# ***Diploma in Java Project***

**The project must be submitted by 8AM on the deadline date (see below), by emailing to brian.v.rogers@gmail.com. Late submissions will not be marked.**

**DEADLINE IS 15 September 2014 at 8am**

**In exceptional circumstances the director of the Fitzwilliam Institute may agree to extensions of the deadline. This must be agreed before the deadline.**

**All work must be your own, you may be asked to attend an interview to explain your project.**

**You may submit each part as you go or you may submit entire project when completed. Marks will be based on the final version you submit.**

## Deliverables

**Your project should be submitted as Java source files only. You do not need to provide a database.**

**This should then be submitted as a zip file by email to brian.v.rogers@gmail.com. You will receive an emailed acknowledgement within 48 hours. If you do not receive this acknowledgement please resubmit.**

**ALL the files necessary to compile the application as it exists after each part. Each source file should contain your name, email address and phone number as a comment.**

# Marking Schema

**There will be three parts to the assessment for this course. These are listed below.**

1. **A written test for which 20% of the marks will be awarded.**
2. **This programming project for which 70% of the marks will be awarded**
3. **A further 10% will be awarded for course attendance.**

**There are 4 parts to the programming project Marks will be allocated for each section as follows:**

| | Mark | % for project | % for all tests |
|---|---|---|---|
| Part 1 | 50 | 12.50% | 8.75% |
| Part 2 | 150 | 37.50% | 26.25% |
| Part 3 | 50 | 12.50% | 8.75% |
| Part 4 | 150 | 37.50% | 26.25% |
| | **400** | **100.00%** | **70.00%** |

| | |
|---|---|
| **Distinction** | **81% to 100 %** |
| **Credit** | **66 to 80%** |
| **Pass** | **40 to 65%** |

# Application Overview Phase 1

A restaurant wants you to develop a windowed (JFrame) application that calculates a customer's bill. The application is to be implemented in two phases. What follows is an overview of Phase 1.

The application should display all the menu items from the restaurant's database in four JComboBoxes. Each JComboBox should contain a category of food offered by the restaurant (**Beverage**, **Appetizer**, **Main Course** and **Dessert**). The user can choose one item from each of these JComboBoxes to add items to a cutomer's bill. When the customer is finished ordering, the user can click the **Calculate Bill** JButton to display the **Subtotal:**, **Tax:** and **Total:** for the table.

The provided restaurant database contains one table, menu, which has four columns—itemID, name, category and price. The values stored in the itemID column are ints. The values stored in the name and category columns are Strings. The values stored in the price column are doubles. You are provided with the file DipJavaProject.sql. This file contains and SQL script that will create the database in mySQL.

You are also provided with 3 CSV files that will populate the database tables created by the script file.

### Miscellaneous Important Points

### Functionality

The program should have all functionality described in this document.

### Exception Handling

The application should have sufficient exception handling to ensure robust operation

**ALL SOURCE CODE SHOULD INCLUDE EXTENSIVE COMMENTS**

**ALL SOURCE CODE FILES SHOULD INCLUDE YOUR NAME AND EMAIL ADDRESS.**

**FOLLOW INSTRUCTIONS PRECISELY TO ENSURE MAXIMUM MARKS.**

# PART 1 (50 Marks)

### Part 1 Create the Graphical User Interface

Create the graphical user interface as shown on the following page. Provide only the functionality required to the display the completed JFrame and terminate the program if the JFrame is closed.

**To help you along some of the code required for this section is already provided in the template. Most of the information required to complete this section is covered in Module 14 GUI Components.**

No layout manager should be used.

All GUI components should be positioned by program code. To do this, remove the default layout manager from the each Component/Container used. This can be done by using method setLayout and passing null to this method

All the GUI variables required are supplied in the code template. All the method headers are also provided.

Once you have removed the layout manager you can position each GUI item by using method setbounds, for example

```
beverageJLabel = new JLabel();
beverageJLabel.setBounds( 8, 24, 80, 24 );
```

Use a clear, easy to read font to display the information. Apply this font to all components.

Display the word **Restaurant** in a JLabel.

Create a JPanel waiterJPanel to display the Table Number and Waiter Name.

Create a JPanel named **menuItemsJPanel** to display the Beverage, Appetizer, Main Course and Desert combo boxes and associated labels.

