INTPROG TB2: OBJECT ORIENTED PROGRAMMING

Individual Coursework 2017/18: Pizza Order System

Set: 26th February 2018

Electronic Submission: Submit a zipped BlueJ or NetBeans project folder to Moodle by **17:30 on Wednesday 21st March 2018**.

Demonstrations: In your practical class between Friday 23rd and Tuesday 27th March 2018.

Worth: 20% of the unit marks.

This piece of work is to be done individually. As the assessment will primarily be via a demonstration of your final program it will not be marked anonymously.

1. Aim

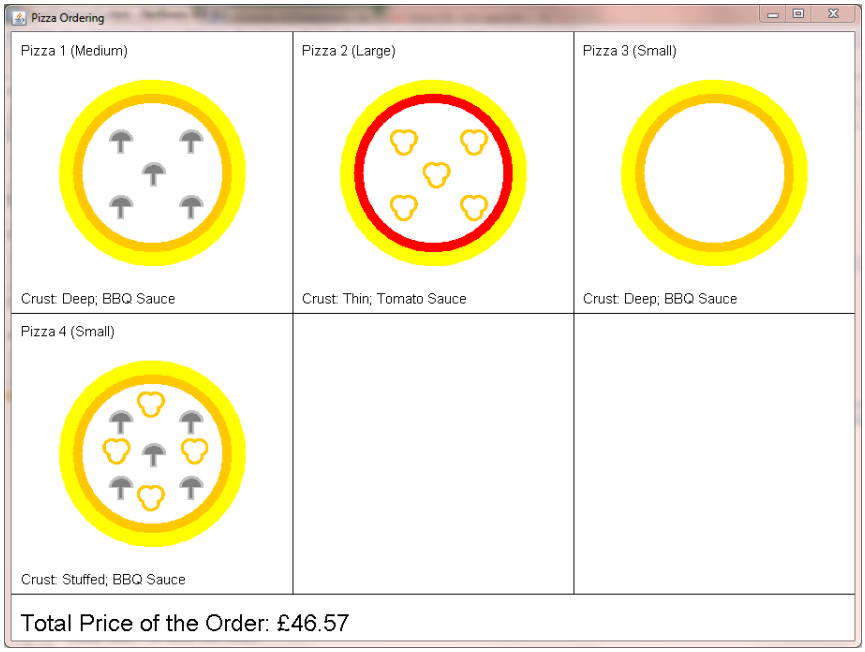
To develop object-oriented programming skills, including designing, documenting, testing, and debugging of Java code.

2. Description

Your task is to develop a Pizza ordering system. The ordering system will take inputs from the user on the console and then display a picture of the chosen pizza with the associated cost. This ordering system is slightly different to other ordering systems in that users are limited in their pizza choices. They will be limited to 6 pizzas and will only be able to choose 2 different types of topping, depending on your student number. The users will be able to choose their pizzas using a text-based interface. The total price of the order is displayed at the bottom of a graphical display screen.

The below example shows an order of the following 4 pizzas:

- A medium deep pan pizza, with BBQ sauce and mushrooms
- A large thin crust pizza, with tomato sauce and peppers
- A small deep pan pizza, with BBQ sauce
- A small stuffed crust pizza, with BBQ sauce, mushrooms, and peppers.



To help you get started, a project called "PizzaOrderSystem" is available on Moodle. This provides some of the functionality that the final program will need. However, much needs adding. Study this project and understand how the code in PizzaMain, OrderingSystem, and Pizza works before you start working on this coursework. You should not need to change the PizzaMain file.

2.1.Pizza Elements

The user will need to enter all the details of their chosen pizzas using a text-based interface, in the console. All the pizzas must be drawn within the provided display window, which will contain at most 6 pizzas, together with the total price of the order.

For each pizza, the user will have the following choices:

- The size of the pizza (small, medium, or large)
- The crust of the pizza (deep pan, thin crust, or stuffed crust)
- The sauce on the pizza (tomato, or BBQ)
- The toppings (the user will be able to choose whether to have no toppings, 1 topping, or 2 toppings).

2.1.1. Pizza Base and Sauce

The user should be able to choose from 3 sizes for each pizza:

- Small (10" diameter)
- Medium (12" diameter)
- Large (14" diameter)

In addition to the size, the user should be able to choose between 3 types of crust:

- Deep Pan
- Thin Crust
- Stuffed Crust

All the crusts come with a tomato sauce and mozzarella topping by default. However, for each pizza the user should have the option to change the sauce from tomato to BBQ, at an additional cost.

2.1.2. Pizza Toppings

The user has the choice of up to two toppings for their pizza. The pizza company will put 5 pieces of their first chosen topping on the pizza, and 4 of the second topping. The user is able to choose the same topping twice, in which case, there will be 9 pieces of topping on the pizza.

The below table shows all the possible toppings for the pizza:

Number	Topping Description	Topping Diagram
0	Pineapple	
1	Pepperoni	 *take care with this one, the inner circles alternate between pink and white*
2	Crispy Bacon	
3	Mushroom	
4	Prawn	
5	Jalapeño Slice	
6	Pepper	
7	Tuna	
8	Red Onion	
9	Anchovy	

You need to implement 2 toppings based on the last 2 numbers of your student ID. For example, if your student number is 888**14**, you will need to implement toppings 1 and 4, which are pepperoni and prawn, respectively.

If the last 2 numbers of your student ID are the same, you will need to subtract one of the numbers from 9, then implement the resulting number too. For example, if your student number is 888**11**, you will need to implement toppings 1 and 8 (as 9 – 1 = 8), which are pepperoni and red onion, respectively.

