© Manfred Kerber
Alexandros Evangelidis

School of Computer Science
University of Birmingham

Worksheet 2

MSc/ICY Software Workshop

Assessed Exercise: 9% of the continuous assessment mark.

Submission: Thursday 20 October 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/courses/21955/modules/items/581785`.

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

Exercise 1: (Basic, 30%) Define a class `Employee` and a constructor to create it. An employee should be constructed from the three field variables `firstName`, `surname`, and `yearOfJoining` of types `String`, `String`, and `int`, resp. Implement methods `public String getFirstName()`, `public String getSurname()`, `public int getYearOfJoining()`, and `public void setSurname(String newSurname)`, sets the surname to the `newSurname`. Furthermore write a `public String toString()` method that is used for printing objects of class `Employee` in a user friendly way. (Note, the `toString()` method in this exercise will not be tested, that is, you have flexibility how to write it.)
Note that you have always to comment and test your programs appropriately, not just for this exercise on this worksheet. We will not write this to the exercises in future, but still if you fail to do so you cannot gain full marks.

Exercise 2: (Basic, 20%) Define a class `Student`. A `Student` should be represented by the field variables `name`, `dateOfBirth`, `studentID`, and `degreeProgramme`, each of type String. Write a constructor `Student`. Furthermore write the following getters: `public String getName()`, `public String getDateOfBirth()`, `public String getStudentID()`, and `public String getDegreeProgramme()`. No setters required in this exercise.
Also write a method `public String toString()` that produces the following format: `"[John Smith, 5 October 1993, ID: 1111111, MSc Computer Science]"`. (Note, the `toString()` method in this exercise will be tested, that is, you have to follow the format very precisely.)

Exercise 3: (Medium, 20%) In exercise 2 of Worksheet 1, you wrote a program that converts masses given in the imperial system into the metric system. Write a `Java`-program that can deal with weights given in kilograms. Make use of the conversions:

1 pound 0.45359237 kilograms
1 ounce 1/16 pounds

Define a `Weight` class, and write a constructor `Weight(double kg)` to generate a weight given in kilograms. Furthermore, implement methods `public double getPounds()`, `public double getKilograms()`, `public double getOunces()`, which return the weight in pounds, kilograms, and ounces, respectively. Each of these three methods does not take an argument, and each returns a double. E.g., in order to get the weight of an object `w` in kilograms, you make a call `w.getKilograms();`

Exercise 4: (Advanced, 20%) Write a class `Employee`. Each employee is represented by their `name`, their `hourlySalary`, and their `numberOfHours` of types `String`, `double` and `int`, respectively. Write a class with constructor, getters, setters, and a `toString` method. Note that the naming of constructors, getters, and setter must follow the strict naming convention. Furthermore write two methods: First the monthly salary, `public double monthlySalary()`, which computes for an `Employee` object their monthly salary (as the product of hourly salary and their number of hours). Second, `public void increaseSalary(double percentage)`, which increases the hourly salary by the percentage in the argument.

Exercise 5: (Advanced, 10%) In Exercise 4 of Worksheet 1 we looked at addition and multiplication of two fractions. In the current exercise fractions should be represented by a `Java` class `Fraction`. You have to define this class, which in addition to the constructor and the getters `getNumerator` and `getDenominator` has methods `toString` (used to print a rational number) as well as `public Fraction add(Fraction summand)`, `public Fraction multiply(Fraction factor)`, and `public boolean less(Fraction comp)` which add to a fraction another fraction, multiplies a fraction with another fraction, and checks whether the fraction is less than a second fraction, respectively. The first two return corresponding objects of Class `Fraction`, the latter a boolean. For instance, if we generate fractions `Fraction f1 = new Fraction(1,2);` and `Fraction f2 = new Fraction(3,7);` then

- `f1.toString()` should return the string `"1/2"`;
- `f2.multiply(f1).toString();` should return the String `"3/14"` (which corresponds to the product of `f2` and `f1`); and
- `f2.add(f1).toString();` should return the String `"13/14"` (which corresponds to the sum of `f2` and `f1`).
- `f2.less(f1);` should return the boolean `true` (since `f2` is less than `f1`).