Prof. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 23/24

Exercises for the Class

Elements of Computer Science: Programming

Live Assignment 6

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) (5 Points)

In the given class `Main` an interface `myShape` is used but not defined. Add this missing interface to the given class.

Exercise 2 (Evaluation: predefined `main` method) (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.

Exercise 3 (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 4 (Evaluation: vorgegebene `main`-Methode) (15 Points)

In the following task, you are to record the best times of athletes in a map (denoted as `Map<String,Integer> records`). It is important that whenever an athlete has run a new best time (i.e. the new time is lower than the one saved in the map), this is entered in the map. This means that the map always saves the athletes' personal best times. The given program reads strings in the following form:

```
Peter:120
Peter:110
```

```
Katja:115
Peter:111
exit // this is a keyword to abort the user input
```

Implement the method

```
addTime(String line, Map<String,Integer> records).
```

The method has as input a `String` (this is a single line from the example above, e.g. "Peter:120") and a `Map`. As mentioned, the map stores the personal best time for each athlete. For the example above, the content of such a map looks like this:

```
Peter: 110
Katja: 115
```

To create such a map, you must follow these steps: (1) You must split the string `line` into its two components (the name of a person and the time). (2) Then you must check whether there is already an entry for the person in the map. (3) If this is the case, you have to compare the stored value with the newly entered value. (4) If the newly entered value is less than the stored value, the old value have to be updated. In case the newly entered value is higher, nothing happens. (5) If no entry for the person in the map you have to store the name of the person with the given value inside the map.

Attention: The map is input as a reference. This means that if you change the content of the map inside of the method `addTime()`, the content of the map will also be changed outside of this method, e.g, in the `main` method. For this reason, the method does not require a return statement.