# Section A

**TK1143 - Tutorial 5 Graphical User Interface (GUI)**

*Topic: JFrame and ContentPane.*

1. Complete the following class (*L1-L3, L5-L6* and *L10, L12-L14*) to create a Java frame whose title is “My GUI Frame”, width is 350, height is 250, and content pane’s background color is magenta (as shown below).



Java.awt.\*

|  |  |
| --- | --- |
| L1 | import //import package awt  Java.swing.\* |
| L2 | import //import package swing |
| L3 | class MyFrame extends { |
| L4 | public MyFrame () {  getContentPane(); |
| L5 | Container pane =  setBackground(Color.magenta); |
| L6 | pane. //set background |
| L7 | } |
| L8 | public static void main(String[] args) |
| L9 | {  MyFrame frame = new MyFrame |
| L10 | //create frame |
| L11 | frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setTitle(“My GUI frame”); |
| L12 | //set title  frame.setsize(350,250); |
| L13 | //set size  frame.setVisible(true); |
| L14 | //set visible |
| L15 | } |
| L16 | } |

*Topic: Layout Manager, GUI Component and JPanel*

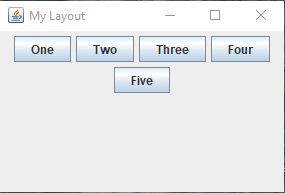
1. Name one Java Swing components that can be used for the following purposes:

|  |  |
| --- | --- |
| Purpose | Component |
| a. Displays text on the Java form but cannot receive any input. | JLabel |

|  |  |
| --- | --- |
| b. User can choose only one from many options | JRadioButton |
| c. Displays lines of text that can be chosen by user | JList |
| d. A component that hides text from user. | JPasswordField |
| e. Display information, warning or input | JOptionPane |
| f. User can select more than one items from many items to choose. | JCheckBox |
| g. User can key in only one line of text inside it | JTextField |
| h. Provide a list of items from which the user can make a single selection. | JComboBox |
| i. It has a label and generates an event when pressed. | JButton |

1. Complete the line of statement based on the following figure.

b2 b3



b1 b4

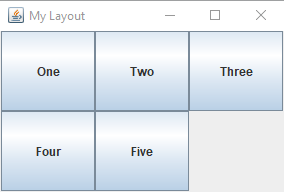
b5

Java.awt.\*;

|  |  |
| --- | --- |
| L1 | import  Java.swing.\*; |
| L2 | import |
| L3 | JFrame |
| L4 | class MyLayout extends {  private JButton b1,b2,b3,b4,b5; |
| L5 | //Declare component |
| L6 |  |
| L7 | public MyLayout() { |
| L8 | Container pane = getContentPane();  pane.setLayout(new FlowLayout(FlowLayout.CENTER)); |
| L9 | //Set Layout |
| L10 | b1=new JButton("One"); //create b1  b2=new JButton(“Two”); |
| L11 | //create b2  b3=new JButton(“Three”); |
| L12 | //create b3  b4=new JButton(“four”); |
| L13 | //create b4  b5=new JButton(“Five”); |
| L14 | //create b5 |
| L15 | pane.add(b1); //Add b1  pane.add(b2); |
| L16 | //Add b2  pane.add(b3); |
| L17 | //Add b3  pane.add(b4); |
| L18 | //Add b4  pane.add(b5); |
| L19 | //Add b5 |
| L20 | } |
| L21 |  |
| L22 | public static void main(String[] args) { |

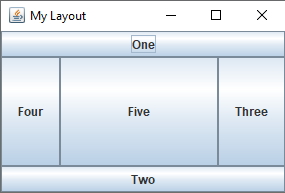
|  |  |
| --- | --- |
| L23 | MyLayout frame = new MyLayout(); |
| L24 | frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  My Layout |
| L25 | frame.setTitle(" "); |
| L26 | frame.setSize(300, 200); |
| L27 | frame.setVisible(true); |
| L28 | } |
| L29 | } |

1. Modify the line of statement of *Question 3 in L9* to set this Layout in Container (as shown below).



* pane.setLayout(new GridLayout(2,3));

1. a) Modify the line of statement of *Question 3 in L9, L15-L19* based on the following figure.



