CIS355A Lab Report

**Your Name:**

**Date:**

**Lab Week \_\_\_\_**

**Objective/Purpose of the program**

Briefly describe the program’s requirements.

Objective of this assignment is to create point of sale application for burgerRU. Application shall be developed using swing components and achieve the real time sales orders.

**Analysis/Design**

Describe the approach/structure of program.What classes/functions were used?

Application is swing based component, which captures data from swing user interface and performs several operations based on user input. System has several components such as radio buttons, checkboxes , text areas , text fields and all the 9 fields shall be kept in sync when operations are performed such as add new order.

**Testing/Results**

Does your program satisfy all requirements of the lab?

If yes, how did you test it? Indicate test cases used, expected values, and *show results with screen shots.* **For example:**

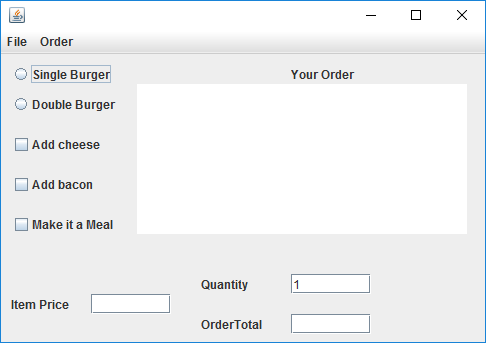
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Case** | **Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1 | Describe what you are testing | Specify the test data | What is expected | Actual results – reference an image that you include below. |
| 1 | Item Price | Select burger type | Item price is displayed correctly  Quantity displayed correctly  Order total displayed correctly | Item price displayed correctly  Quantity displayed correctly  Order total displayed correctly |
| 2 | Item Price | Add Cheese | Item price is displayed correctly  Quantity displayed correctly  Order total displayed correctly | Item price displayed correctly  Quantity displayed correctly  Order total displayed correctly |
| 3 | Item Price | Add Bacon | Item price is displayed correctly  Quantity displayed correctly  Order total displayed correctly | Item price displayed correctly  Quantity displayed correctly  Order total displayed correctly |
| 4 | Item Price | Make it a meal | Item price is displayed correctly  Quantity displayed correctly  Order total displayed correctly | Item price displayed correctly  Quantity displayed correctly  Order total displayed correctly |
| 5 | Order Total | quantity | Order total shall be quantity x item price | Order total displayed accurately |
| 6 | Add to order | When clicked on Add to order | Order text is displayed in text area when clicked on Add to Order | Order text is displayed in text area when clicked on Add to Order |
| 7 | New Order | Click on New Order | All the previous data shall be cleared |  |
| 8 | Clear for next item | Click on Clear | Previous items shall be reset | Item total is reset and selection of checkboxes and radio buttons are reset. |

If any requirements are NOT met, *document the known issues.* What did you do to try to solve them?

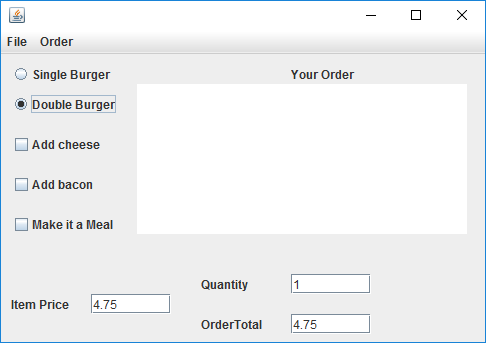
Make sure you demonstrate in your testing the parts that are working correctly.

