CSE 174 - Lab 11

Part 1 (10 points): Calculating the cost of a pizza

1. Download Pizza.java. DO NOT MODIFY THIS FILE.
2. Open `Pizza.java` in jGRASP, compile it, and then close `Pizza.java`.
3. In a browser window, open the API documentation for the Pizza class here: http://krumpe.com/javadoc/Pizza
4. Remember, you don't need to understand every code in Pizza.java file. You only need to understand the Pizza class by studying the documentation.
5. In jGRASP, create a new class named `PizzaShop`.

In this Pizza class

- the valid pizza sizes are: small, medium, and large
- the valid crust styles are: thin, thick, and stuffed

6. Create a `main()` method and familiarize yourself with the `Pizza` class:
    - Create a small, stuffed-crust pizza
    - Add pepperoni and onions
    - The pizza should be for delivery
    - Print the pizza using `toString()`

7. There is a constructor that will construct a random pizza based on a lucky number (any int value). Try creating a pizza using that constructor, and print the pizza.

8. After the `main()` method, create a method with the following header:

```
public static double pizzaCost(Pizza p)
```

9. Implement the pizzaCost() method as follows (and you can paste these comments into your code), so that it returns the cost of the specified pizza.

```
/*
      $4 for small, $5.50 for medium, $7 for large
      Stuffed crust: $1 extra for small, $2 for medium, $3 for large
      No extra charge for thin/thick
      Toppings $0.75 each. However, there an additional $0.50 charge if
         at least one of the toppings is anchovies.
         For example, if the toppings are: anchovies, anchovies, onions,
           and pepperoni, then the toppings charge will be $3.50
           (4 toppings cost $3.00, and an extra 50 cents because at least
           one of the toppings is anchovies).
      Delivery: $2 (free for pizzas costing $10 or more)
*/
```

10. In your `main()` method, check that your pizza cost method is computing the correct cost for various pizzas. Here are a few random lucky-number pizzas you can test:

- `new Pizza(234)` should have a cost of $7.75.
- `new Pizza(8675)` should have a cost of $10.00.

   You should test with many pizzas until you are confident that your method works.

11. When you are confident that your `pizzaCost()` method works, and you have fixed any style issues, and added comments to the top of your code and to the `pizzaCost()` method, submit `PizzaShop.java` to the autograder. Notes:
    - It does not matter what you put in your `main()` method because your `main()` method will not be run. The autograder will run its own main method, calling your `pizzaCost()` method with several random pizzas, and display the result.
    - There is only one test case, and that test case contains multiple random pizzas. In order to submit, your method must correctly compute the cost for all pizzas.
12. **After your code passes the autograder test, be sure to submit it.**

Rubric

Criteria	Full Credit	No Credit
Successful Submission via CODE	A fully successful submission to CODE that passes all of the required tests will earn full credit. **8 points**	If your submission is not accepted by CODE, you will receive no credit. **0 Points**
Correct Style in CODE	If your submission has 0 style errors in CODE you will receive full credit. **1 Point**	If there are any style errors present in CODE, you will receive no credit. **0 Points**
Correct Comment	If your submission has proper comments at the top of your class and at the top of your pizzaCost() method, you will receive full credit. **1 Point**	If your submission is missing either of these comments, you will receive no credit. **0 Points**

Part 2 (10 points): Building pizza orders from a file

Begin this part of the lab after your `pizzaCost()` method is working correctly.

In this part of the lab, you will write a method that reads a customer's pizza order from a file, constructs and prints the pizza, along with its cost.

