ICSI201/ECE141 Fall 2022 – Project 3
Pizza Shop

Due Sunday, December 4th at 11:59pm via Blackboard
Note, this is a firm deadline, and no extensions will be given (see syllabus for deadline policies).

(100 points)

ASSIGNMENT OVERVIEW

The goal of this project is to practice the programming knowledge and skills that you have acquired so far to implement a pizza ordering system. The user program will provide the user with a menu for ordering pizzas, selecting sizes, and selecting toppings. The menu also provides the ability to order specialty pizzas that have custom configurations and rules that must be incorporated into the program's logic.

Objectives:

This project focuses on the following learning objectives:

- Implementing logic using more complex Java objects and classes
- Practicing debugging larger, more sophisticated programs
- Applying knowledge of arrays or ArrayLists into your program solution
- Applying inheritance to avoid duplication of code logic

Reading:

Read and review the following chapters to assist you in completing this project.

- **Chapter #7:** Arrays & the ArrayList Class
- **Chapter #8:** A Second Look at Classes and Objects
- **Chapter #10:** Inheritance

~~**GRADING DEADLINES:**~~

~~**This assignment is to be completed individually.** Any assignments flagged for group work will be considered plagiarism. See my policy on cheating for more details.~~

~~**Important NOTE on Formatting and Code Compliance:** You should make sure your code compiles in your own IDE before submitting. As a reminder, please submit your .java files. If you submit .class files, we will not be able to grade your program and you will receive 0 points.~~

The breakdown of points for this assignment is in the table below.

Grading Item	Points
The software meets the specification	
- Correctly implements Pizza class and enumerations (Part #1)	15
- Correctly implements PizzaOrder class (Part #2)	15
The design follows the principles of OOP	
- Correctly incorporates inheritance into solution for VeggieLoversPizza and MeatLovers Pizza classes (Part #3)	15
The software is well documented	
- Includes good documentation (appropriate comments)	5
The program works correctly	
- Correctly displays menu based on user input (Part #4)	15
- Correct functionality based on user input	15
The implementation is completed and well-tested	
- Solution is robust, handles both provided and unprovided test cases, and performs appropriate error checking	15
The programming style is efficient	
- Follows proper coding style (good indentation and self-documenting variable names)	5

Procrastination Warning: This assignment has multiple parts and will take significant amount of time to complete. No matter how confident you feel now, when you get in the details you will have questions. The more time you have to approach your instructors and TAs the more successful you will be. Instead of assuming things should be done a particular way, ask questions! Use Piazza to ask clarifying questions (never post your code publicly), discuss your approach with peers and use the instructors' and TAs' office hours to get help if you are stuck.

Assignment Turn-in: You must submit your source code, so we can assess your implementation. In other words, you must submit .java files. If you submit a .class file, you will receive 0 points. You must submit the following .java files. If you decide to use enumerations as part of your design, then those should also be submitted (Size.java and Topping.java)

1. Driver.java (contains main)
2. Pizza.java
3. PizzaOrder.java
4. Size.java (*optional*)
5. Topping.java (optional)
6. VeggieLoversPizza.java
7. MeatLoversPizza.java

Cheating policy: Cheating is not tolerated! We will be comparing your code against that of other students in the class and against similar assignments from online student support platforms. All students involved in a cheating accident (i.e. whether sharing or receiving code) will be penalized. Please, read the syllabus for our policies on cheating and refer to slides from Lecture 1 for examples of what we consider cheating, as those are not only limited to what you submit. Students caught cheating will receive 0 points for the assignment and will be reported.

ASSIGNMENT DETAILS:

Part #1: Pizza Class

Referring to the UML diagram below, build a **Pizza** class. It is recommended using enumerations called **Toppings** and **Size**. You can download some sample code defining the Toppings and Size enumerations and a driver class here to assist you here: (Topping.java, Size.java, EnumDriver.java).

