© Manfred Kerber
Alexandros Evangelidis

School of Computer Science
University of Birmingham

Worksheet 4

MSc/ICY Software Workshop

Assessed Exercise: 2% of the module mark.

Submission: Thursday 16 November 2017 2pm

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

JavaDoc comments are mandatory. Follow the guidelines in `submission.pdf`

Public tests will be provided a week before the submission deadline. The exercises must pass all these public tests.

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.

<u>**Exercise 1:**</u> **(Basic, MSc: 20%, ICY: 25%)**

(a) Implement an `Aircraft` class with the field variables `private double maxSpeed`, `private double maxWeight`, and `private int maxPersons` with a constructor `public Aircraft(double maxSpeed, double maxWeight, int maxPersons)`, all getters and setters as well as a `public String toString()` method which returns a `String` of the kind:
`"The aircraft has a maximal speed of 10.0 km/h and a maximal weight of 300.0 kg. It can carry 2 persons."`
(Note that after a `1` there should follow a singular, i.e., `"person."`, for any other value of `maxPersons` it should be a plural, i.e., `"persons."`)

(b) Write two sub-classes `HotAirBalloon` and `Aeroplane` of the class `Aircraft` with the additional field variables `private double gasTemperature` and `private double range`, respectively. Again write constructors, all getters and setters as well as `public String toString()` methods which return `String`s of the kinds:

- `"The aircraft has a maximal speed of 10.0 km/h and a maximal weight of 300.0 kg. It can carry 2 persons. It has a gas temperature of maximally 110.0°C."`, and
- `"The aircraft has a maximal speed of 871.0 km/h and a maximal weight of 68000.0 kg. It can carry 222 persons. It has a range of 5300.0 km."`, respectively.

Make use of inheritance as much as possible.

Exercise 2: **(Basic, MSc: 20%, ICY: 25%)** Write a class `Statistics` with two methods:

- `public static double mean(Function<Double,Double> f,`
  `double[] argumentValues)` and
- `public static double standardDeviation(Function<Double,Double> f,`
  `double[] argumentValues)`

that compute the mean and the standard deviation of the application of a function to a non-empty array of elements, respectively. The mean 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 differences of the elements from the mean. In formulae this is for n elements $a_0, \ldots, a_{n-1}$ (with $\sum$ standing for sum, corresponding to a `for` loop):

$$\mu = \frac{1}{n}\sum_{i=0}^{n-1} f(a_i)$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(f(a_i) - \mu)^2}$$

For instance, for `argumentValues` as {`1.0, 2.0, 3.0, 4.0`} and the function `x -> x*x;` the mean is computed as $(1.0 + 4.0 + 9.0 + 16.0)/4 = 7.5$, and the standard deviation as

$$\sqrt{\tfrac{1}{4} \cdot ((1-7.5)^2 + (4-7.5)^2 + (9-7.5)^2 + (16-7.5)^2)} =$$
$$\sqrt{(42.25 + 12.25 + 2.25 + 72.25)/4} = \sqrt{129.0/4} = \sqrt{32.25} \approx 5.678908345800274.$$

Try to make your code computationally efficient.

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

(a) Implement an interface `Sortable` with the public method signature for a method `double compareValue()`. Furthermore, provide implementations for the method in two classes `Car` (with field variables `private double maxSpeed`, `private String carNumber`, and `private String make`) and `Customer` (with field variables `private String name`, `private double totalMoneySpent`, and `private String address`). The implementations may be minimal and contain only constructors, `toString()` methods, and provide implementations to the `Sortable` interface, that is, contain methods `public double compareValue()` that return the maximal speed and the total amount of money spent in their corresponding classes `Car` and `Customer`, respectively.

(b) Modify the `quickSort` algorithm in the class `Sorting` from the lab lecture of Week 5 so that it works for any array of objects of classes implementing the interface `Sortable` (such as `Car[]` or `Customer[]`) with respect to the values given by `compareValue()`.

<u>Exercise 4:</u> (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`. That is, the user would have to enter the correct answer; however, no user interface is required here.

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.

The `toString()` method should return:

```
"Question (Maximal mark: 10): 2+3 = ? Correct answer is: 5"
```

`q1.mark(5)` should return `10`, and `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.toString()`:

```
"Question (Maximal mark: 10): 2+3 = ?
Possible answers are: [4, 5, 10, 20]
Correct answer position is: 2"
```

`q2.mark(2)` should result in the full `10` marks and `q2.mark(3)` in 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 and rounding the points to the next `int` (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 explanation should form an executive summary – to be included in your zip file – and consist of two A4 pages with point size 11 font and be submitted in accessible PDF format (see, `https://www.gov.uk/guidance/how-to-publish-on-gov-uk/accessible-pdfs`).