Concordia University
Computer Science & Software Engineering
Faculty of Engineering and Computer Science

**Concordia University**
**COMP 248 – Fall 2015**
# Assignment 4 - REVISED

| | |
|---|---|
| **Due Date:** | By 11:59pm Wednesday December 2, 2015 |
| **Evaluation:** | 5% of final mark (see marking rubric at the end of handout) |
| **Late Submission:** | none accepted |
| **Purpose:** | The purpose of this assignment is incorporate arrays of objects and the different types of statements we have seen since the beginning of the course into one more elaborate scenario. |
| **CEAB/CIPS Attributes:** | Design/Problem analysis/Communication Skills |

## General Guidelines When Writing Programs:
Please refer to the handout of Assignment #1.

## Part a)
Take the *Pizza* class from assignment # 3 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 static attribute *numberOfPizzas* (integer) which will keep track of the number of *DeluxePizza* objects created.
- Update the existing non-default constructor such that it increments the *numberOfPizzas* by one each time it is called and requires 6 arguments instead of 4 to accommodate the two new non-static attributes.
- Modify the default constructor to set the new attributes to "zero" and to increment the *numberOfPizzas* attribute by one.
- Add a copy constructor which will 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 *equals()* method so that 2 pizzas are considered equal if they have the same non-static attributes.
- Modify the *toString()* 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 b)**

**Mama Nancia**, the owner of **Mama Nancia** 's Pizzeria, needs help in keeping track of the number and types of pizzas she has cooked on a given day and to produce statistics regarding the choices her customers made. You task is to write a Java application which will contain the following methods..

1. **a main method**, that will:
    a. Display a welcome message
    b. Prompt **Mama Nancia** for the maximum number of pizzas she can prepare on that day and create an empty array called *todaysPizzas* that will have the potential of keeping track of that many *DeluxePizza* objects.
    c. Display a main menu (fig 1) with the following choices. If **Mama Nancia** enters an invalid choice keep prompting her until she enters a valid choice.

```
Mama Nancia, 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 **Mama Nancia** for her password. (Make sure you have a constant variable containing the password "deluxepizza" - don't use any other password as it will be easier for the marker to check your assignments). **Mama Nancia** 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 **Mama Nancia** how many pizzas she wants to enter. Make sure that she is not going over her limit for that day(size of array *todaysPizzas*). If she is within the limit then add the pizzas otherwise tell **Mama Nancia** that she only has enough ingredients to make the number of remaining places in the array and add them. (You decide how to input the information for each pizza added).

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

- Prompt Mama Nancia for her password. (Same as option 1). Again **Mama Nancia** has 3 tries to enter the correct password. After the 3rd illegal entry, the main menu in figure 1 is displayed.
- Ask **Mama Nancia** which pizza number she wishes to update. The pizza number is the index in the array *todaysPizzas*. If there is no *DeluxePizza* object at the specified index location display a message on the screen, and ask **Mama Nancia** if she wish 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 **the pizza number (which is the array index),** all of the current information for that pizza in the following format (same as *toString()* 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 **Mama Nancia** which attribute she wishes to change by displaying the following menu.

```
Mama Nancia, 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 **Mama Nancia** 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 **Mama Nancia** for additional changes until she enters 7. Each time **Mama Nancia** is prompted for a choice make sure that she enters a number from 1 to 7, otherwise keep prompting her until she enters a valid number.

When **<u>option 3</u>** of main menu is selected:

- prompt **Mama Nancia** for the size of the pizza she is interested in and display all information for all pizzas of that size by invoking the static 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 **<u>option 4</u>** of main menu is selected:

- display the following menu:

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

When **<u>option 5</u>** of the main menu is selected:

- display a closing message and end the driver.

2. A separate static 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 (there is a static method below to help). The pizza number in the display should reflect the index of the location of each pizza in the array 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 Mama Nancia, you made xx size pizzas today!
```

3. A separate static method in the driver called *cheaperThan()* which will list the number (index) and the price of all pizzas less than the requested price.

4. A separate static method in the driver called *lowestPrice()* which will find and return the index of the pizza with the lowest price in the array.

5. a separate static method in the driver called *highestPrice()* which will find and return the location of the pizza with the highest price in the array (step f).

6. a separate static method in the driver called *numberOPizzasOfSize()* which will return the number of pizzas of the specified size.

## Submitting Assignment 4

- Make one zip file that includes all of the .java and .html files.
- Naming convention for zip file: Refer to the handout for Assignment #3.
- Submit your zip file as **Programming Assignment #4**

## **Evaluation Criteria for Assignment 4** (20 points)

| | |
|---|---|
| **Part a** (Class DeluxePizza) | **5 pts** |
| Default & copy constructors | 1 pt |
| Accessor/mutator method for static attribute | 2 pts |
| Modified equals & toString methods | 2 pts |
| **Part b** (Driver & other static methods) | **15 pts** |
| Driver /main method | 1 pt |
| Handling of password | 1 pt |
| Handling of option 1 | 1 pt |
| Handling of option 2 | 1 pt |
| Handling of option 3 | 1 pt |
| Handling of option 4 | 5 pts |
| Handling of option 5 | 1 pt |
| Static method **pizzasOfSize()** | 1 pt |
| Static method **cheaperThan()** | 1 pt |
| Static methods **lowestPrice() & highestPrice()** | 1 pt |
| Static method **numberOfPizzasOfSize()** | 1 pt |
| **Total** | **20 pts** |