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)

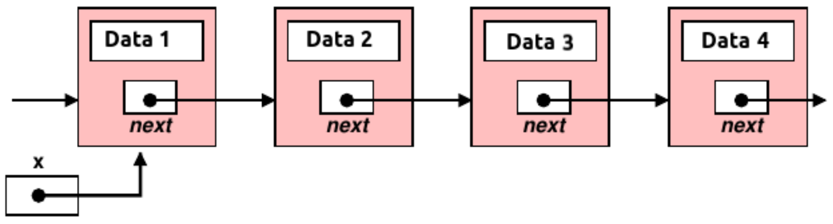
Consider the class `Queue` (see example `K5B05E_Queue_Linked` of the lecture). Extend this class with a method **`public`** `Object peekFirst()`, which returns the first data object of a queue, and a method **`public`** `Object peekLast()`, which returns the last data object of a queue.

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

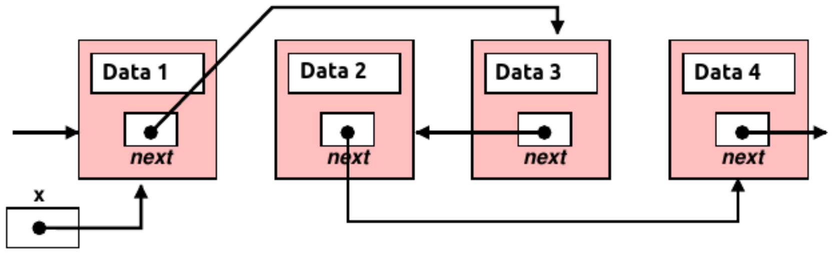
When editing, you have access to `Queue.java`, `Test.java` and `Elem.java`, where only `Queue.java` is to be edited by you. For the correction we will use the original version of the other classes.

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.