Project – Due May 28th

**Introduction to Programming**

The assignment can be done **individually** or in teams of up to **three**. Submit one assignment per team of two via Omnivox and NOT MIO. Assignments sent via MIO will be deducted marks. Assignments must be done alone or in groups and collaboration between individuals or groups is strictly forbidden.

**General Guidelines When Writing Your Programs:**

Include the following in comments at the top of your program:

# -----------------------------------------------------

# Assignment (include number)

# Written by: (include your name(s) and student ID(s))

# Short Description of your project/code and how you designed it.

# -----------------------------------------------------

Throughout your program, include comments in your program describing the main steps.

**Part One: Pizza Class:**

Create a Pizza class which stores information about a single pizza. It contains the following:

-Instance variables to store the size of the pizza (small, medium or large), (1)the number of cheese toppings, (2)the number of pepperoni toppings, and the (3) number of mushroom toppings.

- Constructor: that takes four arguments and sets all of the corresponding instance variables

and that initializes all instance variables to the zero of their type.

- Methods to get (accessor) and set (mutator) each instance variable individually.

- A method called **calcCost()** that returns the cost of the pizza. Pizza cost is determined by: Small: $10 + $2 per topping Medium: $12 + $2 per topping Large: $14 + $2 per topping.

-A method which returns a string for the pizza size, quantity of each topping, and the pizza cost as calculated by the following example, a large pizza with 1 cheese, 2 pepperoni and 1 mushroom toppings should cost $22 (can overwrite \_\_str\_\_).

**Part Two: DeluxePizza**

Take the *Pizza* class from the above and modify it in the following way:

* Rename it to *DeluxePizza*.
* Add a *stuffedWithCheese* (boolean) attribute to the class.
* Add a *veggieTopping* (integer) attribute to the class. This attribute keeps track of the number of vegetables toppings excluding mushrooms.
* Add a class attribute *numberOfPizzas* (integer) which will keep track of the number of

*DeluxePizza* objects created (this is not an instance variable).

* Update the existing constructor such that it increments the *numberOfPizzas* by one(+ 1) each time it is called and requires 6 arguments instead of 4 to accommodate the two new attributes (*stuffedWithCheese and veggieTopping)*.
* Modify the constructor to set the new attributes to “zero” and to increment the

*numberOfPizzas* attribute by one.

* Add an accessor and mutator method for the *stuffedWithCheese* and the *veggieTopping*

attributes.

* Add an accessor method for the *numberOfPizzas* attribute. (No mutator attribute.)
* Modify the *calcCost()* method to add $2, $4 or $6 dollars for *cheese stuffed dough* depending on the size of the pizza as well as an extra $3 per vegetable toppings.
* Modify the *\_\_str\_\_* method so that when an object of type *DeluxePizza* is printed all of the attributes are displayed in the following format:

Pizza #

Pizza size:

Cheese filled dough: Number of cheese toppings:

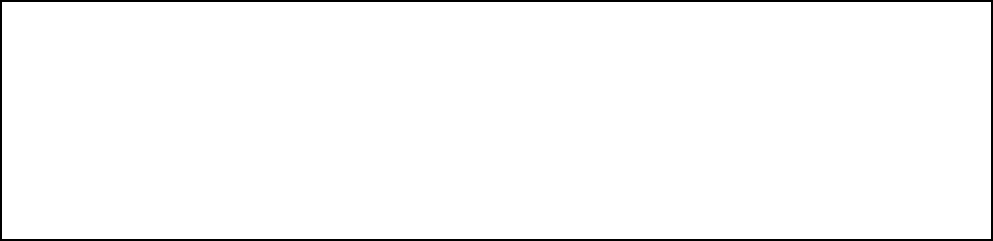
Number of pepperoni toppings: Number of mushroom toppings: Number of vegetable toppings:

## Part Three

Papa John, the owner of Papa Johns Pizzeria, needs help in keeping track of the number and types of pizzas he has cooked on a given day and to produce statistics regarding the choices her customers made. You task is to write a Python application which will contain the following methods..

1. **a main driver**, that will:
   1. Display a welcome message
   2. Prompt Papa John for the maximum number of pizzas he can prepare on that day and create an empty list called ***todaysPizzas***that will have the potential of keeping track of that many *DeluxePizza* objects.
   3. Display a main menu (fig 1) with the following

choices. If Papa John enters an invalid choice keep prompting him until he enters a valid choice.



Papa John, what do you want to do?

1. Enter a new pizza order (password required)
2. Change information of a specific order (password required)
3. Display details for all pizzas of a specific size (s/m/l)
4. Statistics on today’s pizzas
5. Quit

Please enter your choice >

Fig 1. Main menu

When **option 1** of the main menu is selected:

* Prompt user for the password. (Make sure you have a constant variable

containing the password “**deluxepizza**” - don’t use any other password. John has 3 tries to enter the correct password. You know with greasy hands is not always easy to type the correct password on the 1st try. After the 3rd illegal entry, the main menu in figure 1 is displayed again.

* If the correct password is entered, ask John how many pizzas he wants to enter. Make sure that he is not going over her limit for that day. If he is within the limit then add the pizzas otherwise tell Papa John that he only has enough ingredients to make the number of remaining places in the list and add them. (You decide how to input the information for each pizza added).