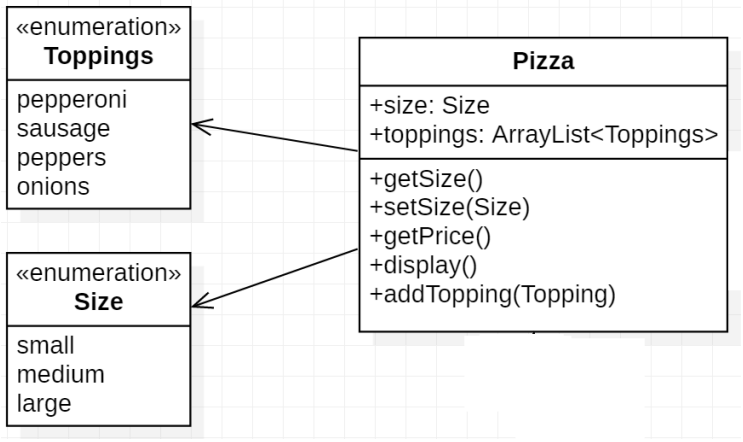

Figure 1. UML diagram of the `Pizza` class.

Alternative design approaches: Students may (at their discretion) choose to:

- Avoid using enumerations and implement the same desired functionality by treating the size and toppings as String data types. If you follow this approach, your mutator (or setter) methods must prevent setting size to anything other than (small, medium, or large) and prevent adding a topping to the array other than (pepperoni, sausage, peppers, or onions).
- You can choose to implement the toppings data member as either an ArrayList or an array.

Define fields **size** and **toppings** within the Pizza class with data types consistent with the UML design shown above (*or the alternative design options discussed above*).

Implement the following methods in addition to the other accessor and mutator methods:

- **addTopping**: adds one additional topping into the toppings array *(or ArrayList)*. Display an error message if the topping has already been added and do not add the topping.
- **getPrice**: returns price of the pizza. Assume that price is $1 per toppings, plus $10 for a small, $15 for a medium, and $20 for a large.
- **display:** Displays to the console the snapshot of the pizza. Follow the formatting standard shown below.

```
Size: MEDIUM
TOPPING:
  ONIONS
  PEPPERONI
Item Price: $17
```

Part #2: Pizza Order Class

Referring to the UML diagram below, build a **PizzaOrder** class.

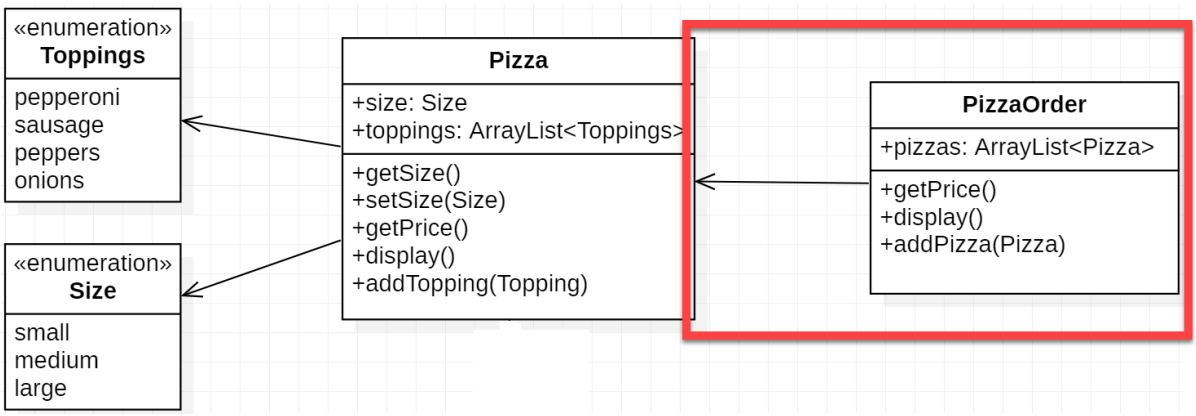

Figure 2. UML diagram of the `PizzaOrder` class.

