© Manfred Kerber
Alexandros Evangelidis

School of Computer Science
University of Birmingham

Worksheet 4

MSc/ICY Software Workshop

Assessed Exercise: 9% of the continuous assessment mark.

Submission: Thursday 17 November 2016 2pm

5% late submission penalty within the first 24 hours. No submission after 24 hours.

JavaDoc comments are mandatory. Follow the submission guidelines on
`https://canvas.bham.ac.uk/files/3242907`.

The public tests will be provided a week before the submission deadline.

Note that for exercises 1-4 you have also to provide your own JUnit tests. MSc students have to do all five exercises, ICY students exercises 1 – 4.

Exercise 1: (Basic, MSc: 20%, ICY: 25%)

(a) Implement an interface `Measurable` with the public method signature for a method `double getMeasure()`. Furthermore, provide implementations for the method in two classes `Invoice` (with field variables `private String accountNumber`, `private String sortCode`, and `private double amount`) and `Patient` (with field variables `private String name`, `private int age`, and `private double weight`). Your implementations may be minimal so that they just provide implementations to the `Measurable` interface so that `getMeasure()` returns the `amount` and the `weight`, respectively.

(b) Write a class `Statistics` with two static methods:

- `public static double mean(ArrayList<Measurable> list)` and
- `public static double standardDeviation(ArrayList<Measurable> list)`

that compute the mean value and the standard deviation of a non-empty ArrayList of elements.

The mean value μ (or arithmetic average) is computed by summing up the elements and dividing by the number of elements. The standard deviation σ is defined as the square root of the average quadratic difference of the elements from the mean value. In formulae this is for n elements $a_1, \ldots, a_n$ (with $\sum$ standing for sum, corresponding to a `for` loop):

$$\mu = \frac{1}{n}\sum_{i=i}^{n} a_i$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \mu)^2}$$

Exercise 2: (Basic, MSc: 20%, ICY: 25%)

(a) Implement a `Vehicle` class with the field variables `private double maxSpeed` and `private int maxPassengers`, with a constructor, all getters and setters as well as a `public String toString()` method which returns a String of the kind:
```
"The vehicle has a maximal speed of 24.0 km/h.
It carries 1 passenger."
```
(Note that after a `1` there should follow a singular, for any other value of `maxPassengers` it should be a plural, i.e., `"passengers."`)

(b) Write a sub-class `Car` of the class `Vehicle` with the additional field variable `private double fuelConsumption`. Again write a constructor, all getters and setters as well as a `public String toString()` method which returns a String of the kind:
```
"The vehicle has a maximal speed of 24.0 km/h.
It carries 2 passengers.
Its fuel consumption is 6.3 l/100km."
```
Make use of inheritance as much as possible.

Exercise 3: (Medium, MSc: 20%, ICY: 25%)

Assume you want to build a web crawler (an essential part in a web search engine), that is, an engine that starts with a particular web page and then collects web pages which can be reached from this start page by first collecting all web pages that can be reached in a single step from the page. The first step is to collect all these web pages. Write a class `WebCrawler` that contains a static method `public static ArrayList<String> collectUrls(String urlString)` which reads the page related to `urlString` and returns an ArrayList of all the URLs that can be found on the corresponding page `urlString` (an empty list if the original URL cannot be reached). [You may write auxiliary methods.]

As a starting point look at the `Html.java` class from the Wednesday lecture in Week 4. In order to find URLs assume that they all have the structure
`<a href="http://www.cs.bham.ac.uk">Computer Science</a>`.

From such a String you should extract the String `"http://www.cs.bham.ac.uk"`. In order to find the URLs you can make use of the method `public String[] split(String regex)`, see `https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split-java.lang.String-` for details.

We provide a test file at `http://cs.bham.ac.uk/~mmk/teaching/java/test.html` for which the method should return an `ArrayList<String>` corresponding to:
`["http://www.cs.bham.ac.uk/", "http://www.eps.bham.ac.uk/", "http://www.bham.ac.uk", "https://canvas.bham.ac.uk/courses/21955"]`.

[A full web crawler would remove duplicates and then continue with the extraction of further URLs from these newly found web pages. We are not looking into this here.]

