**CSC1021 – Project 2**

**Testing and Output Documentation**

**By**

**Joseph Whitten**

**140197916**

Program Design Overview

For this project I was asked to design and implement a prototype system that would allow me to create pizzas and define their type of base and toppings, the system must also allow for the calculations such as cost of the created pizza and whether or not it is suitable for vegetarians.

# What my code achieves

After testing my code I can confirm that I have met all the requirements and that I have successfully carried out all the tasks asked of me by the specification.

Requirements asked of me by the specification:

* **Creation of three classes (Pizza, PizzaBase and PizzaTopping):** The Pizza class correctly carries all the essential information about the pizza (its base and toppings), the class also provides the methods that can be called on a pizza object to calculate its cost and its suitability for vegetarians by checking the properties of the pizza’s base and toppings, and the method that prints the report of the information of the pizza including the name and corresponding price of each ingredient, suitability for vegetarians and total cost. Since this project would require the creation of many base and topping constants I have opted to use enums as the type for PizzaBase and PizzaTopping this allows me to easily define a collection of constants of each and make use of methods such valueOf() and values() to easily access and iterate over the objects, these enums correctly model the base and toppings of the pizza with their appropriate properties being defined on initialization by a constructor. The member variables for all these classes are private and are only accessible through the use of get and set methods.
* **Build a Pizza:** My BuildPizza class was a simple class that just created a new instance of the pizza class and set the objects base and toppings inside a main method. I then called the printReport method in the Pizza class on the new pizza object to print the pizzas report to the console.
* **Create User Interface:** In order to complete this task I created a new class PizzaChoice and created two methods inputBase and inputToppings, these methods take in a pizza object as a parameter and then set the base and toppings of the pizza based on the input of the user. Both methods print out a user interface that includes prompts on what input is required and a list of valid inputs based on the names of the enum constants that have been initialized. For the inputBase method I have implemented it so only one base can be entered, once a valid base has been inputted the while loop breaks and the method finishes and the user is told what they have selected. The inputToppings method will allow for multiple toppings to be entered, when a valid topping is entered it adds the topping to the list of toppings on that pizza and asks the user for another input, once the user has finished selecting they can enter “end” to break the while loop and finish the method. Invalid input on both these methods will produce an exception which I have caught, in my catch statement I ask the user for a valid input, and print a list of valid inputs again. I have then printed a formatted report that is easy to read by calling the printReport method on the now created pizza.
* **Named Pizza:** For this task I created both a PizzaMenu class and a PredefinedPizzas enum, I have created the enum as I again have to define a collection of constants, the enum has its own constructor that takes a name, pizzaBase and an array of pizzaToppings as parameters, I have also declared a Pizza object as a private member variable and for each object in the enum I create its corresponding pizza objects using properties to set the base and toppings of the pizza object. This made implementing the PizzaMenu class much simpler as now I could create a selectPizza method that works in the same manner as the inputBase method in PizzaChoice except this time I would be checking the input against the names of the objects defined in the PredefinedPizzas enum.
* **Vegetarian Options:** To complete this task I added a Boolean member variable named “vegetarian” to the PizzaBase and PizzaTopping enums and made it a parameter for the constructor of each. By doing this I was able to create a method that checked to see if the property was false for any of the ingredients of the pizza, if it was the method returned false, if no ingredient returned false for this property it returned true,

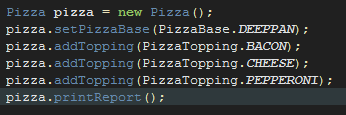
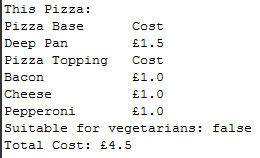
# Inheritance

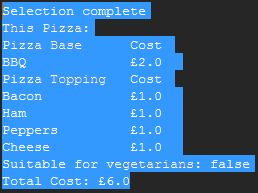
I originally intended on making use of inheritance by making the PizzaBase and PizzaTopping classes inherit from a PizzaIngredient superclass, and the namedPizza class inherit from Pizza, this would have saved me a lot of time and I would avoid the code duplication of the getters and setters. But on revising what the project required I opted to use enums instead, enums allowed me to easily define a collection of constants and tackle the tasks much more efficiently and meant inheritance was not required, or possible since an enum can’t extend a class.

# Program Output

**PizzaBuild:**

**Input Output**



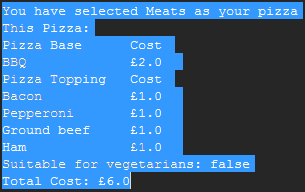
**PizzaChoice:**

**Input Output**

**Base: “bbq”**

**Toppings: “bacon”, “ham”, “peppers”, “cheese”**

**PizzaMenu**

**Input Output**

**Pizza:”meats”**

Testing

In order to make sure the project works to the specifications I have been given I have carried out some tests on the code I have written.

