JAVA FOUNDATIONS ASSIGNMENT

Exercise proposal

Create an application for a world-cuisine restaurant which intents to start receiving on-line orders, so that customers are free to arrange a time shift for dinner and a menu which can consist of a starter (st), a main course (mc), and a dessert (ds) per person. Drinks are not included in this initial version of the application. This beta version of the application will work locally only. Thus, networking features will not be required yet. Instead of that, you will receive customer orders through a text file which will be described later.

The restaurant manager is concerned about food allergies and intolerances, and the menu allows to select gluten-free dishes (gfd), vegetarian dishes (vgd), halal-meat dishes (hmd), and seafood-free dishes (sfd). Furthermore, the restaurant is participating in a survey which tries to get a picture of nowadays-society lifestyle. For this reason, each order including any dish from at least one of the four categories mentioned above, has to be reported it to a remote database (which in this initial version will be an output text file).

A typical on-line order will consist of a CSV (Comma-Separated Values) file, in which the first line will relate to category names *(customerName,type,gfd,vgd,hmd,sfd, extras):*

- ***customerName***. The name (and maybe surname, or nickname) of the customer.
- ***type***. Indicates whether the order refers to a starter (*st*), a main course (*mc*) or a dessert (*ds*).
- **dishName**. The name of the dish.
- ***gfd***. A boolean value (*true* or *false*) which stands for 'gluten-free dish'. If any of the ordered foods (starter, main course or dessert) contains gluten, this field will be set to '*true*', and '*false*' otherwise.
- ***vgd***. Similar to the previous item, but for a 'vegetarian dish'.
- ***hmd***. Similar to the previous item, but for a 'halal-meat dish'.
- ***sfd***. Similar to the previous item, but for a 'seafood-free dish'.
- ***extras***. This field includes different attributes depending on the dish type. It indicates which cutlery is needed to consume the dish if it is a starter(st), which is the main ingredient and the preferred drink (separated by an hyphen) if it is a main course (mc), or how many calories the dish has if it is a dessert (ds).

The on-line order list will be provided in a '.txt' file. As a sample:

```
customerName,dishName,type,gfd,vgd,hmd,sfd,extras
Peter Parker,cinammon custard,ds,false,false,false,true,120
Dora the explorer,cottage salad,st,true,true,false,true,Regular cutlery
Bruce Wayne,Kyoto-style ramen,mc,false,true,false,true,Noodles-Rose wine
```

As a developer, you will have to create an application that allows to read the file and manage its content. The idea is creating an OOP software that will show on the console each order (line) in an appealing way.

To ensure you address the problem showing some of your Java development skills, there are some requirements to be fulfilled:

1. An abstract class called 'Dish' must be used to define common attributes and

methods across dishes (e.g. every dish has a name).
2. The 'main' class MUST implements an interface which defines distinct operations (methods) to be performed on the on-line order list.

Both code snippets are provided to the developer.

Program performance

The application has to read from the on-line order file (*onlineOrderSample.txt* is provided) and organise the information in a List. At this point, (as minimum requirements) the developer can simply execute every method in the implemented interface (*OnlineOrderOps.java*) and show the results on the console. Doing so, the user should see the number of orders returned by getNumberOrders, the description of a particular order returned by getOrder(), etc. As mentioned above, the stats of the every dish type (gfd, vgd, hmd, sfd) must be provided as well (e.g. if there are 10 orders and 2 of them include gluten-free dishes, a message should be prompted saying that *"20% of the ordered dishes are gluten-free"*).

For extra marks, the developer can choose to set a menu with several options so that the user can select the info to seek for. Furthermore, the application can provide stats per type of dish and per type of dish and customer. Doing so, and following the previous example, if there are 10 orders and 5 different customers, the stats will be calculated out of 10 and out of 5.

TIPS

- It is advisable to start doing the minimum-requirements exercise, and later try to add a user menu.
- The user menu (if implemented) can use numbers, letters, words, etc. to select among the distinct options provided.
- Take into account that some interface methods do not return String type values. Thus consider overriding the `toString()` method.
- `List`[1] is a Java abstract class. This means it cannot be instantiated. For this reason, the developer has to use any of the subclasses offered by the Java platform, such as `AbstractList`, `AbstractSequentialList`, `ArrayList`, `AttributeList`, `CopyOnWriteArrayList`, `LinkedList`, `RoleList`, `RoleUnresolvedList`, `Stack`, `Vector`.
- Remember polymorphism in Java, which allows to do things like:

```
List<String> myList = new Vector<>();
```

- Take into account that in Java every class inherits from the `Object` class.
- To read from and write to files, the developer can use Java API classes such as `File`[2], `BufferedWriter`[3], `BufferedReader`[4], and `Scanner`[5], for instance.
- The developer can (and should) create any other method which considers necessary to make the code clear and understandable.

1 https://docs.oracle.com/javase/8/docs/api/java/util/List.html
2 https://docs.oracle.com/javase/7/docs/api/java/io/File.html
3 https://docs.oracle.com/javase/7/docs/api/java/io/BufferedWriter.html
4 https://docs.oracle.com/javase/7/docs/api/java/io/BufferedReader.html
5 https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

- It is strongly advise to include comments to make the application more easily shareable across programmers.
- The application assessment will consist of trying distinct order files checking that the on-screen output is right.
- If you spot any error, please report it as soon as possible.
- The submission date has been set up in class for:

__/__/2016