Define a field that stores an ArrayList of Pizzas (*It's also acceptable to implement this as an array and can assume a max order size of 20 pizzas*). This class should contain the following methods:

- **addPizza**: adds one additional pizza into the array *(or ArrayList)*
- **getPrice**: Returns the price of all pizzas in the order
- **display**: Displays all the pizzas that have been added to the order following the formatting standards shown below. In addition to listing the order information, it should display the subtotal, the tax (based off an 8% sales tax rate), and the total that includes taxes. The tax rate should be defined as a private constant variable in the Pizza class.

```
------------------------
   ORDER RECEIPT
------------------------
  Size: LARGE
  TOPPING:
    SAUSAGE
    PEPPERS
  Item Price: $22
------------------------
  Size: MEDIUM
  TOPPING:
    ONIONS
  Item Price: $16
------------------------
SUBTOTAL: $38
TAX $3.04
TOTAL: $41.04
```

Part #3: Veggie Lovers and Meat Lovers Pizza

Referring to the UML diagram below, build the **VeggieLoversPizza** and **MeatLoversPizza** classes, such that they inherit from the Pizza super class.

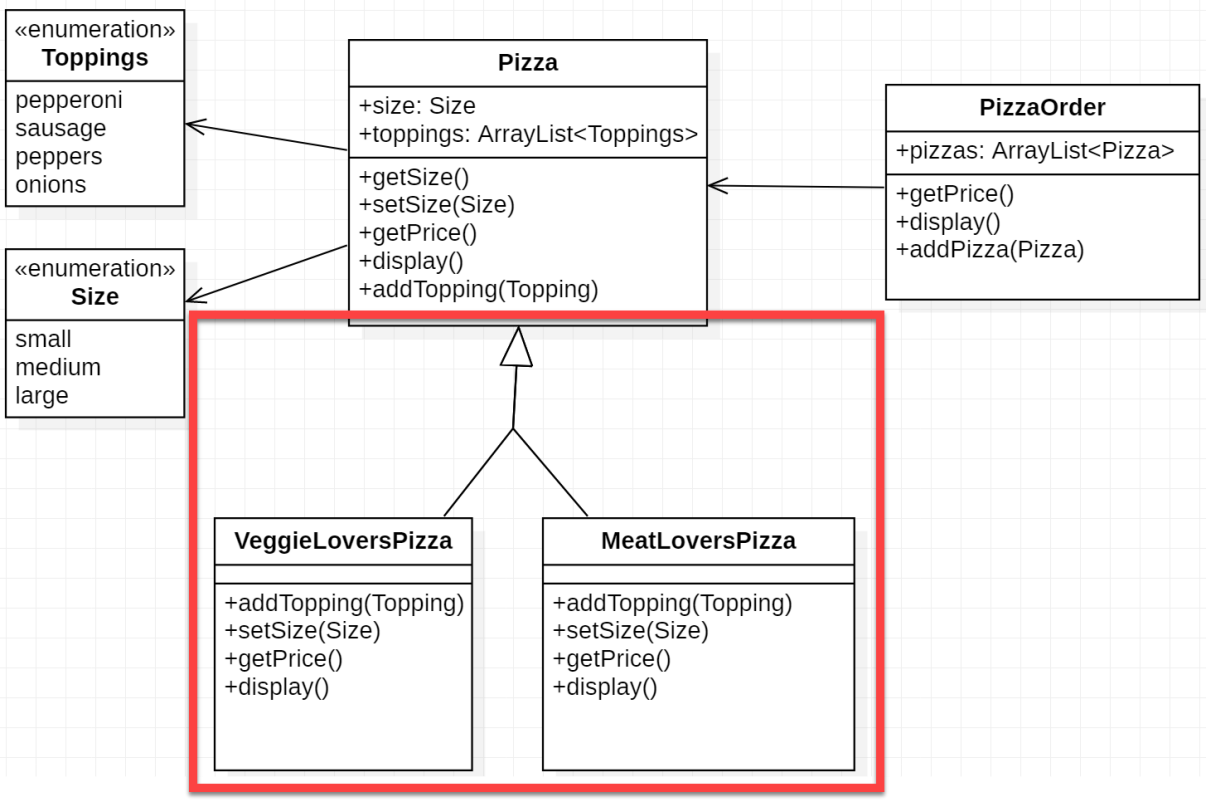

Figure 3. UML diagram of the `VeggieLoversPizza` and the `MeatLoversPizza` classes.