Exercise 4: (Advanced, MSc: 20%, ICY: 25%)

(a) Write a class `ExamQuestion` with field variables `private String questionText` and `private int maximalMark`. Write a suitable constructor, getters/setters for the two field variables, and a `public String toString()` method which return for a question with maximally 10 marks and a `questionText` of `"What is 2 times 3?"` the string `"Question (maximal mark: 10): What is 2 times 3?"`.

(b) Write a subclass `ExamQuestionNumeric` of class `ExamQuestion`, in which the answer is supposed to be a numerical answer of type `int` (you have to represent the correct answer).

Write a method `public int mark(int value)` in the subclass that returns full marks if the answer is correct and `0` otherwise. Furthermore write a suitable `public String toString()` method, making use of inheritance as far as possible.

E.g. assume a question:
`ExamQuestionNumeric q1 = new ExamQuestionNumeric("2+3 = ?", 10, 5);`

where `10` is the maximal number of marks and `5` the correct answer.

`q1.mark(5)` should return `10`, whereas `q1.mark(6)` should return `0`.

(c) Write a subclass `ExamQuestionSimpleChoice` of class `ExamQuestion`, in which the `possibleAnswers` are an `ArrayList<String>`, that is, the answer is supposed to be a choice from the list and the `correctAnswer` is of type `int`, representing the position of the correct answer (start counting from 1, since that is what humans normally do).

Write a method `public int mark(int value)` in the subclass that returns full marks if the answer is correct and `0` otherwise. Furthermore, write a suitable `public String toString()` method, making use of inheritance as far as possible.

For instance, if the answer to "2+3" is to be tested, it may be possible to offer the values 4, 5, 10, and 20 as possible answers to an `ArrayList<String>` with `ArrayList<String> a = new ArrayList<String>();` and `a.add("4"); a.add("5"); a.add("10"); a.add("20");` the right answer to the same question would be `2`, (remember, we start counting the answers from 1 here). That is, with
`ExamQuestionSimpleChoice q2 =`
`new ExamQuestionSimpleChoice("2+3 = ?", 10, a, 2);` we should get from `q2.mark(2)` the full `10` marks and from `q2.mark(3)` 0 marks.

(d) Write a subclass `ExamQuestionMultipleChoice` of class `ExamQuestion`, in which the `possibleAnswers` are an `ArrayList<String>`, that is, the answer is supposed to be a choice from the list and the `correctAnswers` is of type `ArrayList<Integer>`, representing the positions of the correct answers (start counting from 1, since that is what humans normally do).

Write a method `public int mark(ArrayList<Integer> answersProvided)` in the subclass that computes the points by scaling the difference between the number of correct answers and the number of incorrect answers to the maximal points (but not returning less than 0). Furthermore, write a suitable `public String toString()` method, making use of inheritance as far as possible.

For instance, if the solutions to `"x*x = 4"` are to be found, it may be possible to offer the values -2, 0, 2, and 3 as possible answers to an `ArrayList<String>` with

`ArrayList<String> possibleAnswers = new ArrayList<String>();` and `possibleAnswers.add("-2"); possibleAnswers.add("0"); possibleAnswers.add("2"); possibleAnswers.add("3");` the correct answers to the same question would be represented in an `ArrayList<Integer>` with `ArrayList<Integer> correctAnswers = new ArrayList<Integer>();` and `correctAnswers.add(1); correctAnswers.add(3);`. That is, with

```
ExamQuestionMultipleChoice question =
 new ExamQuestionSimpleChoice("x*x = 4", 10,
                              possibleAnswers, correctAnswers);
```

we should get from `question.mark(givenAnswers)` with an `ArrayList<Integer> givenAnswers` as indicated the following marks:

- `[1,3]` full marks, i.e., 10.
- `[1]`, `[3]`, `[1,2,3]`, or `[1,3,4]` half marks, i.e, 5.
- in all other cases (e.g., `[1,2]` or `[1,2,3,4]`) no marks, i.e. 0.

<u>Exercise 5:</u> (Advanced, MSc: 20%, ICY: 0%)
THIS EXERCISE IS FOR MSc STUDENTS ONLY:
Explain the rationale for the use of object oriented programming (that is, which problem(s) does it address and how) and its limitations (to which degree are the problem(s) not fully solved this way). The explanation should be factual, well argued and supported by references. For answering this question, you should do background reading and make appropriate use of referencing the material that you have read.
Your executive summary – to be included in your zip file – should consist of two A4 pages with point size 11 and be submitted in accessible PDF format.