2.2.Pricing

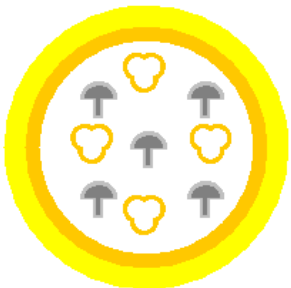
You will also need to calculate the total price of all the pizzas which have been ordered. To calculate the price of each pizza, the following pricing structure should be used:

Base Description	Price (per square inch)
Deep Pan	11p
Thin Crust	8p
Stuffed Crust	14p

Sauce Description	Price (single price for all pizza sizes)
Tomato	0p
BBQ	50p

Number	Topping Description	Price (per piece)
0	Pineapple	6p
1	Pepperoni	4p
2	Crispy Bacon	2p
3	Mushroom	5p
4	Prawn	6p
5	Jalapeño Slice	4p
6	Pepper	2p
7	Tuna	8p
8	Red Onion	3p
9	Anchovy	7p

For example, the following is a medium deep pan pizza, with BBQ sauce and both mushrooms and peppers:



The total price will be £13.27. This can be broken down as follows:

- Price of base: £12.44 (area of pizza is: 113.1; so the cost is 0.11 x 113.1)
- Additional cost for BBQ sauce: £0.50
- Cost of topping 1 (mushroom): £0.25 (5 x 5p)
- Cost of topping 2 (pepper): £0.08 (4 x 2p)

2.3.User Interface

The display screen has been provided in the starter project for this coursework – you should not change the layout of this screen. All your interactions with the ordering system should be completed using a text-based interface in the console (using the KeyboardInput class provided in the starter pack). The program's user interface should be easy to use, friendly, and robust. For example, if the user inputs invalid data the user should receive an appropriate error message, and be re-prompted until the entered data is valid.

3. Required Functionality

Basic Functionality (50% of the marks): The user should be able to enter up to a maximum of 6 pizzas, which should be displayed within the window provided. For each pizza, the user should be prompted to enter the following information (separately, not in one line):

- Size of the pizza
- Type of crust
- Whether they want to change the sauce from tomato to BBQ
- How many toppings they want
- First topping (if relevant)
- Second topping (if relevant)

The system should stop prompting once the order has reached 6 pizzas, or if the user decides to finish before 6 have been ordered. After each entry, the graphical display should be updated to show the new pizza, and the new total price.

Extended Functionality (20% of the marks): The system should be adapted so that users can change or remove any pizzas they have already entered. In addition, the user should be able to enter more than 6 pizzas – continuing until they are finished. The maximum number of pizzas on the screen should remain at 6, but the user should be able to change the page using the text-based interface.

4. Code Quality

After demonstrating the program's functionality, the member of staff will give you some feedback on the quality of your program code. Your program will be awarded marks based on: its overall structure (ensuring that the principles of object-oriented programming have been incorporated), its readability, and the quality of the algorithms used. In addition, you must use inheritance appropriately within the design of your program. There are lots of variations on pizzas and toppings. However, variations do not necessarily mean that inheritance is required. We only expect inheritance to be used in one place, so you need to think carefully about the most appropriate place to implement it.

You should not use parts of the Java language which have not been taught (for example, try/catch, switch statement, etc.). You can, however, use classes which are part of the Java standard libraries (such as ArrayList).

Even if your program works well, the code may obtain very few marks if its readability is poor, or it has been poorly designed.

5. Documentation

You will be required to produce documentation, using the JavaDoc format of commenting the classes. A brief description should be given for each class, constructor, and method. Both constructors and methods need to provide details about their parameters (where relevant). For methods which are not declared as void, details of the value returned should be provided.

6. Submission and Assessment

Submit via Moodle a zipped folder (called upXXXXXX.zip, where XXXXXX is your student ID number) containing the relevant BlueJ or NetBeans project by 17:30hrs on Wednesday 21st March 2018.

The primary assessment of this work will be via a demonstration, to a member of staff, given in your regular INTPROG practical session between Friday 23rd and Tuesday 27th March 2018. Failure to demonstrate your program will result in a mark of 0% even if you have submitted. Late submissions or demonstrations will be capped at 40% (subject to ECF amendment).

6.1. Marks Breakdown

Element	Percentage of Marks
Basic Functionality	50% of the marks
Extended Functionality	20% of the marks
Code Quality	20% of the marks
Documentation	10% of the marks

7. Advice on Completing the Coursework

You may ask me or any of the INTPROG teaching team for any help you may require - you will not be penalized for asking for help. We will not do the coursework for you but will help with any specific problems you have. You may also get help from other students taking the unit but if any submissions have an excessive amount of duplicate code the University's plagiarism procedures will be enforced. In simple terms - what you submit must be your own work not copied from, or done by, someone else.

All code must be professionally written (e.g. with JavaDoc comments and correct indentation) and will be assessed for:

- correctness
- appropriate use of language constructs
- style (commenting, indentation, etc.)

Develop your code in small parts. Test each part as you implement it. Compile your program frequently - don't add more than a handful of program statements without compiling it to check what you've done. Keep backup copies of working code as you complete each task. Don't reject well thought out code just because you get compilation errors - work out why you've got the error and correct it, don't throw away the good with the bad.

Think carefully about the design of each class you create - what attributes and operations should a class offer?

Don't be over ambitious and do not spend excessive time on this coursework. You should spend approximately 20-30 hours (3-4 full days). We do not want you to write your own versions of the Canvas, or KeyboardInput classes – you will not gain any extra credit for doing so.

If you have any doubt about what is required or believe there is any ambiguity in the coursework brief please email me with your query and I will post answers to all such queries on Moodle.