

## Programming Assignment 6

### Getting Started

This assignment is designed as a practice on the basics of class definitions. It requires knowledge of the material in Sections 4.1 and 4.2. Review class handouts and posted practice problems.

### Programming Project

**CreateAPizza:** Define a class, representing a pizza

**worth 17 points**

A pizza take-out shop has contracted you to create an application for managing the orders it receives from customers. In this assignment, you will work on the first step in this application: creating a class type representing a pizza.

Define class **Pizza** as follows. Each object of this class should include the following private instance variables **(1 point)** :

1. *size* - a character (type *char*), representing the size of the pizza: M - for medium, S - for small, and L - for large;
2. *toppings* - an array of strings (type *String []*), specifying the set of pizza toppings (excluding the cheese),
3. *status* - an integer, representing the status of the ordered pizza, which can be either 0 - not started, 1 - in progress, or 2 - ready;

All instance variables should be declared **private**, and no other instance variables should be included in this class.

Define the following methods:

1. **(2 points) instance method setSize()**, which should be passed a parameter of type *char*, representing the size of the pizza. The method should check that the parameter is equal to 'S', 'M' or 'L', and if it is, assign the calling object's *size* instance variable to that parameter. Otherwise, the method should report that the size value is not recognized and set the instance variable *size* to 'M'. The method should return true, if the value of the size parameter was appropriate, and false otherwise.
2. **(2 points) instance method setStatus()**, which should be passed an integer parameter, representing the status value. The method should check that the parameter is between 0 and 2, and if it is, assign the calling object's *status* field to that parameter. Otherwise, the method should report that the size value is not recognized and set the instance variable *status* to 0. The method should return true, if the parameter defined a valid status, and false otherwise,
3. **(1 point) instance method setToppings()**. This method should be passed a parameter of type *String []* (i.e. String array), representing the requested pizza toppings. The method should assign the calling object's instance variable *toppings* to the parameter that was passed. This method should return no value, i.e. be declared as *void*.
4. **(1 points) Accessor methods** for each instance variable.

5. **(1 point) instance method numToppings().** This method should check if the instance variable *toppings* is not null, and in that case return the length of the array stored in the *toppings* instance variable. If the *toppings* array is null, the method should return 0.
6. **(2 points) instance method calcPrice.** This method should calculate and return the price of the pizza (as a value of type double) according to the following rules: the price equals the base price plus additional price per each topping. The base price for small, medium and large pizza, respectively, is \$8.00, \$9.00 and \$10.00. Each topping costs \$1.00 for small pizza, \$1.50 for medium, and \$2.00 for large.

Further, define class **TestPizza** in the same project folder with class **Pizza**. **TestPizza** should have a single main method which will be used for testing the functionality of the **Pizza** class. The main method should **(5 points)**

1. Create a new object of class **Pizza**.
2. Prompt the user to enter and read a character indicating pizza size.
3. Ask the user about the number of toppings and create an array to store the specified number of toppings.
4. Run a loop to read the topping names and store them in the array.
5. Call the appropriate methods of the **Pizza** class to set instance variables of the created pizza object to the values entered by the user.
6. Print a message about the pizza, including the price information, using the appropriate instance methods of the **Pizza** class. The price should be formatted to display 2 digits after the period.

The following two sample interactions illustrate how the **TestPizza** class should run (user input is shown in **boldface**):

```
Please enter the size of pizza (S, M or L): L
How many toppings would you like to add to the pizza? 3
Please enter topping number 1 mushrooms
Please enter topping number 2 peppers
Please enter topping number 3 chicken
*****
Created a pizza of size L and 3 toppings:
1. mushrooms
2. peppers
3. chicken
This pizza costs $16.00
Pizza readiness status is 0
*****
```

