CIS 162 Lab 9
Graphical User Interfaces (GUI)

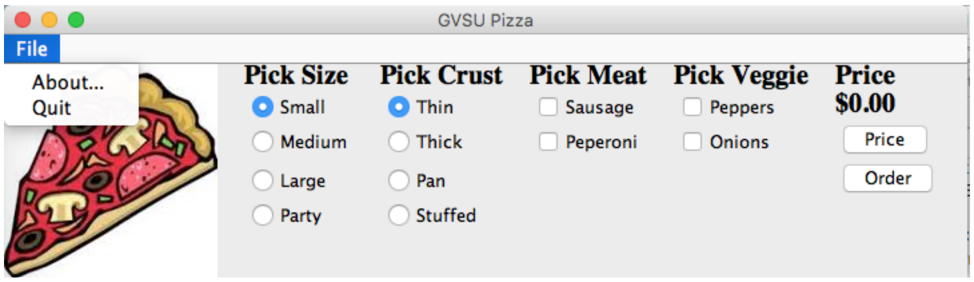
Preparation
Do the following before arriving at lab:
- Read section 9.1 – 9.4

Objectives
After completing this lab, you should be able to:
- *Position* buttons and text fields in a `JFrame`
- *write* an action listener
- *describe* the advantage of separating the model and the GUI (view)
- *describe* how the user controls program actions by clicking on a button

Model-View Design Pattern
A common design strategy is to divide applications into at least two parts: the user interface (View) and a back end that takes care of the data (Model). For this lab, we provide a completed Model class that stores a pizza order and is responsible for calculating the cost. You will expand an initial GUI that allows a customer to specify an order.

The Model-View pattern enforces clear responsibilities between the model and view. For example, the GUI (view) should not be responsible for maintaining any data or knowing how the data are processed. The sole responsibility of the GUI is to support user interaction with the Model.

Sample Display
Your GUI will eventually look similar to the following:



Step 1: Create a New BlueJ Project

Step 2: Download the provided PizzaOrder class to your project

Rather than writing your own class, we are providing a completed class for you.

Right click on the `PizzaOrder.class` file to download and Save As… on your computer in the folder for the newly created BlueJ project. Pay attention where it is saved and be sure to provide the correct name! You might not see the class icon within BlueJ but try closing and the opening again.

You do not have access to the code but need to understand how to use each of the following methods:

- `PizzaOrder()` – constructor
- `void clearOrder()` – reset order to small, thin crust with no meats or veggies.
- `void setSize(int size)` – set the pizza size. Valid parameters include: PizzaOrder.SMALL, MEDIUM, LARGE and PARTY.
- `void setCrust(int crust)` – set the crust style. Valid parameters include: PizzaOrder.THIN, THICK, PAN and STUFFED.
- `void addMeat()` – add one meat topping to the order.
- `void addVeggie()` – add one veggie topping to the order.
- `double getPrice()` – calculate and return the price.
- `String getOrder()` –return a text summary of the order.

Sample Usage

```
PizzaOrder theOrder = new PizzaOrder();
theOrder.setSize(PizzaOrder.LARGE);
theOrder.setCrust(PizzaOrder.PAN);
theOrder.addMeat();
double price = theOrder.getPrice();
String summary = theOrder.getOrder();
```

Step 3: Modify the provided PizzaGUI class

Rather than writing your own GUI class, we are providing code to get you started. Create a new class in BlueJ called `PizzaGUI` and delete all of the provided code. Copy and paste the provided code from (PizzaGUI.java) into the newly created class. It should compile with no changes.

We modeled our design on examples from the book in section 9.3. Read the code and comments carefully to understand how it works. You start the program by invoking `main()`.

Instantiate the model object

- Instantiate the `PizzaOrder` object in the constructor. Look for FIX ME comments in the code.

```
        theOrder = new PizzaOrder();
```

Does the GUI compile? Can customer price a small thin pizza?

Add sizes and crusts

- Add size options for Large and Party. You must define additional `JRadioButtons` and use the current code as a guide.
- Add crust options for Pan and Stuffed. You must define additional `JRadioButtons` and use the current code as a guide.
- Add if statements within `updatePizzaOrder()` to let the model object know what was selected.

Does the GUI compile and run? Can customer price different size pizzas?

Add meat and veggie options

- Add at least two more meat options of your choice. You must define additional `JCheckboxes` and use the current code as a guide.
- Add at least four more veggie options of your choice. You must define additional `JCheckboxes` and use the current code as a guide.
- Add if statements within `updatePizzaOrder()` to let the model object know what was selected.

Does the GUI compile? Can customer price different pizzas with different options?

Add File / Quit option

- Find the `quit()` method that is invoked when the customer selects Quit from the file menu. Add the following to quit the application on demand.

```
            System.exit(0);
```

Does the GUI compile? Can customer quit the application?

Add Another Menu Item: About

A file menu with one item is provided for you and is created with `setupMenu()`.

1. Add an additional menu item labeled “About…”. Look for FIX ME within `setupMenu()`.
2. Add a method named `about` that displays your names within a pop-up JOptionPane message.

```
        JOptionPane.showMessageDialog(this,"Our names go here");X
```

3. Call `addActionListener` on your new `JMenuItem`.

Does the GUI compile? Can customer display the About information?

Add Another Button: Order

- Add an additional JButton labeled "Order". Look for FIX ME.
- Add method to display the current order. You will have to ask the PizzaOrder object for the order summary.

```
String summary = theOrder.getOrder();
JOptionPane.showMessageDialog(this, summary);
```

- Call `addActionListener` on the new button.

Does the GUI compile? Can customer display the order?

Add an Image

Code is provided to display an appropriate pizza image to the left of the frame.

- Find an appropriate "jpg" or "gif" image on the Internet and download to your BlueJ project folder.
- Update the code to reflect the filename of your image.

Does the GUI compile? Does the image display?

Grading Criteria

Demo working app for your instructor. This lab is worth 1 point for attendance. No code is uploaded to zyBook.