**An entire restaurant**

Department: Electronic and Electrical Engineering

Title:Sheffield restaurant

Student Name: your name

Registration Number: --------

**I declare that this work is my own and I acknowledge the contribution of**

**others where appropriate.**

**Overview**

The whole project consists of three packages, namely, *drink*, *meal* and *Sheffield*. In the drink package, multiple drinks are implemented, such as *beers* and *redwines*. In the *meal* package, three main foods and ingredients are implemented. Finally, the *Menu* class is used to instantiate and display the above mentioned classes, that is, to integrate them into a fixed menu style.

The UML of the project is as follows:

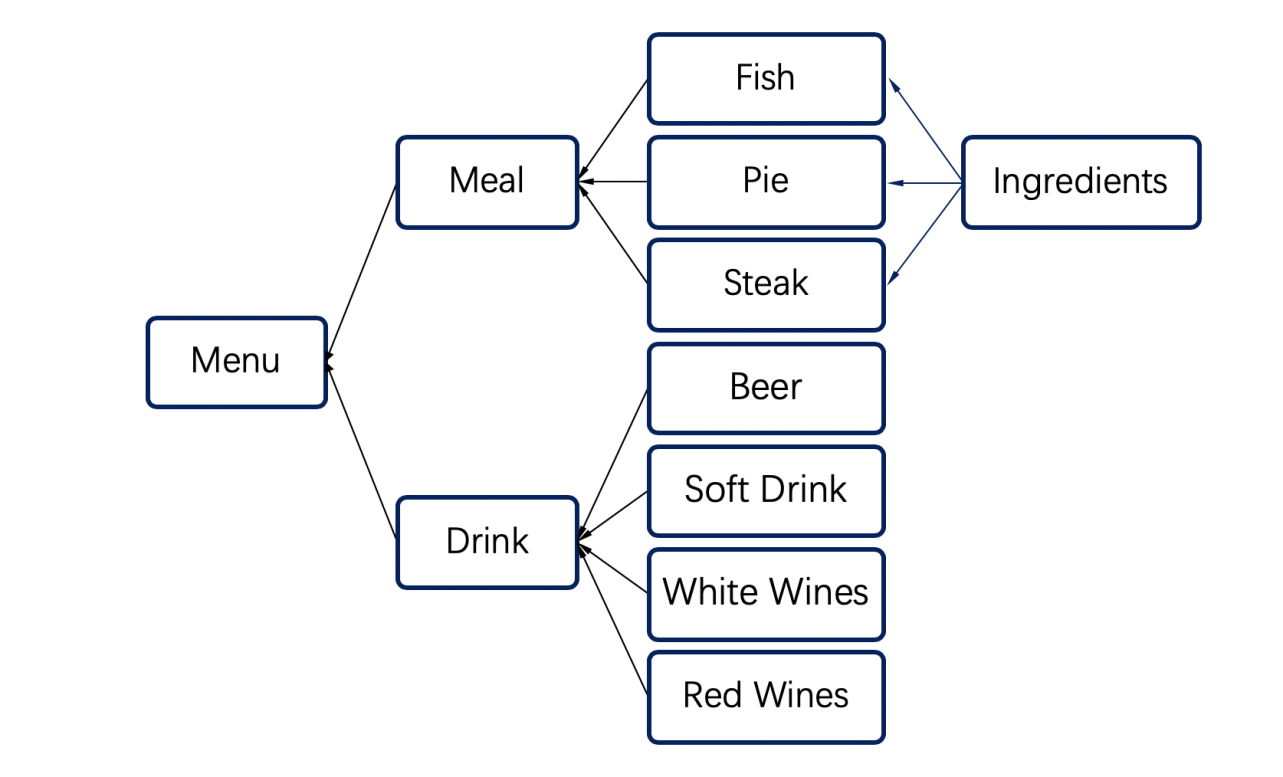


Figure 1 A Full Menu

**Meals（Package）**

There are totally 5 classes in packages in Meals. Which are Meal, Fish, Pie, Steak and Ingredient. The UML of it is show as figure 2.

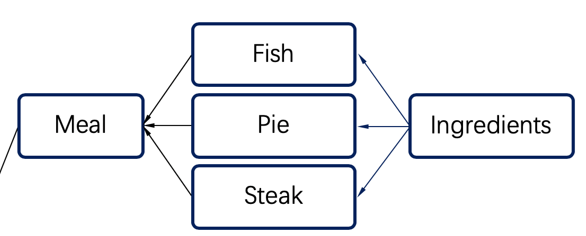


Figure 2 Meals

* *Meal*

Meal is an abstract class. The member variables include the following:

Table 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Price | Type | Ingredintes | Isvegetarian | Isvgan | count |

The name and type (*pie, fish or steak)* are determined by the restaurant manager's input of the dish name through the method public meal (string name, string type).