pane.getLayout();

|  |  |  |
| --- | --- | --- |
| L9 | //Set  pane.add(b1,BoarderLayout.NORTH); | Layout |
| L15 | //Add  pane.add(b2,BoarderLayout.SOUTH); | b1 |
| L16 | //Add  pane.add(b3,BorderLayout.EAST); | b2 |
| L17 | //Add  pane.add(b4,BoarderLayout.WEST); | b3 |
| L18 | //Add  pane.add(b5,BoarderLayout.CENTER); | b4 |
| L19 | //Add | b5 |

b) Based on *Question 5a)* sketch the output if we remove or comment statement(s)

1. L19
2. L18
3. L17 & L18
4. L6

|  |  |
| --- | --- |
| Remove/comment L19 | Remove/comment L18 |
|  |  |
| Remove/comment L17 & L18 | Remove/comment L16 |
|  |  |

1. Consider the following statements.

|  |  |
| --- | --- |
| L1 | import javax.swing.\*; |
| L2 | import java.awt.\*; |
| L3 |  |
| L4 | public class TextField extends JFrame { |
| L5 | private JLabel name, matric; |
| L6 | private JTextField textName, textMatric; |
| L7 |  |
| L8 | public TextField() |
| L9 | { |
| L10 | Container pane = getContentPane(); |
| L11 | pane.setLayout(new GridLayout(2,1)); |
| L12 |  |

|  |  |
| --- | --- |
| L13 | name = new JLabel("Name"); |
| L14 | textName = new JTextField(20); |
| L15 | matric = new JLabel("Matric No"); |
| L16 | textMatric = new JTextField(20); |
| L17 | pane.add(name); |
| L18 | pane.add(textName); |
| L19 | pane.add(matric); |
| L20 | pane.add(textMatric); |
| L21 | } |
| L22 |  |
| L23 | public static void main(String [] args) { |
| L24 | TextField frame = new TextField(); |
| L25 | frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
| L26 | frame.setTitle("Student Info"); |
| L27 | frame.setSize(300, 100); |
| L28 | frame.setVisible(true); |
| L29 | } |
| L30 | } |

* 1. How many components display in the frame? List the name and type of components involve based on the following statements.
* 4 components in total
* 2 labels : name , matricNo
* 2 Text Fields : textName , textMatric
  1. Sketch the expecting output



1. Consider the following statements.

|  |  |
| --- | --- |
| L1 | import javax.swing.\*; |
| L2 | import java.awt.\*; |
| L3 |  |
| L4 | public class CheckBox extends JFrame { |
| L5 | private JCheckbox java, c, python; |
| L6 |  |
| L7 | public CheckBox() |
| L8 | { |
| L9 | Container pane = getContentPane(); |
| L10 | pane.setLayout(new FlowLayout()); |

|  |  |
| --- | --- |
| L11 |  |
| L12 | java = new JCheckBox("Java"); |
| L13 | c = new JCheckBox("C++"); |
| L14 | python = new JCheckBox("Python"); |
| L15 | pane.add(java); |
| L16 | pane.add(C); |
| L17 | pane.add(python); |
| L18 | } |
| L19 |  |
| L20 | public static void main(String [] args) { |
| L21 | CheckBox Jframe = new CheckBox(); |
| L22 | frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
| L23 | frame.setTitle("Programming"); |
| L24 | frame.setSize(300, 100); |
| L25 | frame.setVisible(); |
| L26 | } |
| L27 | } |

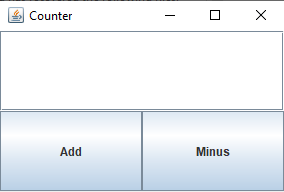
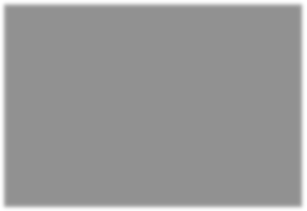
* 1. Trace and fix if have any error(s) in the above statements.
* L5 : private JCheckBox java, c, python;
* L16 : pane.add(c);
* L21 : CheckBox frame = new CheckBox();
* L25 : frame.setVisible(true);
  1. Sketch the expecting output



* 1. What the difference between checkbox and radio button?
* Check Box : represent mutually inclusive, square with a checkmark inside
* Radio Button : represent mutually exclusive selections, circle with a dot inside

