COM1003 Java Programming
Autumn Semester 2021-22
Programming Assignment 3

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

Learning outcomes

This assignment will assess your ability to:

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

This assignment is worth 20% of your mark for the first semester of the module and must be submitted by 19 January 2022. 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 you must output but usually does not tell you how to achieve it. 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 but with minimal documentation. Each of your tasks involves changing one or more methods of one or more classes so that the Task test program runs successfully. This is not sufficient for full marks, you must also produce clear well structured code which meets the specification.

You are only allowed to change the method bodies as described in the task, not anything else in any of the classes.

The scenario revolves around a group of people sharing a meal in a restaurant and splitting the bill fairly even in the situation where one person had three courses and a gin cocktail and another had a glass of water and a plate of chips.

There is a zipped folder called `resturant` which you should download and unzip.

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 to test it.

Task 1

The package `resturant` contains the class `Task1` with the `main` method

```
public static void main (String [] args) {
      Diner.createPartyFor(args);
      Diner.listAllTheDiners();
}
```

but the two methods it calls currently have empty bodies and you must change them to make it work. You will notice `Task1` expects to take values from the `args` parameter (see Exercise1G). These values are the names of the diners. The method `createPartyFor()` should use them to create and populate a static array in `Diner` called `allTheDiners` by creating a new `Diner` object for each name supplied with the correct value for its instance variable `theDinersName`. The second method call in `Task1`, `listAllTheDiners()`, should print out "The diners are" on the first line followed by the names of the diners one to a line ending with a blank line. A static `EasyWriter` object called `screen` has already been declared in the class `Diner` which you must use to do display this information.

To do this successfully you must fill in the bodies of three methods in `Diner`; the constructor, `createPartyFor()` and `listAllTheDiners()`. You may not change anything outside the bodies of these three methods.

So a successful run of Task1 could look like this

```
...>java resturant/Task1 Harry Hermione Ron
The diners are
Harry
Hermione
Ron
```

If you do this correctly with a perfectly written program you will get 40%

Task 2

`Task2`'s main method

```
public static void main (String [] args) {
      Diner.createPartyFor(args);
      Diner.takeEveryonesOrder();
      Diner.listEveryonesOrder();
}
```

starts like Task1 but now you need to find what everyone is ordering and then display it. The first method call `createPartyFor()` is already written. You now need to fill in the bodies of two more methods in `Diner` and some other methods as well.

The method `takeEveryonesOrder()` should go through all the diners in turn to ask for their order by asking them to choose an item and go on asking them to choose items until they don't want anything else. It should then move on the the next diner.

Diner already has an instance variable

```
Order[] thingsOrderedByDiner;
```

to store the things the diner orders. An incomplete version of the class `Order` has been provided as part of the package. An `ArrayList`, rather than an array, would have been better for `thingsOrderedByDiner` but we will be using an array for practice so you should not change this line.

This variable needs a value assigned to it in the constructor for `Diner` and then each item ordered can be added to it up to a maximum defined by the constant

```
static final int
   MAX_NUMBER_OF_THINGS_A_DINER_CAN_ORDER=10;
```

which has also already been declared.

So `takeEveryonesOrder()` should fill in the `thingsOrderedByDiner` array by calling the instance method `askForTheirOrder()` for each diner and `askForTheirOrder()` in turn should call a method called `askForANewThingOrdered()` in the class `Order`. `askForANewThingOrdered()` asks if the item ordered is a Starter, Main, Side, Pudding or Drink and how much the item costs using an `EasyReader` object called `keyboard` already declared for you. There is an `enum` for Starter, Main, Side, Pudding and Drink called `MenuItem` which has also already been declared and it has a method called `Called` which takes a `String` as a parameter and returns either the appropriate `MenuItem` or null if it can't guess what the parameter is meant to be. It will accept any identifiable shorthand for the menu item and is not case sensitive. You have no need to change it. If the item ordered is not identifiable `askForANewThingOrdered()` must return null otherwise it asks the price and returns an `Order` object.