Pizza order text files will always look as follows:

- Line 1: the pizza size (should be small, medium, or large, but if it's anything else, it would be considered an INVALID ORDER. Note that one of the Pizza constructors will throw an exception if the size is invalid)
- Line 2: the crust style (should be thin, thick, or stuffed, but if it's anything else, it would be considered an INVALID ORDER. Note that one of the Pizza constructors will throw an exception if the crust type is invalid)
- Line 3: delivery or carryout (line 3 will always be one of these two values)
- The remaining lines, if any, will be a list of the toppings that the customer wants added to the pizza. Note that there might not be any toppings listed. That's fine. Some customers like plain pizzas. You should add all toppings the customer asks for, unless it is a duplicate, or something your shop does not carry...
    - At your pizza shop, you do not allow "duplicate" toppings. So, if a customer asks for onions two times, you will only add it one time.
    - If a customer asks for a topping that is not valid (like apples or shoes), you will let the customer know that it is not a valid topping. Note that one of the Pizza class' methods will help you determine whether the topping is valid or not.

Here are examples of what a pizza order will look like (some of these are not valid):

order1.txt	**order2.txt**	**order3.txt**	**order4.txt**	**order5.txt**
small	large	medium	supersized	small
thin	stuffed	thick	thin	burnt
delivery	carryout	delivery	carryout	delivery
pepperoni	onions		sausage	olives
onions	onions			
mushrooms	sausage			
anchovies	snickers			
	pepperoni			

1. Continue the `PizzaShop.java` class you started in Part 1, because you will need the `pizzaCost()` method.

2. Change your `main()` method to the following:

```
public static void main(String[] args) {
   Scanner kb = new Scanner(System.in);
   System.out.print("Enter order filename: ");
   String fileName = kb.next();
   pizzaOrder(fileName);
  }
```

3. Write the following method header after the main() method:

```
 public static void pizzaOrder(String fileName)
```

4. There are several different problems that could cause exceptions to be thrown: missing file, invalid pizza size, invalid crust type, or some kind of I/O error. Use a single `try {}` block and a single `catch {}` block to handle <u>all</u> exceptions. If any exception is thrown, simply print: **`INVALID ORDER. GOODBYE`**

5. Implement the logic of this method so that it reads the specified file (if it exists), creates the pizza (if it is valid), and adds all valid toppings, showing the result for each topping. The output should match what is shown in the sample runs **(see next page)**.

6. When your code is working correctly, submit only `PizzaShop.java` to the autograder. There are 9 test cases. The first 6 match the test cases shown on the next page, and then there are 3 "hidden" test cases. **After your code passes the autograder test, be sure to submit it.**

Rubric

<u>Criteria</u>	<u>Full credit</u>	<u>Partial credit</u>	<u>No credit</u>
Pass test cases in CODE	A fully successful submission to CODE that passes all 9 tests will earn full credit. **9 points**	A submission to CODE that passes fewer than 9 tests will receive: **1 point per passed test**	If your submission passes no tests, you will receive no credit. **0 Points**
Correct Style	If your submission has 0 style errors in CODE and if you have commented your pizzaOrder() method, you will receive full credit. **1 point**	▨	If there are any style errors present in CODE, or you are missing comments, you will receive no credit. **0 Points**

Sample runs (user input shown in red):

Sample run	**File**
Enter order filename: order1.txt ---Toppings--- pepperoni: added onions: added mushrooms: added anchovies: added ---Here is your order--- DELIVERY ORDER: small, thin toppings: [anchovies mushrooms onions pepperoni] Your total is $9.50. ---Have a nice day---	small thin delivery pepperoni onions mushrooms anchovies
Enter order filename: order2.txt ---Toppings--- onions: added onions: DUPLICATE. NOT ADDED. sausage: added snickers: TRY PAPA JOHNS. pepperoni: added ---Here is your order--- CARRY-OUT ORDER: large, stuffed toppings: [onions pepperoni sausage] Your total is $12.25. ---Have a nice day---	large stuffed carryout onions onions sausage snickers pepperoni
Enter order filename: order3.txt ---Toppings--- ---Here is your order--- DELIVERY ORDER: medium, thick toppings: [] Your total is $7.50. ---Have a nice day---	medium thick delivery
Enter order filename: order4.txt INVALID ORDER. GOODBYE	supersized thin carryout sausage
Enter order filename: order5.txt INVALID ORDER. GOODBYE	small burnt delivery olives
Enter order filename: thisisnotavalidfilename.txt INVALID ORDER. GOODBYE	