

Description

In this project you will design an interactive pizza ordering system for Bucknell. Your program will begin by drawing a plain pizza. It will then ask the user about a variety of toppings and if the user wants a given topping, the topping will appear on the pizza.

Bucknell wants to reward enthusiastic pizza eaters who order many toppings. Additionally, Bucknell Dining services has accidentally ordered 1,000 boxes of anchovies instead of apples. This is their only hope of using up the anchovies. Therefore, if the customer has ordered 3 or more toppings or has ordered anchovies, they will get a free soda. If they don't win a soda, your program will ask if they'd like to buy one.

The available toppings are pepperoni, black olives, anchovies, and bacon. The total price will be displayed in the upper left hand corner. The price will be updated each time a topping or soda is ordered. The pizza picture will also be updated each time a topping is ordered. When the customer gets a soda (for free or by buying it), a picture of the soda will also appear.

(Note: It is not possible to order a soda without the accompanying pizza.)

Pizza Class Details

For this project you will write a `Pizza` class. This class should have the following constructor and methods.

- Boolean instance fields for each topping possibility and for the soda.
- You should need few, if any, additional instance fields. Remember that instance fields capture the essence of an object.
- A single constructor that does not need any parameters. It should initialize the instance fields to appropriate values.
- Mutator methods which can change the boolean instance fields to indicate the presence of a topping or a soda.
- A method which decides if a free soda is warranted. This method should **not** change the instance field for the soda. We'll let the `main` method in the viewer class do all the topping and soda ordering.
- A method which computes and returns the pizza price. This method should make sure the user is not charged for the soda if it was free. This method should **not** change the instance field for the soda.

Plain pizza costs \$10.00. Toppings prices vary by the time of day.

If there are two or fewer toppings, they cost \$1.50 each. If there are three toppings, they cost \$1.25 each. If there are more than three toppings, they cost \$1.00 each.

If the pizza is being ordered before noon, the toppings cost \$1.10, \$.70 ,or \$0.60 using the same setup seen above.

If the pizza is being ordered after 8pm, the toppings cost \$1.35, \$1.05, and \$0.85 using the above setup.

Soda costs \$1.99 regardless of what time it is.

GregorianCalendar provides a simple way to get the time of day. Use its get method along with the following fields.

- Calendar.*HOURLY*
- Calendar.*AM_PM*
- Calendar.*AM*
- Calendar.*PM*

(Note: If the pizza is ordered after midnight, it gets the early bird pricing. That will simplify any need for extreme late night pizza pricing.)

- A draw method similar to those in the Car and Triangle projects.

The pizza will be centered in a window and have a diameter of 400.

The outer circle, your crust, must be a light brown color. The next inner circle, your sauce, must be a red color. The innermost circle, your cheese, must be a yellowish-white color. If you don't like the colors that come with Java, then design your own. If you get RGB values from the Internet or a book source then cite that source in a comment with your code.

For each topping that has been ordered, you will declare, instantiate, and draw at least 5 objects of that type. Choose a placement strategy that sprinkles them around your pizza. It is OK if toppings overlap as long as they can obviously be seen on your pizza.

If a soda is ordered (or free), it should appear on the screen but not on top of the pizza.

The current price of the whole dinner should be calculated and then displayed in the upper left hand corner of the screen.

PizzaComponent Class Details

Create a pizza component class that you will use to display a pizza. This component is similar to the one from the Car project. Your component class will need:

- an instance field for a `Pizza` object.
- a constructor which accepts a `Pizza` object as a parameter.
Why? You will be declaring and instantiating your pizza in the `PizzaViewer` class. That will enable you to make changes to your pizza and have the screen update to show the changes.
- a `paintComponent` method similar to those in Triangle project. This method should do little more than call the `draw` method in the `Pizza` class.

PizzaViewer Class Details

The viewer for this class will be very similar to those we've used before. It will setup the frame as usual (frame size 600 by 600) and then declare and instantiate a `Pizza` object. Then it will call the `add` method of your frame and pass the pizza as a parameter to your `PizzaComponent` constructor. After you have made the frame visible, you can proceed with the pizza ordering.

Import all the files found in `~csci203/2009-fall/student/projects/proj6`. Run `OliveViewer` to see an example of graphics updating. The method call that causes the screen to redraw is `JFrame` method `repaint` in the viewer. This works because the `Olive` is instantiated in the viewer and passed as a parameter to the component constructor. You will use this idea to instantiate a `Pizza` object in the `PizzaViewer` and call `repaint` after each update to the pizza.

