© Manfred Kerber
Alexandros Evangelidis

School of Computer Science
University of Birmingham
Wednesday 21 November 2018

Worksheet 8

MSc/ICY Software Workshop

Assessed Worksheet: 2% of the module mark.
Submission Deadline is Thursday, 6 December 2018, 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. This worksheet will be vivaed. No public tests will be provided, but you should still write your own JUnit tests.

Exercise 1: (Basic, 30%) In the lecture (see the Canvas pages for Week 7) we have seen in a package `company` an abstract class `Employee` with two subclasses `HourlyEmployee` and `MonthlyEmployee`. Adapt these classes (and the interface `Payable`) so that the salary is of type `double`. Write a class `Company` with field variable `ArrayList<Employee> employees` with the usual getter and setter. Furthermore write in the `Company` class a method `public void increaseSalaries(double rate)` which increases the salaries of all employees (whether paid on an hourly rate or with a fixed monthly salary) by the fixed rate. (Note, it makes sense to have corresponding `increaseSalary` methods – abstract or not – in the three classes dealing with employees.)
For instance, with a pay increase of 0.02, somebody earning £10 per hour would then earn £10.20 per hour and somebody earning £1800 a month would then earn £1836 per month. That is, the method `increaseSalaries` should go over the whole `ArrayList` of all `employees` and increase all their salaries.

Exercise 2: (Medium, 30%)

(a) Implement a shopping cart. The customer can add products to the shopping cart and can inspect the current state by printing it. The two classes, `Product` and `ShoppingCart` should reside in the package `shop`.

The class `Product` should have the three field variables `private String name`, `private double price`, and `private int quantity` with a standard constructor and standard getters. The field `price` stores the price for a single product. The class should also contain a method `public double getTotalPrice()`, which computes the total price by multiplying the single price and the quantity.

The class `ShoppingCart` has to contain a constructor `public ShoppingCart()`, a `toString` method and a method `public void add(Product p)` to add a product to the shopping cart. Note that products with the same name and price should not be added multiple times to the ArrayList, but that their quantity is to be adjusted.

For instance, a shopping cart containing the three products `Product("Milk (1l)", 1.12, 2)`, `Product("Bread", 0.78, 2)`, and `Product("Apples", 0.49, 4)` should be printed as:

```
 2 * GBP    1.12 Milk (1l)       = GBP    2.24
 2 * GBP    0.78 Bread           = GBP    1.56
 4 * GBP    0.49 Apples          = GBP    1.96
                                 ---------------
                           TOTAL GBP    5.76
```

Note that prices should be rounded and displayed to exactly two digits after the decimal point.

(b) Introduce multibuy discounts using a class `public class MultiBuyProduct extends Product` with constructor `public MultiBuyProduct(String name, double price, int quantity, int minDiscountedQuantity, int discountPercent)` and method `public double getTotalPrice()`.

`new MultiBuyProduct("Tomato", 0.5, 20, 10, 3)` means that you buy 20 tomatoes at a price of £0.50 and since you buy at least 10 you get a discount of 3%. `MultiBuyProduct` should reuse functionality from its superclass wherever possible.

For instance, a shopping cart created by adding the products `Product("Milk (1pt)", 0.79, 9)`, `MultiBuyProduct("Tomato", 0.5, 20, 10, 3)`, `Product("Tomato", 0.53, 5)`, and `Product("Milk (1pt)", 0.79, 4)` should be printed as:

```
13 * GBP    0.79 Milk (1pt)      = GBP   10.27
20 * GBP    0.50 Tomato          = GBP    9.70
  (Multibuy Discount: GBP 0.30)
 5 * GBP    0.53 Tomato          = GBP    2.65
                                 ---------------
                           TOTAL GBP   22.62
```

<u>Exercise 3:</u> (Advanced, 30%) For this exercise you may make use of the `Date` class from the lab lecture of Week 4. Write a program that allows users to book a number of rooms. Concretely, it should be possible to book a room by a method `public boolean book(String room, Date date, int hour, String purpose)` where the `room` is taken from a fixed number of rooms such as `String[] rooms = {"R217", "R222", "R225", "R245"}`, `date` is a date such as `Date nov22 = new Date(22, "November", 2018)`, `hour` is the hour of the day for which the room is needed such as `14` to book the room from 14:00 to 15:00, and `purpose` describes what the room is needed for. It should be possible to construct objects by:
`RoomBooking bookings2018 = new RoomBooking(2018, rooms)`.
The method `book` returns `true` if the room is available, `false` else. Bookings can be made for hours starting at `9:00`, `10:00`, ..., `17:00`.
Write also a method `public void cancel(String room, Date date, int hour)` that deletes a booking if it exists, so that the room is again available for further bookings. It should do nothing if the booking does not exist.
For instance, it should be possible to make bookings such as:

```
bookings2018.book("R222", nov22, 12, "Tutor meeting");
bookings2018.book("R222", nov22, 12, "Java meeting"); //no booking since room booked already
bookings2018.book("R222", nov22, 13, "Interviews");
bookings2018.book("R245", nov22, 16, "Project meeting");
bookings2018.book("R225", nov22, 14, "Top-up tutorial");
```

Write also a method `public String displayDay(Date date)` so that `bookings2018.displayDay(nov22)` produces a `String` such as:

```
                                       22 November 2018

      |      R217          |      R222          |      R225          |      R245          |
------+--------------------+--------------------+--------------------+--------------------+
  9:00|                    |                    |                    |                    |
------+--------------------+--------------------+--------------------+--------------------+
 10:00|                    |                    |                    |                    |
------+--------------------+--------------------+--------------------+--------------------+
 11:00|                    |                    |                    |                    |
------+--------------------+--------------------+--------------------+--------------------+
 12:00|                    |     Tutor meeting  |                    |                    |
------+--------------------+--------------------+--------------------+--------------------+
 13:00|                    |        Interviews  |                    |                    |
------+--------------------+--------------------+--------------------+--------------------+
 14:00|                    |                    |   Top-up tutorial  |                    |
------+--------------------+--------------------+--------------------+--------------------+
 15:00|                    |                    |                    |                    |
------+--------------------+--------------------+--------------------+--------------------+
 16:00|                    |                    |                    |   Project meeting  |
------+--------------------+--------------------+--------------------+--------------------+
 17:00|                    |                    |                    |                    |
------+--------------------+--------------------+--------------------+--------------------+
```

<u>Exercise 4:</u> (Debugging, 10%) In the folder `Ex4` on Canvas you find a class `PGMImage` that contains a method `public void rotate (String filename)` that should rotate a `PGMImage` by 90° clockwise and write it to a file `filename`. However, tests with images show that the images are not correctly rotated. What is the problem? Fix it.