown, the string `unknown` should be returned.

In order to keep the solution simple, you can assume that no **`null`** references occur as parameters. **You can change the test class but for the evaluation we will use the original version.**

Exercise 3 (Evaluation: predefined `main` method) (10 Points)

First a string is read in the specified program. Then an attempt is made to convert it into an **`double`** number and then output. During the conversion, however, depending on the string, conversion errors may occur.

Modify the program by using exceptions in such a way that the input is repeated so often (without producing further output), until for the first time a string could be converted into a number without errors.

Note: Use exception handling, e.g. **`try`**–**`catch`** statements.

Exercise 4 (Evaluation: predefined `main` method) (10 Points)

In the given files an interface `myDataInterface` is defined, in which there are two methods: One method `init(int array)` is to copy(!) the data of an **`int`** array to instances of the interface, a second method `at(int n)` is to read and return the value at the position `n`.

Write an appropriate class `myData` (with matching constructor) that implements this interface. You must decide yourself how to store the data transferred during initialization.

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!

Prof. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Live Assignment 05

Submission of solutions until 6:00 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!

Exercise 1 (Evaluation: predefined test class `Test`) (10 Points)

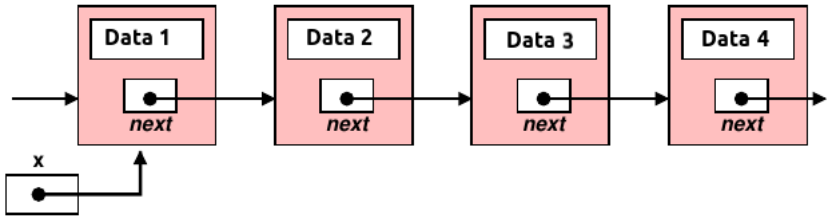
Consider the class `Queue` (see example `K5B05E_Queue_Linked` of the lecture). Extend this class with a method **`public`** `Object peekFirst()`, which returns the first data object of a queue, and a method **`public`** `Object peekLast()`, which returns the last data object of a queue.

Both methods should not change the queue. If the queue is empty, the **`null`** reference is to be returned.

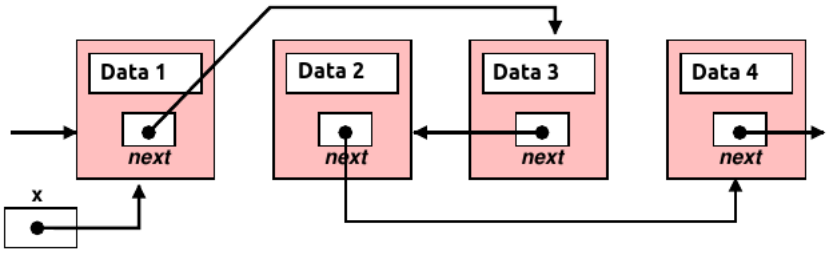
When editing, you have access to `Queue.java`, `Test.java` and `Elem.java`, where only `Queue.java` is to be edited by you. For the correction we will use the original version of the other classes.

Exercise 2 (Evaluation: predefined test class `Test`) (15 Points)

Using the `Elem` class for single linked lists, write a method **`void`** `modify(Elem x)` that swaps the order of the entries `x.getNext()` and `x.getNext().getNext()` in the concatenation (see visualization below).



After calling `modify(x)` the concatenation should look like this:



For the sake of simplicity, you can assume that there are no **`null`** references in the area you want to modify. Swapping the references to the data components does not count as a solution (and is also recognized as incorrect by the test routine...).

Exercise 3 (Evaluation: predefined test class `Test`) (10 Points)

Given is a test class `Test` and an interface `Dictionary`. The interface dictates that any class implementing the interface must implement the following methods:

- **`public void`** `add(String key, String value);`
- **`public`** `String get(String key);`

Your task now is to implement a class `ArrayDictionary` and `DictionaryElement`.

A `DictionaryElement` stores a `String key` and a `String value`. These variables should only be accessed by the `DictionaryElement` class itself. Therefore, implement all necessary getter methods of the class.

In addition, you have to implement a constructor

`public` `DictionaryElement(String key, String value)`

that initialises both parameters accordingly.

The class `ArrayDictionary` shall implement the interface and store an array named `dictionary` (of type `DictionaryElement[]`). The size of the dictionary is specified via the constructor and the dictionary is initialised accordingly in the constructor. Additionally, implement the methods of the interface as follows:

- **`void`** `add(String key, String value):`
  The method searches in `dictionary` the next free entry/position and then stores a new `DictionaryElement` there with the values `key` and `value`. If the array is already full or there is already an entry with the same value for `key`, nothing should happen.
- `String get(String key):`
  The method searches in the `dictionary` array for an `DictionaryElement` with the same `key`. If no `DictionaryElement` with the searched `key` exists, return the **`null`** reference.

Exercise 4 (Evaluation: predefined test class `Test`) (15 Points)

Given are a test class `Test` and the class `Sample`. An object of type `Sample` stores the name of a biological sample (`String name`), when this sample was taken (`String date`) and whether it was contaminated (**`boolean`** `tainted`). Since the variables of the class `Sample` are public, there are no getter and setter methods.

First implement the abstract class `SampleSet`. This class should have a global array `Sample[] samples`. Furthermore, implement a method

```
void addSample(Sample sample)
```

such that at the first free position in the array `samples` the sample `sample` is stored. If the array is already full, nothing should be stored. Additionally, declare the abstract method **`boolean`** `test()` for the class `SampleSet`.

Next, implement another class called `FastSampleSet`. This class should extend the class `SampleSet`. Next, you have to implement a constructor

```
public FastSampleSet(int size)
```

that initialises the Array `Sample[] samples`.

Your next step is to implement the `test()` method of the `FastSampleSet` class as follows: The method performs an experiment with the samples stored in `Sample[] samples` and outputs as **`boolean`** whether the experiment was successful. For this, the first step is to test whether no sample is contaminated (`tainted`). If a sample in the array is contaminated, the test was not successful and **`false`** is returned. This implementation of the experiment has the drawback that, if one sample in the array is contaminated, unfortunately all other samples will also be contaminated. In this case the method `test()` must change the status **`boolean`** `tainted` of each sample in `Sample[] samples` to `tainted =` **`true`**. If no sample is tainted, the experiment was successful and `test()` returns **`true`** as output.