Test 1

The calculateCost method will be the subject of this test, I shall test to see if the method is correctly calculating the cost of the pizza object on which it will be called. During this test I will be exercising the Pizza, PizzaBase and PizzaTopping classes. This is due to the fact that the calculateCost method makes calls to the Pizza getter methods for both its private member variables and also makes calls to the getCost methods of both the PizzaBase and PizzaTopping methods. Successful results from this test will indicate not only that the calculateCost method applies the correct arithmetic but also that the private member variables, constructors and methods in these three classes have been correctly implemented.

I shall apply this test a few times with different inputs.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Comments |
|  | 3.7 | 3.7 | Cost of BBQ base = 2.0  Cost of Cheese toping = 0.8  Cost of Ham topping = 0.8  2.0 + 0.8 + 0.9 = 3.7  Correctly calculated pizza cost |
|  | 4.8 | 4.8 | Cost of Stuffed Crust base = 2.5  Cost of Bacon toping = 1.2  Cost of Ground Beef topping = 1.1  2.5 + 1.2 + 1.1 = 4.8  Correctly calculated pizza cost |
|  | 2.5 | 2.5 | Cost of Stuffed Crust base = 2.5  No toppings = 0.0  2.5 + 0.0 = 2.5  Correctly calculated pizza cost |
|  | Null Pointer Exception | Null Pointer Exception | Method is calling a getter that is returning a PizzaBase that has not been initialized yet causing a Null pointer exception |

The method correctly calculates the total cost of a pizza given a base and list of toppings, given no toppings it will work out the cost of a plain pizza (just the base and no toppings). The method returns a null pointer exception when the pizza object has not been assigned a base, although it may be reasonable to leave it this way, I have decided to go back and implement a try catch block into the method that will print out an error message to the console telling the user the pizza they created has no base, because you can’t have a pizza that consists of just toppings.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Comments |
|  | Error message  0 | Error: This pizza has no base!  0.0 | Method now catches the exception and tells the user what has gone wrong, returns 0 as a default value |

Test 2

I shall now test the isVegetarian method in the Pizza class, this method checks each ingredient of the pizza to see if it is suitable for vegetarians and if all of these return true then the pizza itself will be suitable for vegetarians. Again this will exercise the Pizza, PizzaBase and PizzaTopping classes and successful results from this test will indicate not only that the method applies the correct logic but also that the private member variables, constructors and methods in these three classes have been correctly implemented.

I shall apply this test a few times with different inputs.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Comments |
|  | true | true | All ingredients have true as there vegetarian value so result has been correctly calculated |
|  | false | false | Bacon has false for its vegetarian value so result has been correctly calculated |
|  | True | True | Pizza base has true for its vegetarian property so result has been correctly calculated |
|  | False | false | Pizza base has false for its vegetarian property so result has been correctly calculated |
|  | Null Pointer Exception | Null Pointer Exception | Method is calling a getter that is returning a PizzaBase that has not been initialized yet causing a Null pointer exception |

The method correctly determines whether or not a pizza is suitable for vegetarians given a base and a list of toppings, it also correctly determines if it is suitable for plain pizza with just the base. The method does however have the same problem the calculateCost method had, and I have decided that I will implement a similar fix into this method.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Comments |
|  | Error message  0 | Error: This pizza has no base!  0.0 | Method now catches the exception and tells the user what has gone wrong, returns false as a default value |

Test 3

Next I will be testing the inputBase method in the PizzaChoice class, the purpose of this test is to check that only valid bases can be selected, that valid input will select the respective base, that the base is correctly set to the pizza Object and that invalid input will be caught and return an error message. A successful result should show that the method will only accept valid input from the printed list and that it will continue to ask the user for a valid input until it receives one, at which point the method should terminate (can’t have more than one base). This test will exercise the PizzaChoice class and also the PizzaBase and Pizza classes as the method will make use of getters and setters from these classes as well as the use of the valueOf and Values() methods from the PizzaBase enum.

I shall apply this test a few times with different inputs.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Comments |
| “DeepPan” | Message telling me I have selected Deep Pan as the pizza Base | You have selected Deep Pan as your pizza base | Due to the toUpperCase method I have implemented into the method the user does not have to stick to capital letters format of the enum constants |
| “deepPan” | Message telling me I have selected Deep Pan as the pizza Base | You have selected Deep Pan as your pizza base | As expected |
| “deep pan” | Message telling me this base does not exist and that I need to input a valid base | Selected base does not exist, please select another base[DEEPPAN, STUFFEDCRUST, THINANDCRISPY, BBQ] | User does however have to stick to the same format as regards to positioning of the letters |
| “Invalid” | Message telling me this base does not exist and that I need to input a valid base | Selected base does not exist, please select another base[DEEPPAN, STUFFEDCRUST, THINANDCRISPY, BBQ] | This base is not a valid choice so I ask the user for another input |

The method correctly returns the base for a certain valid input and asks the user for another input when the input is invalid. The method will only terminate once a valid input has been received and assigned to the PizzaBase parameter of the pizza object in question. While I have implemented a way to avoid any errors cause by lowercase letters the user still has to exactly enter the name of the base as it has been written in the enum, this has cause a bit of a confliction as java coding conventions dictate that constants must be in all capital letters and any spaces between the words is replaced with a underscore \_ , but in order to make it easier on the user I have decided that I will break this convention as it would be easier on the user not to have to enter underscore as just ignore spaces altogether, since this is a prototype system I believe this is an appropriate thing to do for a proof of concept and is something that could be revised at a later point of necessary, because of this I have made no changes to my code.

