Programming Assignment 7

Getting Started

This assignment is designed as a practice on programming with classes. Review class handouts and posted practice problems.

I will make make my solution to the previous homework - Pizza.java and TestPizza.java available via email. You can use my solution, or yours, provided, that your Pizza.java is defined in accordance with the specifications laid out by the homework project.

Programming Project

PizzaOrder: Taking pizza orders **worth 28 points**

In this assignment, you will continue your development of the Pizza class, and will also create a class type representing a customer order for a pizza take out. Your classes and methods should be named as specified.

Step 1 - update class Pizza (7 points)

As a starting point, you can use **my solution, or yours, provided, that your Pizza.java is defined strictly in accordance with the specifications laid out by the homework project CreateAPizza.**

Update the **Pizza** class to include the following **public static** constants, representing pizza status:

```
public final static int NOT_STARTED = 0;
public final static int IN_PROGRESS = 1;
public final static int READY = 2;
```

Furthermore, modify instance method *setStatus()* to use the static constants defined above. **(1 point)**

Add the following methods to class **Pizza**. Be sure to follow the requirements on method names, input parameters, return value and functionality precisely. **Use the defined static constants where appropriate.**

1. **(1 point) A no-argument constructor.** This constructor should initialize the instance variables to represent a medium size pizza with no toppings, with status equal to 0, i.e. `NOT_STARTED`.
2. **(1 point) A 2-argument constructor**, which will be passed two parameters: a character representing the size and an array of Strings representing the toppings. The constructor should call the appropriate *set* methods of the String class, to assign the values passed as parameters to the instance variables *size* and *toppings*, and to assign 0 , i.e. *NOT_STARTED*, to the instance variable *status*.
3. **(1 point) instance method statusPhrase()**, which has no parameters and returns a String, describing the calling object's status as a phrase, i.e. one of "Not started", "In progress" or "Ready". This method can be used for displaying the status of the pizza.

4. **(2 points) instance method toString()**, which will be passed no parameters and should return a String describing the pizza, showing the pizza information, as shown in the examples below (you must make use of instance method *statusPhrase()*). The first example describes what *toString()* should return, when called by a large 2-topping (tomatoes and chicken) pizza with status 0.

```
************************************************
Pizza size L. Toppings:
1.  tomatoes
2.  chicken
Not started.
************************************************
```

   The next example shows what toString() should produce for no topping pizza:

```
************************************************
Pizza size M. No toppings.
Ready.
************************************************
```

5. **(1 point) method main()** that will test the constructors and method *toString()* by creating Pizza objects from hardcoded data of your choice and printing them out.

When finished, run the main method of the Pizza class to verify that the methods you developed are working properly, before moving on to the next step.

Step 2 - create class PizzaOrder (11 points)

Further, define class **PizzaOrder**, representing a customer order of pizza pies. Include the following private instance variables **(1 point)**

- String customer
- Pizza [] orderedPizzas

There should be no other instance variables in this class.

Add the following methods to class **PizzaOrder**. Be sure to follow the requirements on method names, input parameters, return value and functionality precisely.

1. **(1 point) A 2-argument constructor.** This constructor must be passed two parameters: a String representing the customer name and an array of Pizza objects. The constructor should initialize the instance variables to the values passed through parameters.
2. **(1 points) Accessor methods** for each instance variable.
3. **(2 points) instance method totalPrice()**. This method should take no parameters and should calculate and return the total price of the order (as a value of type double), as the sum of prices of all pizzas stored in the calling object's array *orderedPizzas*.
4. **(3 points) instance method toString()**, which will return a String, containing the description of a pizza order, showing the customer and the list of all pizzas, which includes the size, number of toppings, status (as a phrase) and the price of pizza. Include the order total as the last line, as shown below. You must make use of the appropriate instance methods of the Pizza and PizzaOrder classes to produce the result.

```
Order for customer Peter Stone:
Large pizza, 2 toppings, $14.00 -- In progress
Small pizza, 4 toppings, $12.00 -- Ready
Medium pizza, 0 toppings, $9.00 -- Not started
Order total:  $35.00.
```

5. **(3 points) instance method isReady()**. This method should return `true` if and only if the status of all pizzas stored in the *orderedPizzas* instance variable equals `Pizza.READY`. Otherwise, return `false`.

As you are working on this step, it is best to simultaneously work on step 3.

Step 3 - create a main method in class TestOrder - (8 points)

This step should parallel the development of the **TestPizzaOrder** class. Create a new class, called **TestPizzaOrder** in the same project folder with the other two classes. **TestPizzaOrder** should have a single main method which will be used for testing the functionality of the **PizzaOrder** class. Instead of getting the data from the user, use hardcoded values for convenience.

The main method should:

1. Create a 3-element array of type Pizza. Initialize it with 3 Pizza objects: a large pizza with 2 toppings, a small one with 4 toppings and a medium one with no toppings. Note that before you create each pizza, you will need to create a separate array of toppings for it, except for the one with no toppings. Use any toppings you like.
2. Create a new object of class PizzaOrder for customer Peter Stone and the array of pizzas created earlier.
3. Print the order.
4. Update the order by setting the status of the first pizza to `Pizza.IN_PROGRESS`, and the third one to `Pizza.READY`. **You must use the accessor method of the PizzaOrder class to access the pizza array stored with the order.**
5. Print the order.
6. Call method *isReady()* on the created pizza order, and print out the result.
7. Update the order by setting the status of all pizzas to `Pizza.READY`.**You must use the accessor method of the PizzaOrder class to access the pizza array stored with the order.**
8. Print the order.
9. Call method *isReady()* on the created pizza order, and print out the result.

The following illustrates how the TestOrder class should run:

```
Order for customer Peter Stone:
Large pizza, 2 toppings, $14.00 -- Not started.
Small pizza, 4 toppings, $12.00 -- Not started.
Medium pizza, 0 toppings, $9.00 -- Not started.
Order total: $35.00

Order for customer Peter Stone:
Large pizza, 2 toppings, $14.00 -- In progress.
Small pizza, 4 toppings, $12.00 -- Not started.
Medium pizza, 0 toppings, $9.00 -- Ready.
Order total: $35.00
```

```
order ready is false

Order for customer Peter Stone:
Large pizza, 2 toppings, $14.00 -- Ready.
Small pizza, 4 toppings, $12.00 -- Ready.
Medium pizza, 0 toppings, $9.00 -- Ready.
Order total: $35.00

order ready is true
```

Submit *Pizza.java*, *PizzaOrder.java* and *TestPizzaOrder.java*. 2 points will be awarded for good programming style.