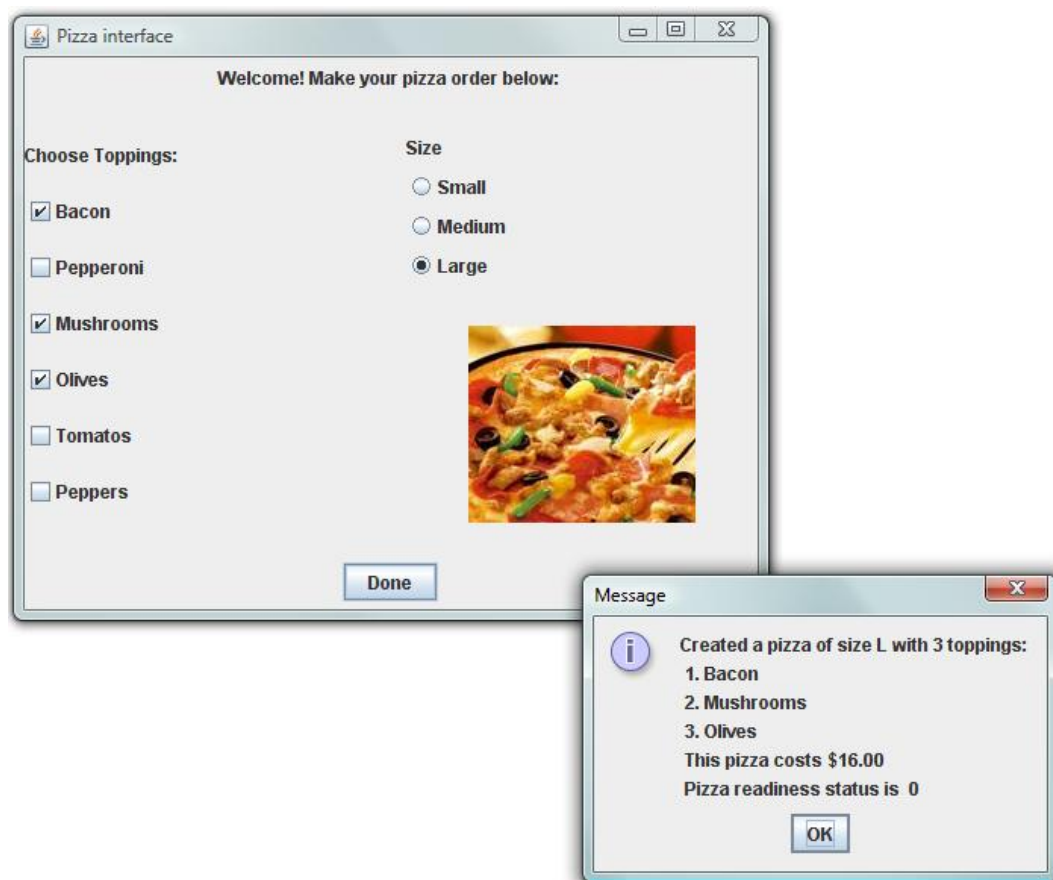
Another interaction shows how the incorrect pizza size and 0 additional toppings are handled:

```
Please enter the size of pizza (S, M or L): W
How many toppings would you like to add to the pizza? 0
Unrecognized size: W. Setting size to M
*****
Created a pizza of size M with 0 toppings:
This pizza costs $9.00
Pizza readiness status is 0
*****
```

**Add graphical user interface to work with your Pizza class.** You don't need to write any additional code for this part. I am supplying a graphical interface that uses `Pizza` class to demonstrate how your class can be used with another class file. The supplied interface includes two files: `PizzaGUI.java` and `pic3.jpg` (an image file used by `PizzaGUI.class`). These two files should be placed in the same project folder, where you keep `Pizza` and `TestPizza` classes. If your project has `src` and `bin` folders, place these files in `src`.

When you are finished defining and testing your `Pizza` class, you can run the `PizzaGUI`, and it should work by allowing you to select pizza parameters and displaying the pizza information, upon clicking the `Done` button. The following image illustrates the `PizzaGUI` application running.

If your `Pizza` class is not defined according to the provided above specification, The `PizzaGUI.class` will generate an exception when you click on the `Done` button. This may happen if you are missing a method, or have defined it with a different set of parameters or misspelled its name. You will need to fix your code before you submit it to get full credit.



**Submit** `Pizza.java` and `TestPizza.java`. Note that using good programming style counts for **2 points** in this assignment. In addition to the rules of good style defined before, each method should have an introductory comment, describing its purpose, parameters and return value.