1. Consider the following figure.

tDisplay



0

panel1

btnAdd

btnMinus

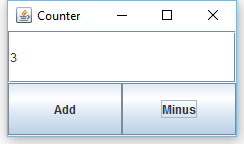
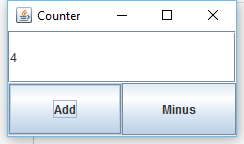
* 1. What is the purpose of panel1?
* To make together a combination a GUI components in the same position
  1. Explain the similar and different between frame and panel?
* The similarity is both the frame and panel are containers
* The difference is Frame has its own Title and ContentPane whereas panel doesn't have both of them.
  1. Complete the line of following statements.

|  |  |
| --- | --- |
| L1 | import javax.swing.\*; |
| L2 | import java.awt.\*; |
| L3 |  |
| L4 | public class Counter extends JFrame |
| L5 | {  private JPanel panel1; |
| L6 | //declare panel1  private JTextField tDisplay; |
| L7 | //declare tDisplay  private JButton btnAdd, btnMinus; |
| L8 | //declare btnAdd and btnMinus |
| L9 |  |
| L10 | public Counter() |
| L11 | { |
| L12 | Container pane = getContentPane(); //get content pane  pane.setLayout(new GridLayout(2,1)); |
| L13 | //set layout pane  tDisplay = new JTextField(“0”); |
| L14 | //create tDisplay |
| L15 | JPanel panel1 = new JPanel(): |
| L16 | //create panel1  panel1.setLayout(new GridLayout(2,1)); |
| L17 | //set layout panel1  btnAdd = new JButton(“Add”); |
| L18 | //create btnAdd  btnMinus = new JButton(‘Minus”); |
| L19 | //create btnMinus  panel1.add(btnAdd); |
| L20 | //add btnAdd in panel1  panel1.add(btnMinus); |
| L21 | //add btnMinus in panel1 |
| L22 |  |
| L23 | pane.add(tDisplay); |
| L24 | //add tDisplay in pane  pane.add(panel1); |
| L25 | //add panel in pane |
| L26 | } |
| L27 |  |
| L28 | public static void main (String[] args) |
| L29 | { |
| L30 | Counter frame = new Counter(); |
| L31 | frame.setTitle("Counter"); |

|  |  |
| --- | --- |
| L32 | frame.setSize(300, 200); |
|  |  |
| L33 | frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
| L34 | frame.setVisible(true); |
| L35 | } |
| L36 | } |

*Topic: Event Handling.*

1. Based on previous question *(Question 8),* write the code statements to handle event so that when the Add button in the GUI is clicked, the number in the box will increased by 2, meanwhile if Minus button is clicked, the number in the box will decrease by 1 as shown in the following figures.



**Step 1:** The previous class Counter should import event package (*add in L3*) and implement an event listener (*modify L4*).

L3 import java.awt.event.\*;

L4 **public class** Counter **extends** JFrame implements ActionListener{

**Step 2:** Register event listener with the appropriate event source (component in the GUI) in the previous constructor (write statement in L22-L23).

L22 btnAdd.addActionListener(this) ; L23 btnMinus.addActionListener(this ;

**Step 3:** Write the code to handle event as shown in the above figures (insert the statement before main method).

**public void** actionPerformed(ActionEvent e) {

Object obj = e.getSource();

tDisplay.getText(); ;

**if** (obj == btnAdd){

value=value + 2 ;

tDisplay.setText(“ “ + value) ;

}

**else**

{

value = value - 1 ;

tDisplay.setText(“ “ + value) ;

}

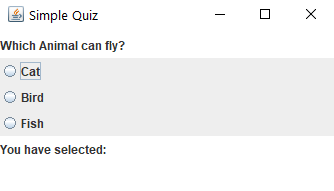
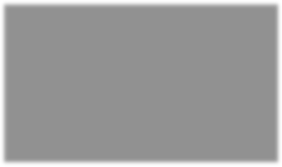
}

