Project #4 (160 points)

Due Date

Monday, November 27, by 11:59pm (Thanksgiving break is November 23 – 26)

Submission

- Zip your project folder and submit it to Canvas. The zipped file MUST include the following grading items.
  (1) **Source folder src**, containing the folders below. [130 points]
    - Java package folder, including all the Java source files (the "Model" and the "Controller")
    - Resource folder, containing all the fxml files (the "View") and image files.
  (2) Class Diagram. [10 points]
  (3) A JUnit test class for BuildYourOwn class. [15 points]
  (4) Javadoc folder. [5 points]
- The submission button on Canvas will disappear after **November 27, 11:59pm**. It is your responsibility to ensure your Internet connection is good for the submission, and you have enough time for the submission. **You get 0 points** if you do not have a submission on Canvas. **Projects sent through the emails will not be accepted**.

Project Description

Your team will develop a software for RU Pizza to manage the pizza orders. Your team must use JavaFX to develop the GUIs for taking orders, placing orders, and cancelling orders. The pizzeria offers five specialty pizzas, Deluxe, Supreme, Meatzza, Seafood and Pepperoni, with store customized toppings as shown in the table below. The pizzeria also offers "build your own" pizzas, which allow the customers to choose the toppings they like, up to 7 toppings. The store charges $1 each for extra cheese and extra sauce. The store staff will be the one using the software to take the orders from the customers. Each order is uniquely identified by an order number. For simplicity, the software system will not keep track of the dates and customer information of the orders.

Pizza Types	Toppings	Sauce	Price
Deluxe	Sausage, pepperoni, green pepper, onion, mushroom	tomato	small: $14.99 medium: +$2 of small; large: +$4 of small
Supreme	Sausage, pepperoni, ham, green pepper, onion, black olive, mushroom	tomato	small: $15.99 medium: +$2 of small; large: +$4 of small
Meatzza	Sausage, pepperoni, beef, ham	tomato	small: $16.99 medium: +$2 of small; large: +$4 of small
Seafood	Shrimp, squid, crab meats	alfredo	small: $17.99 medium: +$2 of small; large: +$4 of small
Pepperoni	pepperoni	tomato	small: $10.99 medium: +$2 of small; large: +$4 of small
Build your own	Select from 13 toppings, At least 3 toppings, up to 7 toppings	tomato/alfredo	small: $8.99 medium: +$2 of small; large: +$4 of small The prices above include 3 toppings; for each additional topping, add $1.49.

Below is the list of requirements given by the store manager.

1. The system shall allow the user to select the specialty pizzas or build your own pizza.
2. When a pizza type is selected, the user can select a size, small, medium, or large, and the user can add extra sauce and extra cheese.

3. When build your own pizza is selected, the system shall provide 13 toppings for customization.
4. When the user is customizing a build your own pizza, the user can add/remove toppings, with a maximum of 7 toppings, and the user can select tomato sauce or alfredo sauce.
5. When the user is ordering specialty pizzas, the system shall provide the five specialty pizza options, Deluxe, Supreme, Meatzza, Pepperoni and Seafood.
6. When a specialty pizza is selected, the system shall display the image of the specialty pizza and the store customized toppings. The user cannot change the toppings, but can select, size, extra sauce and extra cheese.
7. When the user is ordering/customizing any pizza, the system shall display the running pizza price with 2 decimal places while the user is changing the pizza size, toppings, extra sauce and extra cheese.
8. The system shall allow the store staff to add one or more pizzas to an order or remove selected pizza from an order.
9. The store staff shall be able to check the detail of the current order before placing the order. These shall include the list of pizzas with the selected toppings, sauce/extra sauce, extra cheese, price for each pizza, total amount of the pizzas in the order, sales tax amount and the order total (total amount plus sales tax.) The tax rate in New Jersey is 6.625%. Sales tax amount = total amount of the pizzas * the tax rate.
10. The system shall be able to keep track of all the store orders placed by the store staff and allow the store staff to browse the store orders and cancel an order. These shall include displaying all the store orders by the order numbers, the order total for each order with 2 decimal places, and the list of pizzas in each order.
11. The system shall be able to export the store orders and save them in a text file. Each order shall include the order number, the list of pizzas with selected toppings, sauce/extra sauce, extra cheese, and the order total.

