COM1003 Java Programming
Autumn Semester 2020-21
Programming Assignment 3

Dr Siobhán North
Department of Computer Science, The University of Sheffield

Learning outcomes

This assignment will assess your ability to:

- Write a program from a specification;
- Write clear, good quality program code;
- Manipulate arrays in Java
- Work with interacting classes and a package

This assignment is worth 20% of your mark for the first semester of the module and must be submitted by 14 December 2020. You will find information about the exact deadline, the marking scheme and how you must submit your work at the end of this document.

The specification below is quite precise in telling you what sort of input you should expect and what you must output but usually does not tell you how to get from one to the other. This is deliberate and part of the test. However it is not meant to be ambiguous so if there is something you don't understand you can ask questions, ideally by email. I will put all the email questions and their answers on an FAQ page as they come in. Even if you think you understand everything it is a good idea to check this page before you submit.

You do not need to do everything described below to hand in your work and get marks for it. The marks will be awarded depending on how much you have achieved (see below for details) but if you hand in a program it must compile and run to get any marks. So a well written program that does something is always better than a program which would do more if it had worked.

For this assignment you have been provided with a package and its documentation. Each of your tasks involves changing one or more methods of one class in the package. To check if you are on the right lines there is a test program for each task but passing the test is not sufficient to get full marks. You must also produce clear well structured code which meets the specification.

The scenario revolves around a flat share. There is a zipped folder called `flatshare` which you should download and unzip. It contains a package which you will need to compile before you can do anything. There are also some test programs that are explained below. The documentation is on the web page, linked from Blackboard, because Blackboard does not cope with html very well.

As usual it is a good idea to upload your work to Blackboard as you go on (see the end of the document for what to upload) but only upload a working program and don't change it at the last minute without recompiling and running it.

Task one

The package `flatshare` contains a class called `Flat.java` which has a constructor that takes a `String` as a parameter. If the parameter's actual value is a `String` containing a list of names separated by commas and possibly spaces then it should create an instance of the flat where the named people are tenants but currently the constructor has only a comment as a body. Replace the body of the constructor with something that will actually work. You must do this without changing anything else in either the package or the class. This is part of the specification and you will get very few marks if you change anything else. You can test your amendment of the class by running `TestFlat1`

If you do the correctly with a perfectly written program you will get 20%

Task 2

If the ultimate purpose of the program is to keep track of people's contributions over time it will be necessary to save everything. Modify the method `saveToFile` of the `Flat` class so it saves everyone's details in a file (whose name is provided as a parameter). People's details should be saved one to a line in the form of the output from the `toString()` method of the `Person class`. Terminate the file by adding a last line that consists of a single asterisk. This avoids complications caused by different versions of non-printing characters on different operating systems.

This is only useful if you can read the file in again. The `Flat` class has another constructor that takes an `EasyReader` as a parameter. Modify its body so that it reads in the output of `saveToFile` and reinstates the class.

You must do this without changing anything except the two method bodies. You can test your amendment of the class by running `TestFlat2`

If you do the correctly with a perfectly written program you will get another 20% so 40% in total.

Task 3

The tenants of the flat are likely to change over time. The class `Flat` has a method called `addTenant()` and another called `removeTenant()`. Both take the name of a tenant as a parameter. `addTenant()` adds a new tenant to the end of the list of tenants output by the `toString()` method and `removeTenant()` removes the tenant without changing the order of the list of tenants or leaving a gap.

You must do this without changing anything except the two methods `addTenant()` and `removeTenant()` in either the package or the class. You can test your amendment of the class by running `TestFlat3`

If you do the correctly with a perfectly written program you will get another 20% so 60% in total.

Task 4

This task deals with buying things for the flat. The `person class` has a contribution variable which is set to zero initially. If one of the tenants buys something for everyone to use their contribution goes up by the price of what they bought but everyone's contribution (including the shopper's) goes down by the price of the item divided by the number of tenants.

There is a method in `Flat` called `purchase()` to deal with this. It takes three parameters; the name of the shopper, the name of the thing they bought and the price in that order. Modify the body of this method to deal with this.

The `toString()` method of `Person` prints out the current contribution in brackets after the person's name if it is non-zero. Obviously not everything will be divisible by the number of people in the flat but that is true of real life as well. As long as it is correct to the nearest penny there is no problem.

You must do this without changing anything except the method `purchase()` in either the package or the class so the number of decimal places output is outside your control. You can test your amendment of the class by running `TestFlat4`

If you do the correctly with a perfectly written program you will get another 12% so 72% in total.

Task 5

There is always a risk that there could be an argument about what is a household necessity. To avoid this the package contains an `enum` called `Required`. Only items on the `Required` list can be charged to the flat. The `Required enum` has a method `itemCalled()` which takes a `String` as a parameter and returns either the `enum`, if the item is on the list or `null` if it isn't. Modify `purchase()` to make sure that only things in the `Required enum` can count for someone's contribution.

You must do this without changing anything except the method `purchase()` in either the package or the class. You can test your amendment of the class by running `TestFlat5`

If you do the correctly with a perfectly written program you will get another 12% so 84% in total.

Task 6

In Task 2 you saved the tenants' details in a file and read them in again. You used the `toString()` method to print them out. Modify the constructor that takes an `EasyReader` as a parameter to cope with an input that may contain a contribution in brackets after the tenants name.

You must do this without changing anything else in either the package or the class. You can test your amendment of the class by running `TestFlat6` but make sure `TestFlat2` still works as well.

If you do the correctly with a perfectly written program you will get another 16% so 100% in total.

Submission and deadline

You must submit *only* the file called `Flat.java` from the `flatshare` package via the submission point on Blackboard by 3pm on Monday 14 December. Do not submit anything else and do not submit it in any other format. Blackboard is very bad at displaying Java programs so you may think that Blackboard has messed up the layout of your program but I will download the program text and the layout will be preserved.

Late work will be penalised using the standard University scale (a penalty of 5% per working day late; work will be awarded a mark of zero if it is more than 5 working days late).

This is an individual assignment. You must work on it alone and hand in your own work. If you work collaboratively and then pretend you did the work alone we will find out (we have a very good plagiarism checker and all submitted work will go through it) and, as you have already been told, we take the use of unfair means in the assignment process very seriously. Don't even think about handing in work you didn't do yourself.

The Marking Scheme

The mark for this assignment is worth 20% of the first semester mark and so 10% of the overall mark for COM1003.

The criteria for program style quality are the same as for the first assignment. The style mark will be calculated as a percentage as follows and will form 25%

Readable layout	40
Correct use of identifiers	20
Useful comments	40
Total	100

of your overall mark. Another 25% will come from meeting the specification. So if you only hand in Task 1, of the 20% 10% will be for how well you do the task and 5% for sticking to the specification and 5% will be your style mark and so on.

The marking scheme for the various versions is as follows:

Version	1	2	3	4	5	6	Total
String manipulation	4					5	9
Use of Arrays	4		10				14
Use of Easywriter		4				1	5
Use of Easyreader		4				1	5
Use of methods	2	2		4	6	1	15
Calculation				2			2
Meeting Specification	5	5	5	3	3	4	25
Style	5	5	5	3	3	4	25
Total	20	20	20	12	12	16	100