© Manfred Kerber
Alexandros Evangelidis

School of Computer Science
University of Birmingham

Worksheet 2

MSc/ICY Software Workshop

Assessed Exercise: 2% of the module mark.

Submission: Thursday 19 October 2017 2pm

5% late submission penalty within the first 24 hours. No submission after 24 hours. JavaDoc comments are mandatory. Follow the submission guidelines on `submission.pdf` on Canvas, available as `https://canvas.bham.ac.uk/files/4543348`.

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

Exercise 1: (Basic, 30%) Define a class `Film` and a constructor to create it. A film should be constructed from the three field variables `title`, `year`, and `length` of types `String`, `int`, and `int`, resp. Implement getter methods `public String getTitle()`, `public int getYear()`, `public int getLength()`, and a setter method `public void setLength(int newLength)` that sets the length to the `newLength`. Furthermore write a `public String toString()` method that is used for printing objects of class `Film` 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 `ClubMember`. A `ClubMember` should be represented by the field variables `name`, `dateOfBirth`, `registrationNumber`, and `membershipType`, each of type String. Write a constructor `ClubMember`. Furthermore write the following getters: `public String getName()`, `public String getDateOfBirth()`, `public String getRegistrationNumber()`, and `public String getMembershipType()`. No setters are required in this exercise.

Also write a method `public String toString()` that produces the following format: `"[John Smith, 5 October 1993, ID: C212121, Gold]"`. (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 distances given in kilometres and can convert them to metres, miles, and yards. Make use of the conversions:

1 mile	1.60934 kilometres
1 metre	0.001 kilometres
1 mile	1760 yards

Define a `Distance` class, and write a constructor `Distance(double km)` to generate a distance given in kilometres. Furthermore, implement methods `public double getMiles()`, `public double getKilometres()`, `public double getMetres()`, and `public double getYards()` which return the distance in miles, kilometres, metres, and yards, respectively. Each of these four methods does not take an argument, and each returns a double. E.g., in order to get the distance of an object `distance` in kilometres, you make a call `distance.getKilometres();`

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(double taxRate)`, which computes for an `Employee` object their monthly salary (as the product of hourly salary and their number of hours, minus the tax computed at a fixed tax rate). Second, `public void increaseSalary(double percentage)`, which increases the hourly salary by the percentage in the argument.
For example, let us assume an employee `Employee john = new Employee("John", 10, 40);`. `System.out.println(john);` should give us `John has an hourly salary of 10.0 £ and has worked last month for 40 hours.`
`john.monthlySalary(20)` should compute the salary of John as $10 * 40$ and then subtract a tax of 20%, that is, 400 minus 80 which gives us 320.
After `john.increaseSalary(1)` John's hourly salary is increased by one per cent and will be 10.1 rather than 10.0.

Exercise 5: (Advanced, 10%) In this exercise we look at complex numbers. Complex numbers consist of pairs of real numbers (represented by `double`), the so-called real part and the so-called imaginary part. They are written as $z = a + bi$. Addition and multiplication of two complex numbers $z_1 = a_1 + b_1 i$ and $z_2 = a_2 + b_2 i$ are defined as:

- $\text{add}(z_1, z_2) = (a_1 + a_2) + (b_1 + b_2)i$
- $\text{multiply}(z_1, z_2) = (a_1 * a_2 - b_1 * b_2) + (a_1 * b_2 + a_2 * b_1)i$

Define a corresponding class `Complex`, which in addition to the constructor and the getters `getRealPart` and `getImaginaryPart` has methods `toString` (used to print a complex number) as well as `public Complex add(Complex summand)` and `public Complex multiply(Complex factor)` which adds to a complex number another complex number and multiplies a complex number with another complex number, respectively. For instance, if we generate complex numbers `Complex c1 = new Complex(1,1);` and `Complex c2 = new Complex(3,7);` as well as `Complex c3 = new Complex(1,0);` then

- `c1.toString()` should return the string `"1.0 + 1.0i"`;
- `c3.toString()` should return the string `"1.0 + 0.0i"`;
- `c2.add(c1).toString();` should return the String `"4.0 + 8.0i"` (which corresponds to the sum of `c2` and `c1`); and
- `c2.multiply(c1).toString();` should return the String `"-4.0 + 10.0i"` (which corresponds to the product of `c2` and `c1`).