When **option 2** of main menu is selected:

* Prompt Papa John for his password. (Same as option 1). Again Papa John has 3 tries to enter the correct password. After the 3rd illegal entry, the main menu in figure 1 is displayed.
* Ask Papa John which pizza number he wishes to update. The pizza number is the index in the list ***todaysPizzas***. If there is no *DeluxePizza* object at the specified index location display a message on the screen, and ask Papa John if he wishes to enter another pizza number or quit this operation and go back to the main menu. If the pizza number exists, display on the screen all of the current information for that pizza in the following format (same as ***\_\_str\_\_***plus the price of the pizza):

Pizza # Pizza size:

Cheese filled dough: Number of cheese toppings:

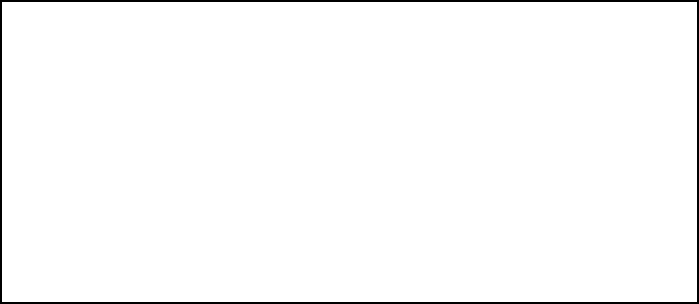
Number of pepperoni toppings:

Number of mushroom toppings:

Number of vegetable toppings:

Price: $

Then ask Papa John which attribute he wishes to change by displaying the following menu.



Papa John, what would you like to change?

1. Size
2. Cheese filled or not
3. Number of cheese toppings
4. Number of pepperoni toppings
5. Number of mushroom toppings
6. Number of vegetable toppings
7. Quit Enter choice >

Fig 2. Update menu

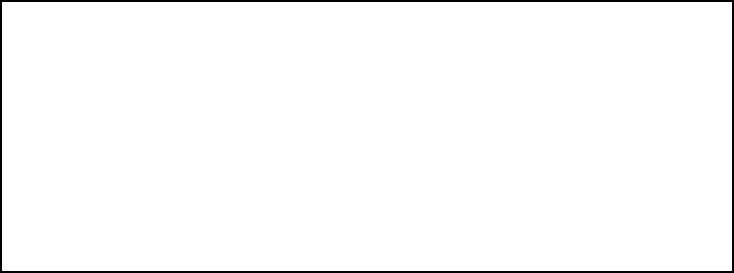
Once Papa john has entered a correct choice, update the attribute and display again all of the attributes to show that the attribute has been changed. Keep prompting Papa John for additional changes until he enters 7 (Quit). Each time Papa John is prompted for a choice make sure that he enters a number from 1 to 7, otherwise keep prompting him until he enters a valid number.

When **option 3** of main menu is selected:

* prompt the user for the size of the pizza he is interested in and display all information for all pizzas of that size by invoking the method *pizzasOfSize()* (see specification in step 2 below) as well as the number of pizzas of the requested size produced. Once done display the main menu (fig. 1) again.

When **option 4** of main menu is selected:

* display the following menu:



Papa John, what information would you like?

1. Cost and details of cheapest pizza
2. Cost and details of most costly pizza
3. Number of pizzas sold today
4. Number of pizzas of a specific size
5. Average cost of pizzas
6. Quit

Enter your choice >

Fig. 3 Statistic menu

* Prompt the user for a choice (making sure it is a valid choice). Perform the necessary action using the methods listed in steps 2 to 7 below when possible, or write the necessary code using the methods of class *DeluxePizza*. Keep prompting Papa John for choices until he decides to quit, at which point you should display the main menu (fig. 1).

When **option 5** of the main menu is selected:

* + display a closing message and end the program

1. A separate method in the driver called *pizzasOfSize*() which needs one argument of type String (the size) and which will display on the screen the pizzas of the requested size. The pizza number in the display should reflect the index of the location of each pizza in the list **todaysPizzas**.

List of *size* pizzas sold today:

Pizza # Pizza size:

Cheese filled dough:

Number of cheese toppings:

Number of pepperoni toppings:

Number of mushroom toppings Number of vegetable toppings:

Price: $price

Pizza # Pizza size:

Cheese filled dough:

Number of cheese toppings:

Number of pepperoni toppings:

Number of mushroom toppings Number of vegetable toppings:

Price: $price

………..

Our Chef, made *xx size* pizzas today!

1. A separate method outside the class called *cheaperThan()* which will list the number (index) and the price of all pizzas less than the requested price.
2. A separate method outside the class called *lowestPrice()* which will find and return the index of the pizza with the lowest price in the list.
3. a separate method outside the class called *highestPrice()* which will find and return the location of the pizza with the highest price in the list (step f).
4. a separate method outside the class called *numberOPizzasOfSize()* which will return the number of pizzas of the specified size.

|  |  |
| --- | --- |
| **Part a** (Class DeluxePizza) | **9 pts** |
| Default constructor | 3 pt |
| Accessor/mutator method for attributes | 3 pt |
| \_\_str\_\_ & other methods | 13pts |
| **Part b** (Driver & other methods) | **45 pts** |
| Driver (outside pizza class) | 3 pt |
| Handling of password | 3 pt |
| Handling of option 1 | 3 pt |
| Handling of option 2 | 3 pt |
| Handling of option 3 | 3 pt |
| Handling of option 4 | 15pts |
| Handling of option 5 | 3 pt |
| Method **pizzasOfSize()** | 3 pt |
| Method **cheaperThan()** | 4 pt |
| Methods **lowestPrice() & highestPrice()** | 4 pt |
| Method **numberOfPizzasOfSize()** | 3 pt |
| **Something Extra You Added** | **4 pts** |
| Extra features of your choosing - more  menu options, additional users, etc. | 2 pt |
| Extra features of your choosing - more menu  options, additional users, etc. | 2 pts |
| **Total** | **60 pts** |

\*60% of final mark