© Manfred Kerber
Alexandros Evangelidis

School of Computer Science
University of Birmingham
Wednesday 7 November 2018

Worksheet 6

MSc/ICY Software Workshop

Assessed Worksheet: 2% of the module mark.

Submission Deadline is Thursday, 22 November, at 2pm via Canvas.

5% late submission penalty within the first 24 hours. No submission after 24 hours. Follow the submissions guidelines on Canvas. JavaDoc comments are mandatory. All submissions must pass the tests provided on 15 November. For Exercises 1 – 3 submit own tests.

Exercise 1: (Basic, MSc: 30%, ICY: 40%) Write a class `Vehicle` with the two field variables `passengerNumber` and `maxSpeed` both of type `int` with a standard constructor, getter(s), setter(s), and a `toString` method. Furthermore write a subclass `Bus` with the additional field variable `fuelConsumption` of type `double`. The class should contain a constructor, getter(s) and setter(s), and a `toString` method. Make use of inheritance as far as possible.

Exercise 2: (Medium, 30%) Define an interface `Measurable` with the public method signature `public double getValue()`.
Write a class `Invoice` that implements the interface `Measurable` to record an invoice by field variables for the `accountNumber`, the `sortCode`, and the `amount` with types `String`, `String`, and `double`, respectively. Furthermore implement a class `Patient` that implements the interface `Measurable`. It defines a patient by field variables `name`, `age`, and `weight` of types `String`, `int`, `double`, respectively.
In a class `Statistics` write three methods `public static double maximum(ArrayList<Measurable> elements)`, `public static double average(ArrayList<Measurable> elements)`, and `public static double standardDeviation(ArrayList<Measurable> elements)`, which return the maximum, the average, and the standard deviation of the corresponding elements in the array list.
[Note, in order to determine the standard deviation you first determine the `average`. Second, you square the differences between the elements and the average. Thirdly, you sum these squared differences up. Fourthly, this value is divided by the length of the array list minus one. Finally, the standard deviation is the square root (`Math.sqrt`) of this value. For details, see for instance, `https://en.wikipedia.org/wiki/Standard_deviation`.]

Exercise 3: (Advanced, 30%) For this exercise you are supposed to write three classes (`BankAccountUser`, `BankAccountStandardUser`, and `BankAccountAdministrator`). The exact specification is given in the form of three interfaces. You can find these and a few classes on which you build up your work in a `zip` file on the Canvas page for the worksheet.
Assume the classes `Customer`, `Transaction`, and `BankAccount` be given. You are supposed to write three classes to add a (non-interactive) implementation of a very rudimentary internet banking system: an abstract class `BankAccountUser` and two subclasses `BankAccountStandardUser` and `BankAccountAdministrator`.
The class `BankAccountUser` has three field variables: `username`, `password`, and `loggedIn` of types `String`, `String`, and `boolean`, respectively. The methods required are specified in a corresponding interface `BankAccountUserInterface` which the class must implement.
The subclass `BankAccountStandardUser` has the field variables `bankAccount` and `loginAttempts` of types `BankAccount` and `int`, respectively. It should implement the interface `BankAccountStandardUserInterface`.
The subclass `BankAccountAdministrator` has the additional field variable `bankAccountUsers` of type `ArrayList<BankAccountUser>` and has to implement the interface `BankAccountAdministratorInterface`.
An administrator shall have the possibility to reset an account of a user by a method `public void resetAccount(BankAccountUser bankAccountUser, String password)`. The method shall first reset the password. For standard users (and only for standard users) it shall also reset the counter `loginAttempts` of failed login attempts to `0`.
For the latter, you may make use of `bankAccountUser instanceof BankAccountStandardUser` in order to check whether the variable `bankAccountUser` is in the class `BankAccountStandardUser`. If it is in the subclass it is still of type `BankAccountUser`, but it can be cast to a type `BankAccountStandardUser` using the standard type casting approach: `(BankAccountStandardUser) bankAccountUser`.

Exercise 4: (Executive Summary, MSc Students Only 10%) Write an executive summary on the topic: "Explain the rationale for the use of object oriented programming and its limitations." The submission must be an accessible PDF document of two pages (including everything) using an 11pt font.