Project Requirements

1. You MUST follow the Coding Standard posted on Canvas under Week #1 in the "Modules". **You will lose points** if you are not following the rules.
2. You are responsible for following the Academic Integrity Policy. See the **Additional Note #14** in the syllabus. The consequences of violation of Academic Integrity Policy are: **(i) your group receives 0 (zero) on the project, (ii) the violation is reported, (iii) a record on your file of this violation.**
3. All methods should not exceed 40 lines, or **-2 points** for each violation, **maximum 5 points off**.
4. You MUST set the titles of all the "stages" (titles for the windows) or **-2 points each.**
5. You are **NOT ALLOWED to use System.out** (write to console) or **System.in** (read from console) ANYWHERE in ALL CLASSES. All read and write must be done through the GUIs, **or you will lose 3 points for each violation, with a maximum of losing 10 points**.
6. Your software must provide 5 GUIs listed below, **-5 points** for each GUI missing. The user can navigate between the GUIs to add/remove pizzas to/from an order, review/place the order, check all orders, and cancel an order.
   - **Main Menu**, which shall include the major functionalities: order specialty pizzas, order build your own pizza, browse current order, browse store orders.
   - **Build your own**, which provides the user to customize the pizza.
   - **Specialty pizzas,** which provides the options of ordering the specialty pizzas.
   - **Current order**, which allow the user to browse the order, including the list of pizzas with selected toppings, sauce/extra sauce, extra cheese, subtotal, sales tax, and order total. All dollar amounts must be displayed with 2 decimal places. The user can review the order, remove the selected pizza, and place the order.
   - **Store orders**, which allow the user to browse all the orders placed so far, including the details and the total amount of each order, with 2 decimal places. The user can select an order and cancel the order. The GUI shall display the remaining orders after the cancellation. The user can also export the store orders to a text file, which shall include the details of every order, consistent with the details displayed on the GUI.
7. For each GUI, you must create a .fxml file and its associated controller class. That is, you will have **5 .fxml files** and **5 controller.java files**. **-5 points** if this is not done properly.

8. You must meet the 11 requirements given by the store manager, or **-5 points** each requirement not met.
9. You MUST use a ComboBox to provide specialty pizza selection, or you will **lose 5 points**.
10. You MUST use Image Buttons and use a ListView for pizza toppings, or **-5 points** each violation.
11. You can use any Java library classes; however, you MUST meet the following project requirements.
    - All the instance variables defined in the controller classes must be "private", **-2 points** for each violation.
    - Must include the **abstract Pizza class** below. You must define subclasses for all pizza types, or **-5 points** each violation.

```
public abstract class Pizza {
    protected ArrayList<Topping> toppings; //Topping is a enum class
    protected Size size;   //Size is a enum class
    protected Sauce sauce; //Sauce is a enum class
    protected boolean extraSauce;
    protected boolean extraCheese;

    public abstract double price(); //polymorphism
}
```

    - Must include a PizzaMaker class below to create an instance of Pizza. This class should be used in the controller classes to create an instance of Pizza class based on a chosen pizza type, such as build your own, deluxe, supreme, meatzza, pepperoni or seafood. That is, this is the ONLY class that can "new" an instance of the subclasses. We are using a **design pattern** called **"Factory Method"** in this project. PizzaMaker class is a "Pizza Factory" responsible to create an instance of Pizza of any subtypes. This design pattern is to hide the details of creating the pizza objects and allow flexibility to add new pizza types in the future without affecting the client's code. You **will lose 10 points** if this is not done, or if you create an instance of Pizza without using this class. You CANNOT add any instance variables and you CANNOT add any other methods, or **-5 points.**

```
//create an instance of subclasses based on the chosen pizza type
public class PizzaMaker {
    public static Pizza createPizza(String pizzaType) { }
}
```

      To adhere to the design pattern, you CANNOT use the subclasses, such as Deluxe, Supreme, Meatzza, Pepperoni, Seafood and BuildYourOwn, anywhere in ALL other Java classes except the PizzaMaker class. **-5 points** for each violation. You should only use the Pizza class as the data type in all other Java classes.

    - Must include an **Order class**. An instance of this class has a unique order number and a list of pizza objects. **-5 points** if the class is not implemented or not used. All instance variables must be private or **lose 2 points**.
    - Must include a **StoreOrders class**. An instance of this class keeps the list of orders placed by the user and **a static integer that provides the next available order number**. **-5 points** if the class is not implemented or not used. All instance variables must be private or **lose 2 points**. You must include an **export() method** in this class to save the store orders to an external text file, **-5 points** if the method is missing in this class.
12. Create a JUnit test class for Build your own class, and test **the price() method**. Use the Black-Box testing technique to design the test cases and ensure the price() method performs properly with various numbers of toppings and different sizes. You MUST include at **least 5 test cases (5 test methods) to get the 10 points.**
13. You are required to **generate the Javadoc** after you properly commented your code. Your Javadoc must include the documentations for the constructors, private methods, and public methods of all Java classes (*.java files.) DO NOT include the *.fxml files, which are NOT java files. DO NOT include the JUnit test class. Generate the Javadoc in a single folder and include it in your project folder to be submitted to Canvas. You are responsible to **double check** your Javadoc after you generated them. The grader will navigate the Javadoc with the "index.html".

You will **lose 5 points** for not including the Javadoc, OR the grader cannot navigate your Javadoc through the "index.html".

14. Create a Class diagram for this project to document your design. You must include all Java classes in this project. DO NOT include the Java/JavaFX library classes, JUnit test class and .fxml files. The class diagram **is worth 10 points**.

System Testing

1. You are responsible to thoroughly test your software and ensure your software is meeting the project requirements. You will **lose 2 points** for each incorrect implementation, incorrect amount or incorrect information shown on the GUIs.
2. Your software must always run in a sane state and **should not crash in any circumstances**. You must catch all Java Exceptions. Your program shall continue to run until the user stops the program execution or closes the window. **You will lose 2 points** for each exception not caught.