Test 4

Next to be tested will be the inputToppings method in the PizzaChoice class, this test will run in a similar manner to the test of the inputBase method, except this time I will also be checking the case for termination of the method (when the user enters “end”), and if the method will continue to loop and allow for multiple toppings to be added to the pizza.

I shall apply this test a few times with different inputs.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Comments |
| “Ham”, “Bacon”, ”end” | Works correctly and tells me what I have selected before terminating | After ham was inputted: You have selected Ham  Toppings: [PEPPERONI, HAM, GROUNDBEEF, BACON, CHEESE, PEPPERS, MUSHROOMS]  After bacon was inputted: You have selected Bacon  Toppings: [PEPPERONI, HAM, GROUNDBEEF, BACON, CHEESE, PEPPERS, MUSHROOMS]  After end was inputted: Selection complete | Method has worked as expected and added the toppings to the pizza in question correctly before terminating |
| “end” | Tells me my selection is complete despite not having entered a topping | After end was inputted: Selection complete | I think it is reasonable that someone may want a plain pizza. |
| “invalid” | Message telling me my selection was invalid and asks me to input a new valid toppings | Selected topping does not exist, please select another topping or enter "done"  Toppings: [PEPPERONI, HAM, GROUNDBEEF, BACON, CHEESE, PEPPERS, MUSHROOMS] | As expected, method is performing checks correctly |
| “Ground beef” | Message telling me my selection was invalid and asks me to input a new valid toppings | Selected topping does not exist, please select another topping or enter "done"  Toppings: [PEPPERONI, HAM, GROUNDBEEF, BACON, CHEESE, PEPPERS, MUSHROOMS] | User has to stick to the given format as regards to positioning of the letters |

The method correctly adds the topping for a certain valid input and asks the user for another input when the input is invalid. The method will only terminate once an input “end” has been received and the toppings added to the PizzaToppings List parameter of the pizza object in question. Similarly to the inputBase method I have decided to ignore java coding conventions of placing an underscore between words in order to make it easier for the user to input a topping. Again since this is a prototype system I believe this is an appropriate thing to do for a proof of concept and is something that could be revised at a later point of necessary, because of this I have made no changes to my code.

Test 5

The selectPizza method will be the focus of this test, I will be testing it to see if it selects the correct Pizza from the PredifinedPizza enum. The test will also exercise the PredefinedPizza enum and its private member variables. A successful test will show that the PrediefinedPizza constructor is correctly creating its corresponding pizza object in its constructor and the getters and setters for the enum are working correctly.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Comments |
| “Meats” | Tells me I have selected the meats pizza | You have selected Meats as your pizza | As expected |
| “vege” | Tells me I have selected the vege pizza | You have selected Vegetarian as your pizza | As expected |
| “invalid” | Tells me this pizza does not exist and asks for a different input | Selected pizza does not exist, please select another pizza  Pizzas: [CHEESE, MEATS, VEGE] | Method recognises invalid inputs and prints an error message |

This method correctly makes a predefined pizza based on the enums defined in PredefinedPizzas and selects the correct pizza based on the user input. This test has been successful and I have made no changes to my code.

Test 6

I will now go about testing the printReport method in the Pizza class, I will test its flexibility and correctness by calling it in the BuildPizza, PizzaChoice, and PizzaMenu (already inside the selectPizza method). The printed report produced should not only contain all of the ingredients of the pizza but also the cost of each, the pizzas suitability for vegetarians and the total cost of the pizza. The report should also be easy to read (to test this I will require the opinions of other people).

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Comments |
| PizzaBuild | Formatted list of all the ingredients and cost of each, the pizzas the suitability for vegetarians and the total cost of the pizza. |  | Works as expected returning the correct ingredients, costs and properties of the pizza |
| PizzaChoice  Input:  Base: stuffedcrust  Toppings:mushrooms, ham, bacon, peppers | Formatted list of all the ingredients and cost of each, the pizzas the suitability for vegetarians and the total cost of the pizza. |  | Works as expected returning the correct ingredients, costs and properties of the pizza |
| PizzaMenu  Input:  vege | Formatted list of all the ingredients and cost of each, the pizzas the suitability for vegetarians and the total cost of the pizza. |  | Works as expected returning the correct ingredients, costs and properties of the pizza |

The method has correctly produced the report in all the classes, it correctly gets the names of the base and toppings of the pizza and their respective prices. It also correctly uses the calculateCost and isVegetarian methods in the Pizza class to print the properties of the pizza in the report. I asked a few people whether or not they thought the report was easy to read, they all agreed that could clearly read and understand the report so it meets the requirements of the specification.

Conclusion

The above tests show that my projects I fully operational and is meeting all the requirements asked of it by the specification. All calculations are being performed correctly, the input validation checks are working and the output is formatted, easy to read and understand.