Finally you need to write the last method in Task2 `listEveryonesOrder()`. This does what the name says but there is a useful `toString` method in Order which you can use to get a tidy output. You don't need to do anything with it except use it.

So for Task 2 you need to fill in the bodies of five methods; `takeEveryonesOrder()` `askForTheirOrder()` and `listEveryonesOrder()` in `Diner` and `askForANewThingOrdered` and the constructor in `Order`. You will need to make a small change to the constructor for `Diner` too but should change nothing else.

A successful run of Task2 could look like this

```
...>java resturant/Task2 Harry Ron Hermione
Time for Harry to order
What do you want? (starter, main, pudding, side, drink
    or nothing) starter
What is the starter's price? 7.50
```

```
What do you want? (starter, main, pudding, side, drink
    or nothing) main
What is the main's price? 15
What do you want? (starter, main, pudding, side, drink
    or nothing) nothing else

Time for Ron to order
What do you want? (starter, main, pudding, side, drink
    or nothing) side
What is the side's price? 3.50
What do you want? (starter, main, pudding, side, drink
    or nothing) drink
What is the drink's price? 5.50
What do you want? (starter, main, pudding, side, drink
    or nothing) pudding
What is the pudding's price? 6.75
What do you want? (starter, main, pudding, side, drink
    or nothing) nothing

Time for Hermione to order
What do you want? (starter, main, pudding, side, drink
    or nothing) drink
What is the drink's price? 7.50
What do you want? (starter, main, pudding, side, drink
    or nothing) nothing

The diners' order is
   Harry
      Starter    7.50
      Main      15.00
   Ron
      Side       3.50
      Drink      5.50
      Pudding    6.75
   Hermione
      Drink      7.50
```

although the items ordered could have been abbreviated to just their initial letters except for starter and side which need the first two letters and nothing could equally well have been "That's all" because anything not identifiable as a starter, main, pudding, side or drink is treated as the end of the list.

If you do this correctly with a perfectly written program you will get another 30% so 70% in total.

Task 3

`Task3`'s main method

```
public static void main (String [] args) {
   Diner.createPartyFor(args);
   Diner.takeEveryonesOrder();
   Diner.listEveryonesOrder();
   Diner.dealWithTheBill();
}
```

is the same as Task 2 except it has one extra line, the call to the method `dealWithTheBill` in `Diner`. `dealWithTheBill` should work out how much the whole party owes and then list their individual contributions on the basis that they will pay whatever they owe rounded up to the nearest whole pound (see `Math.ceil()`). Finally it should say how much they would overpay by that method. All output should use the EasyWriter variable `screen` already declared. So the first line output should be "The diners owe " followed by the amount to two decimal places displayed using the methods of `screen` variable. On the next line it should say "Everyone's debt rounded up is" and then, on subsequent lines, the individual rounded up debts one to a line name first with no decimal places and finally the last line should be "which is too much by " followed, on the same line by the amount to two decimal places using `screen` again.

A successful run of Task3 could look like this

```
...>java resturant/Task3 Harry Ron Hermione
Time for Harry to order
What do you want? (starter, main, pudding, side, drink
    or nothing) st
What is the starter's price? 7.5
What do you want? (starter, main, pudding, side, drink
    or nothing) m
What is the main's price? 15
What do you want? (starter, main, pudding, side, drink
    or nothing) x

Time for Ron to order
What do you want? (starter, main, pudding, side, drink
    or nothing) si
What is the side's price? 3.5
What do you want? (starter, main, pudding, side, drink
    or nothing) d
What is the drink's price? 5.5
What do you want? (starter, main, pudding, side, drink
    or nothing) p
What is the pudding's price? 6.75
What do you want? (starter, main, pudding, side, drink
    or nothing) n

Time for Hermione to order
What do you want? (starter, main, pudding, side, drink
    or nothing) d
```

```
What is the drink's price? 7.5
What do you want? (starter, main, pudding, side, drink
    or nothing) nothing thank you

The diners' order is
Harry
Starter     7.50
Main       15.00
Ron
Side        3.50
Drink       5.50
Pudding     6.75
Hermione
Drink       7.50

The diners owe   45.75
Everyone's debt rounded up is
Harry 23
Ron 16
Hermione 8
which is too much by 1.25
```

If you do this correctly with a perfectly written program you will get another 15% so 85% in total.