These classes represent specials that have specific sizes, toppings, and prices which cannot be changed. Override methods to implement the desired functionality documented below:

- VeggieLoversPizza:
  - Size: Large
  - Toppings: Onions and Peppers
  - Price: Special price $17
- MeatLoversPizza:
  - Size: Large
  - Toppings: Pepperoni and Sausage
  - Price: Special price $18

If someone attempts to modify these attributes above, they should be prevented from changing these configurations and an error message should be displayed.

Part #4: Driver Class (Menu)

Implement a driver class called **Driver.java**. Your driver class should have the following main menu.

```
MAIN MENU:
  (O)rder a pizza
  (S)pecials list
  (R)eceipt diplay
  (Q)uit
Enter Choice:
```

If the user selects 'O' for Order a pizza, they should be prompted to select the pizza size and toppings, as shown below. Follow formatting standards that are consistent with the following example. After the user completes entering the toppings, the display should return to the Main Menu.

```
-------------
MAIN MENU:
  (O)rder a pizza
  (S)pecials list
  (R)eceipt diplay
  (Q)uit
Enter Choice: O
-------------
SELECT PIZZA SIZE:
  (S)mall
  (M)edium
  (L)arge
Enter Choice: M
-------------
SELECT TOPPINGS:
  (P)epperoni
  (S)ausage
  (O)nions
  (B)ell peppers
  (D)one - no more toppings to add
Enter Choice: S
-------------
SELECT TOPPINGS:
  (P)epperoni
  (S)ausage
  (O)nions
  (B)ell peppers
  (D)one - no more toppings to add
Enter Choice: B
-------------
SELECT TOPPINGS:
  (P)epperoni
  (S)ausage
  (O)nions
  (B)ell peppers
  (D)one - no more toppings to add
Enter Choice: D
-------------
MAIN MENU:
  (O)rder a pizza
  (S)pecials list
  (R)eceipt diplay
  (Q)uit
Enter Choice:
```

If the user selects ‘S’, they should be prompted to select one of the two special pizzas. After that selection, the display should return to the Main Menu (as shown below).

```
-------------
MAIN MENU:
  (O)rder a pizza
  (S)pecials list
  (R)eceipt diplay
  (Q)uit
Enter Choice: S
-------------
SELECT SPECIAL:
  (V)eggie Lovers Pizza
  (M)eat Lovers Pizza
  (B)ack to Main Menu
Enter Choice: V
-------------
MAIN MENU:
  (O)rder a pizza
  (S)pecials list
  (R)eceipt diplay
  (Q)uit
Enter Choice:
```

After the user selects their regular pizza and/or pizza special selections, the user can enter ‘R’ to display the Order Receipt.

```
-------------
MAIN MENU:
  (O)rder a pizza
  (S)pecials list
  (R)eceipt diplay
  (Q)uit
Enter Choice: R
------------------------
   ORDER RECEIPT
------------------------
  Size: MEDIUM
  TOPPING:
    SAUSAGE
  Item Price: $16
------------------------
Veggie Lovers Special
  Size: LARGE
  TOPPING:
    ONIONS
    PEPPERS
  Item Price: $17
------------------------
SUBTOTAL: $33
TAX $2.64
TOTAL: $35.64
-------------
```

Selecting ‘Q’ for Quit will terminate the program.

```
-------------
MAIN MENU:
  (O)rder a pizza
  (S)pecials list
  (R)eceipt diplay
  (Q)uit
Enter Choice: Q
Exiting Program....
```

Appendix – Testing Examples

Below we provide a few snapshots of full runs and erroneous input. Please compare your program’s output to the examples below to ensure that it produces both the same prompts and results.

Example 1: Ordering one regular pizza and then printing the receipt (see example #1)

Example #2: Handling erroneous inputs from the main menu (see example #2)

Example #3: Handling erroneous inputs from the size menu and toppings menu (see example #3)

Example #4: Handling erroneous input from the specialty pizza screen (see example #4)

Example #5: Ordering two regular pizzas and a special pizza, then printing the receipt (see example #5)