I*ngredients[]* is the ingredients list price used to store this dish. The price is calculated by price(). It is obtained by multiplying the price of each ingredient by 1.3 (20% labor cost, 10% profit). Variables of *Isvegetarian* and *isvgan* are obtained by checking whether the ingredients are all (V) or (VV) by the functions *isvegetarian()* and *isvgan().*

*Contains()* can determine whether the ingredient is in the main food by entering the name of the ingredient. *Addingredients()* can add an ingredient to the current main food object.

Finally, the abstract method, *drinksrecommendation(),* must be implemented by its subclass inheritance.

IngredientsTo()可以用来将Meal中Ingredients对象的基本信息以一个特定的格式输出，输出的信息包括食材名，食材用量，以及食材的单价单位是/kg

Meal类还提供了 getName（）可以获取Meal的名称，getPrice获取Meal的售价，getIngredients（）能够返回食材的数组对象，getCount（）方法能够返回这个Meal用了多少种食材，

* Pie, Fish, Steak

They inherited the meal class and overridden the *drinksrecommendation()* method. Class Pie will return a *beers* object, which recommends beers as the drink when the main

food is pie. Same as Pie, Class meal whill return a WhiteWineobject.

* Ingredients

*Ingredients* is an public class. The member variables include the following:

Table 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | cost | isVegetarian | isVegan | consumption | unitPrice |

The quantity and unit price of the food material class need to be entered when instantiating its object. *Isvegetarian()* and *isvegan()* are used to judge whether the ingredients are vegetarian. *Addcost()* method to calculate the cost of this ingredient.

在实例化Ingredients对象的时候 需要传入name consumption 和的单价这三个参数，这三个值会记录在成员变量中，同时调用Addcost（）方法计算总价，赋值到cost，并且调用*Isvegetarian()* and *isvegan()*  来判断是否是纯素食或者是素食，返回true or false给成员变量并记录，方便后面用于判断Meal是否是素食

Ingredients类里面还提供了getComsumption（）能够返回这个食材的用量，getUnitPrice（）返回这个食材的单价，getIsVegetarian（）返回这个食材是否是素食，getIsvegan（）返回这个食材是否是纯素食，getName（）返回这个食材的名字

**Prices**

The final sale price of meal depends on the price of the ingredients that make up the meal. The total cost of each ingredient is calculated as above price plus 20% of the staff expenses and 10% of the revenue, that is, the sale price of the meal is 1.3 times the total price of the ingredients

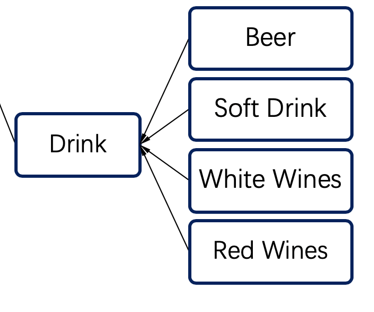


Figure 3 Prices

**Drink**

Package drivers contains five classes, which are driver, beer, soft drinks, white wines and red wines. The main attributes of class *drink* are *name* and *price*, it is the parent of all different drinks. And it can be inherited by other specific drink classes.

This design can simplify the code, and provide setprice() and setname () functions for each different type of drink. For example, when initializing a beer object, setprice() and setname() methods will be called automatically to set the price we set in advance for the beer.

**Menu（Main）**

Menu is the core process of the whole project. In this class, four functions, *addPie ()*, *addFish(),* *addDrink(),* and *addSteak()* are provided to add items to the menu object. The display method is also provided to display the content of the entire menu.

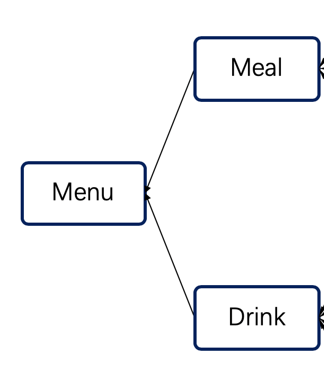


Figure 4 Menu

First, it will read in the data. Create an *ingredient* object and add it to the *ingredients* list. Then create the corresponding *meal* object, repeat the process until all the *ingredients* are added, and then add a specific drink to the menu by instantiating a *drink* object. Finally, call the *display()* function to display the menu.

**Conclusion**

The main function of the whole project is to input information in a certain format and then output a menu in a fixed format. Information of the same kind will be output to the adjacent area and the appropriate price will be calculated automatically. And it will automatically recommend the appropriate drink. Every time a new product needs to be put on the shelf, you only need to add the information of the product to the end of the txt file.

*Java* is an object-oriented programming language, which has three characteristics: inheritance, polymorphism and abstraction. And Class is the carrier to encapsulate the properties and behaviors of objects

The whole project makes good use of the above three characteristics to make the project simple, efficient, easy to update and maintain.