# Section B

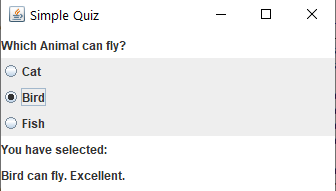
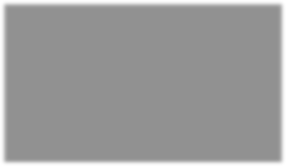
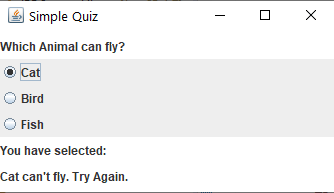
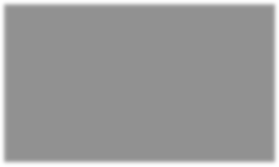
**Practice - Question 1**

Write the GUI code statements and event for the following figure. User can choose only one answer from the radio button. The selected answer will be display in the text field such as the following figure. (Set frame size 350, 200).

Before user click radio button



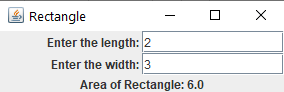
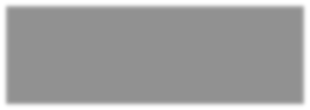
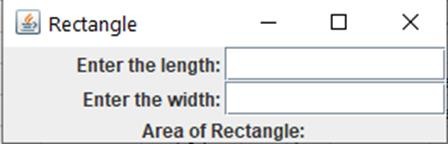
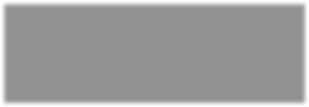
After user click radio button



# Practice - Question 2

Write the GUI code statements and event for the following figure. The output of rectangle area should be display such as the following figure after user insert the data and click “enter”. (Set frame size 300, 100).

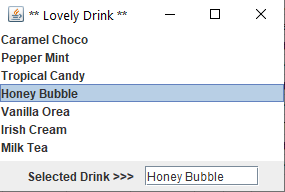
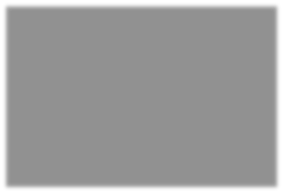
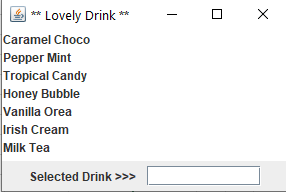
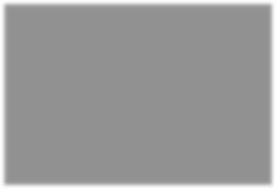
Before user insert the data: After user insert the data and click enter:



# Practice - Question 3

Write the GUI code statements and event for the following figure. Customer can choose from the list of “Lovely Drink”. The selected item will be display in the text field such as the following figure. (Set frame size 300, 200).

Before selection drink After selection drink



# Assignment – Question 4

A company named ‘Create Your Own Pizza’ is a small and famous homemade pizza in rural area at Kelantan. Since the company doesn’t have enough staff and the pizza is hot selling due to very cheap and tasty, the company only limited one pizza for only one customer. The company has a very traditional way on ordering pizza. Only one staff will handle the order and the company doesn’t have any system users to order the pizza. You as a freelance programmer try to help them to develop a simple order system for Create Your Own Pizza. The system should have some graphical user interface and related event for ordering the pizza. Figure 1 below show the GUI of the system.

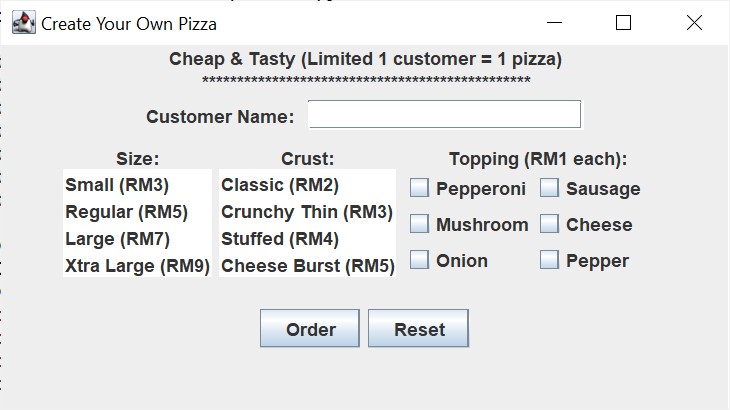


Figure 1

Customer must enter name, select size, crust and topping of pizza. The figure 2 below show the GUI when the form completed and order button clicked.