Task 4

`Task 4` considers the possibility that everyone may not order everything in one go. Its main method is

```
public static void main (String [] args) {
      Diner.createPartyFor(args);
      Diner.takeEveryonesOrder();
      Diner.listEveryonesOrder();
      Diner.getAnotherRound();
      Diner.listEveryonesOrder();
}
```

It includes a new method call `getAnotherRound` which allows the diners to order one new item each. It does this by a call to another new method in Orders also called askForANewThingOrdered but this version takes, as a parameter, a String, the name of the diner being asked about extras. So each diner is asked "What else do you want ....?" where ... is replaced by the diner's name. And second time around there is no need to list the items available.

So a successful run of Task3 could look like this

```
       ...>java resturant/Task4 Harry Ron Hermione
Time for Harry to order
What do you want? (starter, main, pudding, side, drink
    or nothing) st
What is the starter's price? 8.56
```

```
What do you want? (starter, main, pudding, side, drink
    or nothing) m
What is the main's price? 15
What do you want? (starter, main, pudding, side, drink
    or nothing) d
What is the drink's price? 4
What do you want? (starter, main, pudding, side, drink
    or nothing) x

Time for Ron to order
What do you want? (starter, main, pudding, side, drink
    or nothing) si
What is the side's price? 4
What do you want? (starter, main, pudding, side, drink
    or nothing) d
What is the drink's price? 5.9
What do you want? (starter, main, pudding, side, drink
    or nothing) x

Time for Hermione to order
What do you want? (starter, main, pudding, side, drink
    or nothing) d
What is the drink's price? 9.50
What do you want? (starter, main, pudding, side, drink
    or nothing) nothing

The diners' order is
Harry
Starter     8.56
Main       15.00
Drink       4.00
Ron
Side        4.00
Drink       5.90
Hermione
Drink       9.50

What else do you want Harry? d
What is the drink's price? 12
What else do you want Ron? m
What is the main's price? 13.45
What else do you want Hermione? st
What is the starter's price? 8.23

The diners' order is
Harry
Starter     8.56
Main       15.00
Drink       4.00
Drink      12.00
```

```
Ron
Side         4.00
Drink        5.90
Main        13.45
Hermione
Drink        9.50
Starter      8.23
```

Submission and deadline

You must submit *only* the files called `Diner.java` and, if you get beyond Task 1, `Order.java` from the `resturant` package via the submission point on Blackboard by 3pm on Monday 19 January. 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. You can upload the program as many times as you like before the deadline and the last versions you uploaded will be marked but, if you get beyond Task 1, the two files must be uploaded as a pair even if one or other of them has not changed since the previous submission. If you don't upload anything before the deadline the first version you upload will be marked.

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 marking scheme for the various versions is as follows:

Version	Task 1	Task 2	Task 3	Task 4	Total
Use of methods and parameters	6	4	1	3	14
Use of arrays	6	4	1	2	13
Use of objects	6	4	1	1	12
Use of local variables	5	2	1	1	9
Use of loops	3	4	1	1	9
Arithmetic			3		3
Use of EasyReader or EasyWriter		2	2	2	6
Logical structure	5	2	1	1	9
Readable structure	2	2	1	1	6
Correct indention	2	2	1	1	6
Meets specification	5	4	2	2	13
Total	40	30	15	15	100