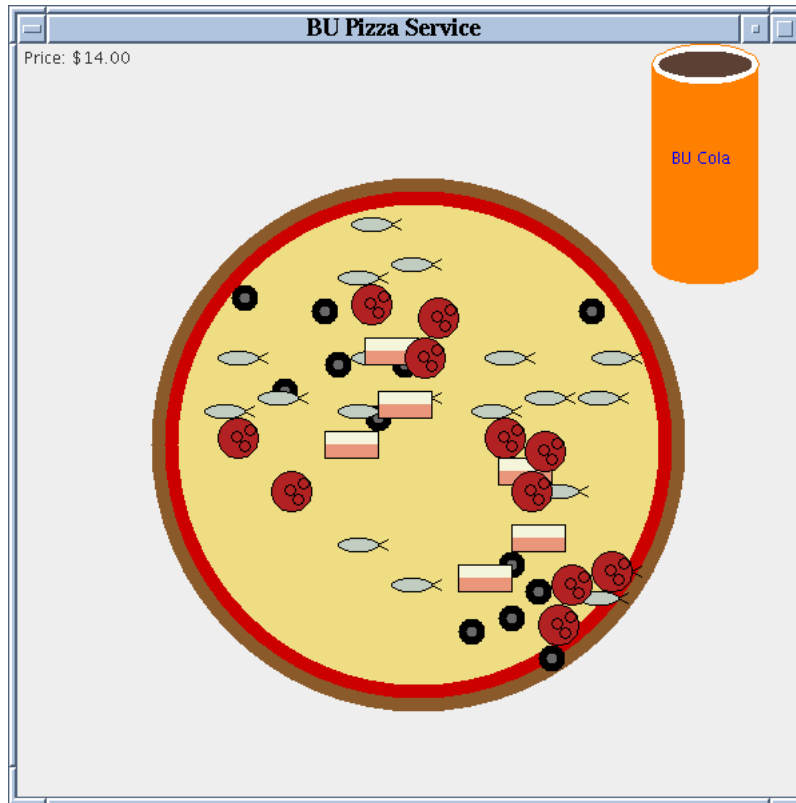
Your `PizzaViewer` will handle the ordering of toppings and soda. One by one, inquire if the customer would like to add a given topping. If the customer agrees, then you should do two things

1. Call the appropriate `Pizza` setter method to indicate the presence of that topping.
2. Call `repaint` on the frame to update the screen.

After all the toppings have been chosen, find out if the customer has won a free soda. If so, order the soda. If not, enquire whether the customer would like to order the soda. If a soda has been ordered (free or otherwise), make sure you call the `repaint` method of the frame so that the soda is displayed on the screen.

Here is a sample run of the ordering system in the console:

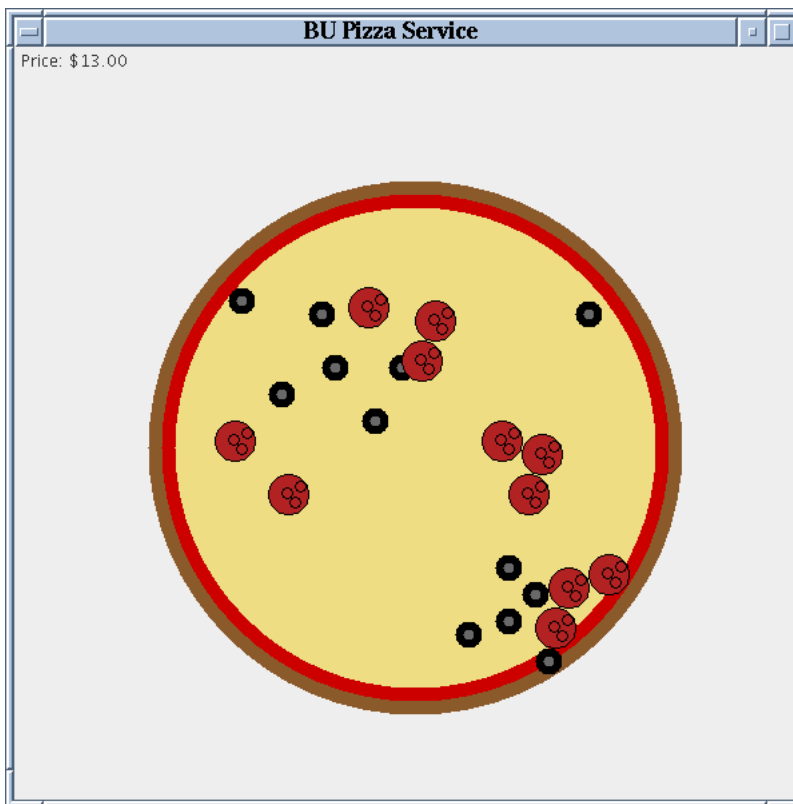
Do you want olives? (y/n) y
Do you want anchovies? (y/n) y
Do you want pepperoni? (y/n) y
Do you want bacon bits? (y/n) y
You've won a free soda.
Thank you for ordering.



Here is another example where the customer does not win a free soda:

Do you want olives? (y/n) y

Do you want anchovies? (y/n) n
Do you want pepperoni? (y/n) y
Do you want bacon bits? (y/n) n
Do you want a soda? (y/n) n
Thank you for ordering.



Classes for Toppings and Soda

You should already have imported the toppings and soda from `~csci203/2009-fall/student/projects/proj6`.

The constructor for these classes takes the x and y coordinates of the top left corner of the bounding box. Please note that draw method for these classes will draw exactly one of the item. It is up to the Pizza class to declare and draw multiple objects.

Do *not* edit these classes. To make it easier to place the toppings and soda on the frame, we have given you a Ruler class. You can add it to your pizza but remember to remove it again before you turn in the project.

Check List

- Be sure to have a class comment for each of your classes.
- Each method you write should have a Javadoc comment.
- Your name and the date should appear at the top of each file.
- Make sure that the names you use in your program are descriptive and that they follow Sun's naming conventions.
- Make sure that the correct pizza toppings appear on the pizza only when they are ordered and if every topping is ordered, you can clearly see and identify them on the pizza.
- Make sure that the price updates correctly as each topping or soda is ordered.
- Make sure the free soda appears only when it has been won and the customer is not charged for free soda.
- Avoid magic numbers!

What to Submit

Drag your `proj6-xyz01` folder in the the drop box with your instructor's name, *not* the lab drop box.