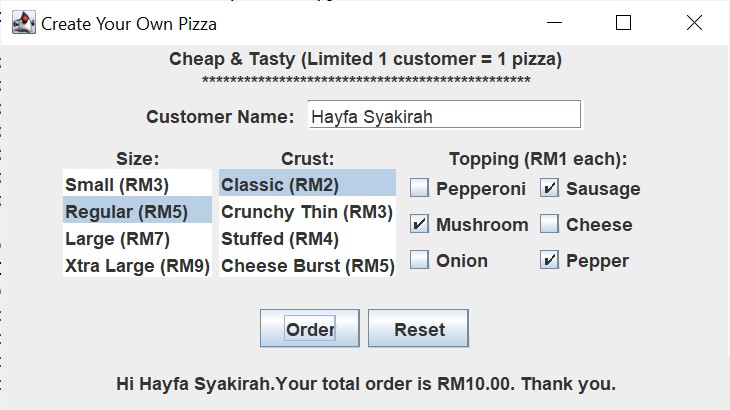


Figure 2

If the reset button clicked, all the data selected and the name filled before will be reset and it will produce the following GUI.

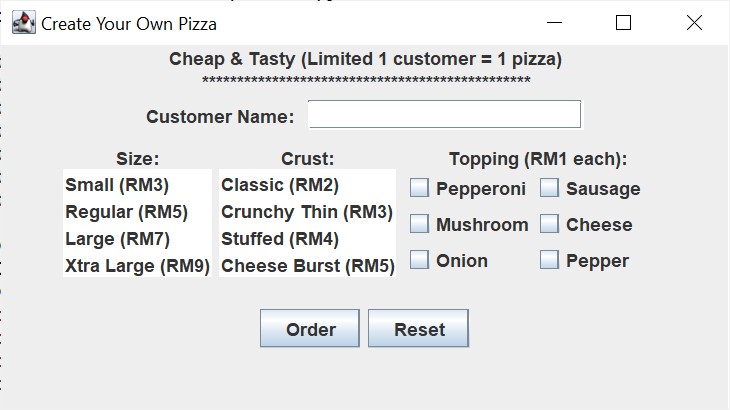
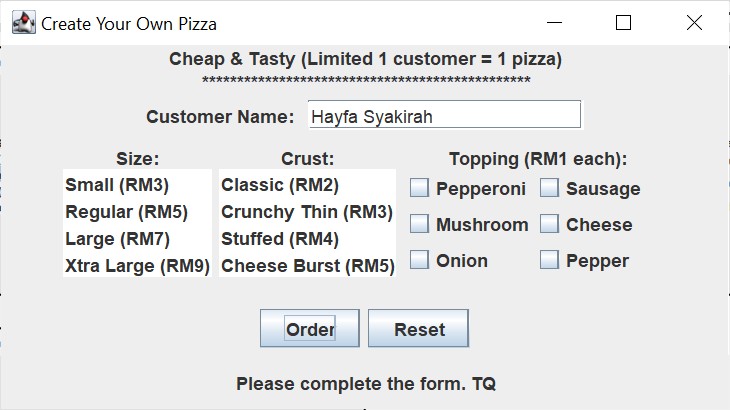
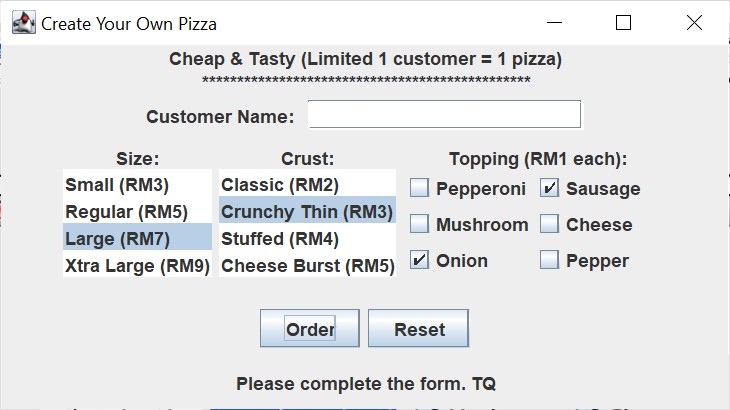
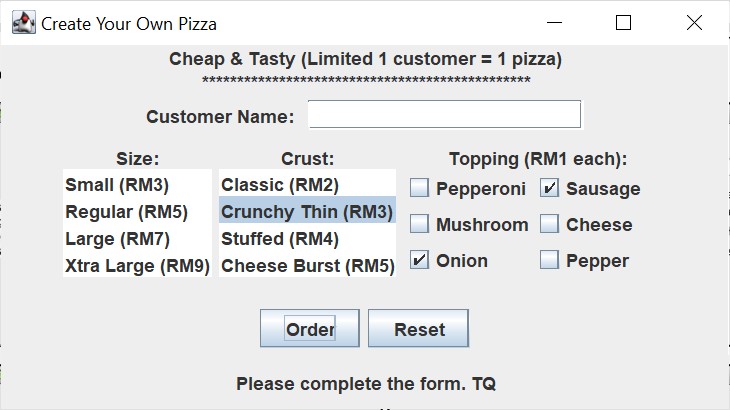
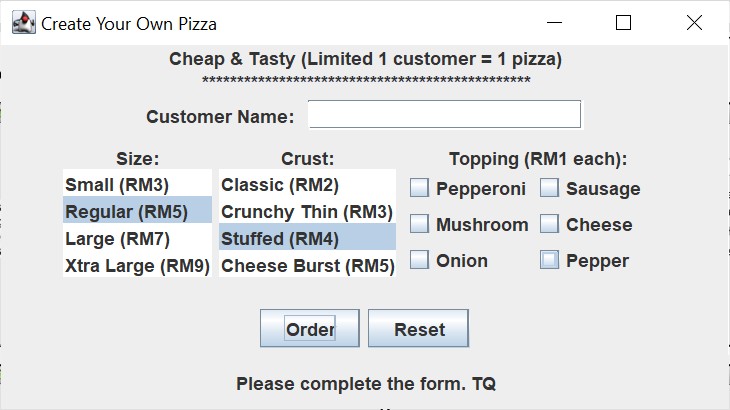