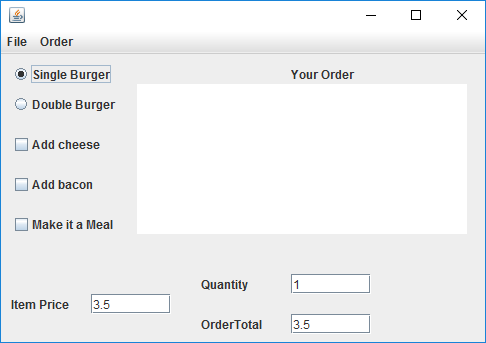
**GUI Screenshots**

1. **Home Screen**

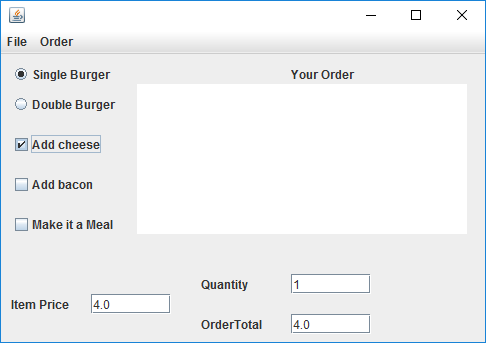
****

1. **Burger Selection**

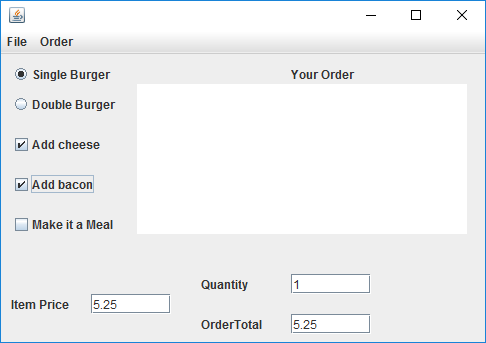
****

****

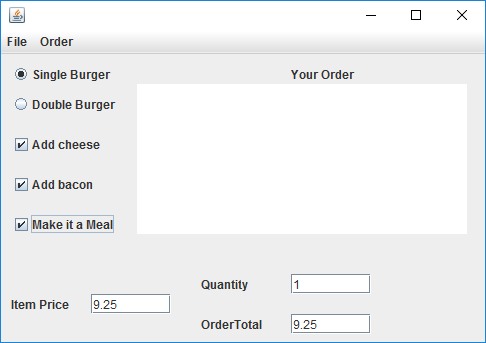
1. **Cheese Selected**

****

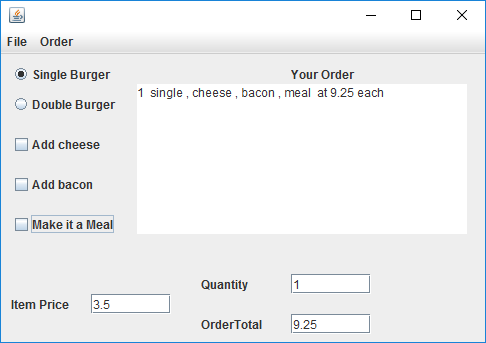
1. **Bacon Selected**

****

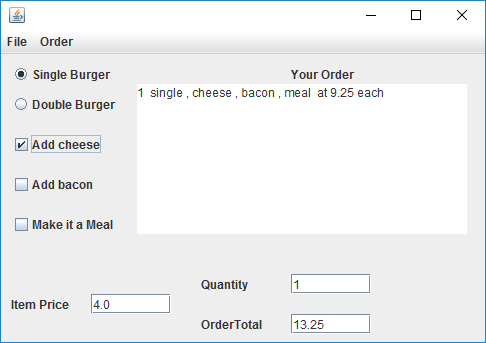
1. **Meal Selection**

****

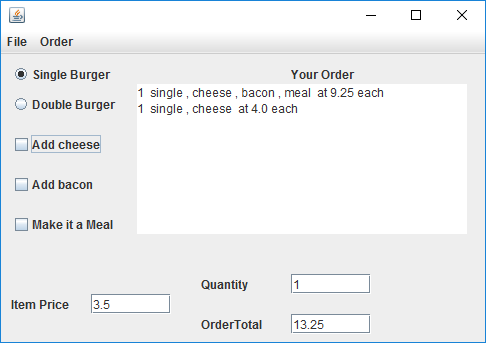
1. **Order Added**

****

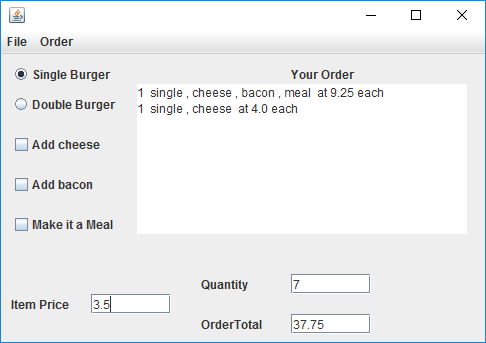
1. **Second Item to Order**

****

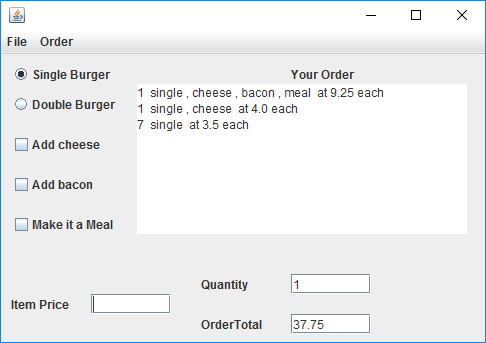
1. **Add Second Item**

****

1. **Quantity Selection**

****

1. **Add to order with quantity to more than 1**

****

**SOURCE CODE**

Table

|  |
| --- |
| BurgerPOS |
| **package** burger.pos;  **import** java.awt.Color;  **import** java.awt.event.ActionEvent;  **import** java.awt.event.ActionListener;  **import** java.awt.event.FocusEvent;  **import** java.awt.event.FocusListener;  **import** java.awt.event.ItemEvent;  **import** java.awt.event.ItemListener;  **import** javax.swing.ButtonGroup;  **import** javax.swing.JCheckBox;  **import** javax.swing.JFrame;  **import** javax.swing.JLabel;  **import** javax.swing.JMenu;  **import** javax.swing.JMenuBar;  **import** javax.swing.JMenuItem;  **import** javax.swing.JRadioButton;  **import** javax.swing.JTabbedPane;  **import** javax.swing.JTextArea;  **import** javax.swing.JTextField;  /\*\*  \* Burger pos application to display the orders and calculate the order amount  \*  \* **@author** Admin  \*  \*/  **public** **class** BurgerPOS **extends** JFrame **implements** ActionListener, ItemListener, FocusListener {  **private** **static** JFrame *frame*;  **private** **final** JLabel lblNewLabel = **new** JLabel(" Burger POS ");  **private** **static** JTabbedPane *tabbedPane* = **null**;  **static** BurgerPOS *burgerPos*;  **private** **static** JRadioButton *sbradio*, *dbradio*;  **private** **static** JCheckBox *addCheese*, *addBacon*, *makeMeal*;  **private** **static** JTextField *itemPrice*, *quantity*, *orderTotal*;  **private** **static** JTextArea *orderDetails*;  **static** ButtonGroup *bgroup*;  **static** **int** *itemNumber*;  **static** **double** *orderValue* = 0;  **double** itemPriceVal = 0;  **static** **int** *burgerSelected* = 0;  **static** JMenu *file*;  **static** JMenu *order*;  **static** JMenuItem *exit*, *addOrder*, *clear*, *neworder*;  **double** previousBalanceChanges = 0;  /\*\*  \* Launch the application.  \*/  **public** **static** **void** main(String[] args) {  **try** {  *frame* = **new** JFrame();  *burgerPos* = **new** BurgerPOS();  *sbradio* = **new** JRadioButton();  *dbradio* = **new** JRadioButton();  *sbradio*.setText("Single Burger");  *dbradio*.setText("Double Burger");  *sbradio*.setBounds(10, 10, 120, 20);  *dbradio*.setBounds(10, 40, 120, 20);  *bgroup* = **new** ButtonGroup();  *bgroup*.add(*sbradio*);  *bgroup*.add(*dbradio*);  *sbradio*.setActionCommand("1");  *dbradio*.setActionCommand("2");  *sbradio*.addActionListener(*burgerPos*);  *addCheese* = **new** JCheckBox("Add cheese", **false**);  *addBacon* = **new** JCheckBox("Add bacon", **false**);  *makeMeal* = **new** JCheckBox("Make it a Meal");  *addCheese*.setBounds(10, 80, 120, 20);  *addBacon*.setBounds(10, 120, 120, 20);  *makeMeal*.setBounds(10, 160, 120, 20);  *itemPrice* = **new** JTextField();  *quantity* = **new** JTextField();  *orderTotal* = **new** JTextField();  *orderDetails* = **new** JTextArea();  JLabel itempricelbl = **new** JLabel("Item Price");  JLabel quantitylbl = **new** JLabel("Quantity");  JLabel orderTotallbl = **new** JLabel("OrderTotal");  JLabel yourOrderlbl = **new** JLabel("Your Order");  itempricelbl.setBounds(10, 240, 120, 20);  *itemPrice*.setBounds(90, 240, 80, 20);  quantitylbl.setBounds(200, 220, 280, 20);  *quantity*.setBounds(290, 220, 80, 20);  *quantity*.setText("1");  orderTotallbl.setBounds(200, 260, 100, 20);  *orderTotal*.setBounds(290, 260, 80, 20);  *sbradio*.addActionListener(*burgerPos*);  *dbradio*.addActionListener(*burgerPos*);  *addCheese*.addItemListener(*burgerPos*);  *addBacon*.addItemListener(*burgerPos*);  *makeMeal*.addItemListener(*burgerPos*);  *quantity*.addFocusListener(*burgerPos*);  yourOrderlbl.setBounds(290, 10, 100, 20);  *orderDetails*.setBounds(136, 30, 330, 150);  *file* = **new** JMenu("File");  *order* = **new** JMenu("Order");  *neworder* = **new** JMenuItem("New Order");  *addOrder* = **new** JMenuItem("Add to Order");  *clear* = **new** JMenuItem("Clear for next item");  *exit* = **new** JMenuItem("Exit");  *file*.add(*exit*);  *order*.add(*neworder*);  *order*.add(*addOrder*);  *order*.add(*clear*);  JMenuBar mb = **new** JMenuBar();  mb.add(*file*);  mb.add(*order*);  *frame*.add(*sbradio*);  *frame*.add(*dbradio*);  *frame*.add(*addCheese*);  *frame*.add(*addBacon*);  *frame*.add(*makeMeal*);  *frame*.add(itempricelbl);  *frame*.add(*itemPrice*);  *frame*.add(quantitylbl);  *frame*.add(*quantity*);  *frame*.add(orderTotallbl);  *frame*.add(*orderTotal*);  *frame*.add(yourOrderlbl);  *frame*.add(*orderDetails*);  *frame*.setJMenuBar(mb);  *frame*.setLayout(**null**);  *frame*.setBackground(Color.***LIGHT\_GRAY***);  *frame*.setBounds(189, 53, 500, 350);  *frame*.setVisible(**true**);  *addOrder*.addActionListener(*burgerPos*);  *clear*.addActionListener(*burgerPos*);  *neworder*.addActionListener(*burgerPos*);  *exit*.addActionListener(*burgerPos*);  } **catch** (Exception e) {  e.printStackTrace();  }  }  /\*\*  \* Create the application.  \*/  **public** BurgerPOS() {  initialize();  }  /\*\*  \* Initialize the contents of the frame.  \*/  **private** **void** initialize() {  }  @Override  **public** **void** itemStateChanged(ItemEvent e) {  // **TODO** Auto-generated method stub  calculateValue();  }  @Override  **public** **void** actionPerformed(ActionEvent e) {  System.***out***.println(e.getSource());  **if** (e.getSource() == *neworder*) {  *itemPrice*.setText("");  *quantity*.setText("1");  *orderTotal*.setText("");  *addCheese*.setSelected(**false**);  *addBacon*.setSelected(**false**);  *makeMeal*.setSelected(**false**);  previousBalanceChanges=0;  *orderDetails*.setText("");  **return**;  }  System.***out***.println(" New order returned;");  **if** (e.getSource() == *clear*) {  *itemPrice*.setText("");  *quantity*.setText("1");  *orderTotal*.setText("");  *addCheese*.setSelected(**false**);  *addBacon*.setSelected(**false**);  *makeMeal*.setSelected(**false**);  **return**;  }  **if** (e.getSource() == *exit*) {  System.*exit*(1);  }  **if** (e.getSource() == *addOrder*) {  **double** preorderitem = Double.*parseDouble*(*itemPrice*.getText());  **int** preorq = Integer.*parseInt*(*quantity*.getText());  **double** totalCurrentVal = preorderitem \* preorq;  previousBalanceChanges += totalCurrentVal;  String textRet = calculateValue();  *orderDetails*.setText(*orderDetails*.getText() + textRet + "\n");  *itemPrice*.setText("");  *quantity*.setText("1");  *addCheese*.setSelected(**false**);  *addBacon*.setSelected(**false**);  *makeMeal*.setSelected(**false**);  *orderTotal*.setText(previousBalanceChanges + "");  **return**;  }  calculateValue();  }  **public** **void** focusLost(FocusEvent e) {  calculateValue();  }  **private** String calculateValue() {  itemPriceVal = 0;  **boolean** sburg = **false**;  **boolean** dburg = **false**;  **boolean** baddCheese = **false**;  **boolean** baddBacon = **false**;  **boolean** bisMeal = **false**;  String orderText = "";  **if** (*bgroup*.getSelection() != **null**) {  **if** ("1".contentEquals(*bgroup*.getSelection().getActionCommand())) {  sburg = **true**;  itemPriceVal += 3.50;  orderText = " single ";  }  **if** ("2".contentEquals(*bgroup*.getSelection().getActionCommand())) {  dburg = **true**;  itemPriceVal += 4.75;  orderText = " double cheese ";  }  }  **if** (*addCheese*.isSelected()) {  baddCheese = **true**;  itemPriceVal += 0.50;  **if** (orderText != **null**)  orderText += ", cheese ";  **else**  orderText += ", cheese ";  }  **if** (*addBacon*.isSelected()) {  baddBacon = **true**;  itemPriceVal += 1.25;  **if** (orderText != **null**)  orderText += ", bacon ";  **else**  orderText += ", bacon ";  }  **if** (*makeMeal*.isSelected()) {  bisMeal = **true**;  itemPriceVal += 4.00;  **if** (orderText != **null**)  orderText += ", meal ";  **else**  orderText += " meal ";  }  *itemPrice*.setText(itemPriceVal + "");  **int** quant = 1;  **try** {  quant = Integer.*parseInt*(*quantity*.getText());  } **catch** (Exception e) {  *quantity*.setText(1 + "");  }  String previousvals = *orderTotal*.getText();  **double** previousval = 0;  /\*  \* if(previousvals!=null&previousvals.trim().length()>0) previousval =  \* Double.parseDouble(previousvals);  \* System.out.println("Previous Balance= "+previousval);  \*/  *orderTotal*.setText(((quant \* itemPriceVal) + previousBalanceChanges) + "");  System.***out***.println("Total Balance= " + *orderTotal*.getText());  **return** quant + " " + orderText + " at " + itemPriceVal + " each";  }  @Override  **public** **void** focusGained(FocusEvent e) {  // **TODO** Auto-generated method stub  }  } |

**Conclusions / Lessons Learned**

What difficult problems did you encounter, and how did you handle them?

* Calculations for every user event is most challenging

What new concepts did you learn/reinforce with this lab?

* How to keep different components in sync and keep local and global variables

Is there anything you would have done differently?

* Custom event handler