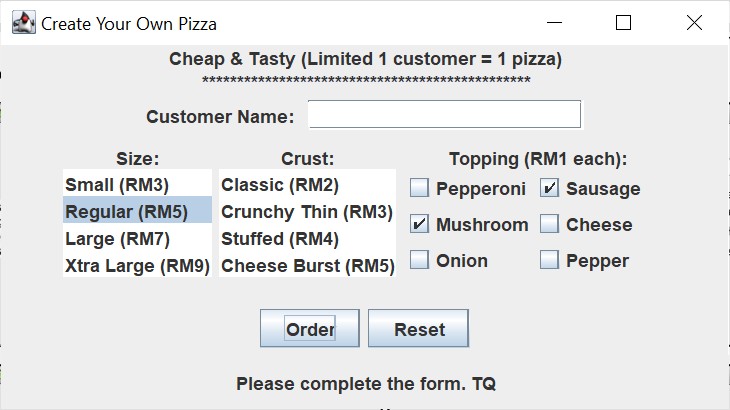
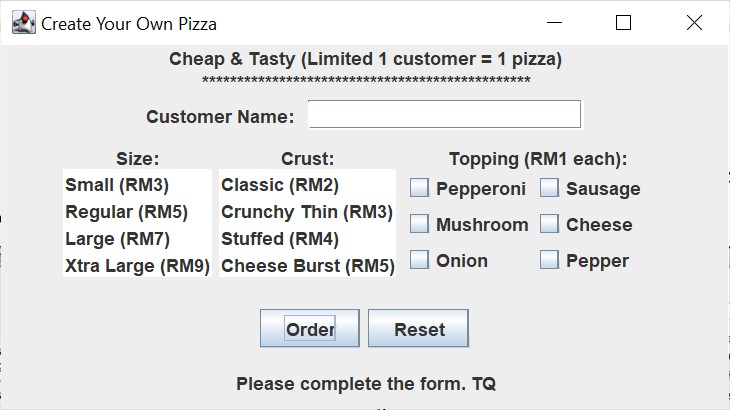


Figure 3

If the form is not completed such as the following figure and the order button is clicked, the feedback will display at the bottom with the message “Please complete the form. TQ”



Note:

* Pizza Size and Crust only can be selected by ONE selection.
* Topping can be selected by multiple selections.