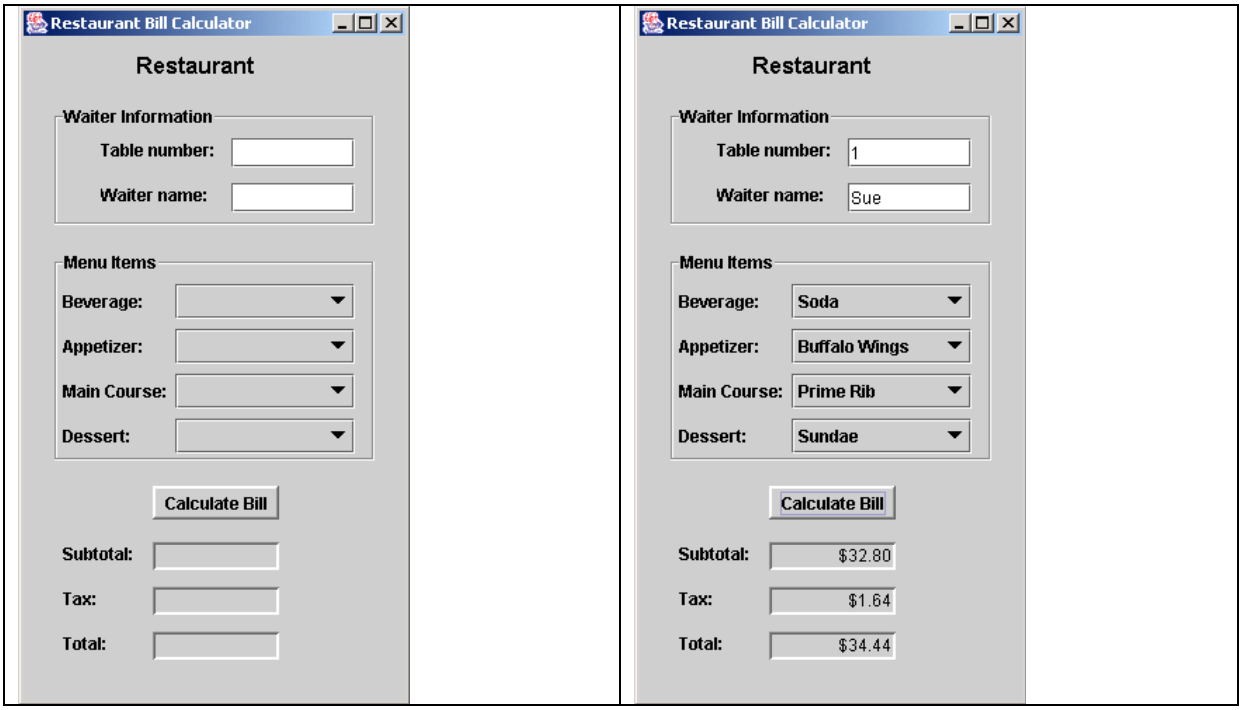
Create a JButton to Calculate the bill, name it **calculateBillJButton.**

Then create text fields and labels to display the subtotal and total.

The GUI is illustrated in the screen shots on the following page

As you work through the project modify the import Statements provided so that only classes used are actually imported. The imports shown below should not appear in the final project.

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.swing.*;
```



# Project PART 2 (150 Marks)

**Part 2 Interacting with the Database and Calculations.**

**Database to Be Used**
The only database to be used is mySQL Version 5 or later. You are provided with SQL scripts to create the database in mySQL.

**To help you along some of the code required for this section is already provided in the template. Most of the information required to complete this section is covered in Module 28 Accessing Databases and early modules.**

**Starting the program**

It should be possible to call the program from the command line and pass two String arguments to the main method. The first argument should specify the **database username** the second argument should specify the **database password**. If these arguments are not provided the program should display a message box telling the user that the Username/Password is incorrect. When the user clicks OK on the message box the program should exit.

**Database Functionality**

To create the database functionality you need to carry out the step listed below.

***Adding a database connection and creating a Statement object.***
In the RestaurantBillCalculator constructor, insert statements that load the database driver, connect to the restaurant database and create a Statement to submit SQL to the database. Assume that the database password and user name are passed to the constructor from main method. Three instance variables—myConnection, myStatement and myResultSet should be declared

***Adding code to the loadCategory method.***
Create the loadCategory method, which immediately follows createMenuItemsJPanel. The loadCategory method takes a String, representing the category to load, and the name of the JComboBox to add items to as arguments. Add a statement that queries the database and retrieves the name column from the menu table for the specified category. Insert a loop that processes the ResultSet and adds each name to the categoryJComboBox. Close the ResultSet after the loop.

***Adding code to the beverageJComboBoxItemStateChanged method.***
Create the beverageJComboBoxItemStateChanged method (which immediately follows loadCategory) and insert code that adds the String representation of the selected item to the ArrayList billItems. [*Hint*: Use the ItemEvent.SELECTED constant to determine whether an item is selected.]

***Adding code to the appetizerJComboBoxItemStateChanged method.***
Create the appetizerJComboBoxItemStateChanged method (which immediately follows beverageJComboBoxItemStateChanged) and insert code that adds the String representation of the selected item to the ArrayList billItems. [*Hint*: Use the ItemEvent.SELECTED constant to determine whether an item is selected.]

***Adding code to the mainCourseJComboBoxItemStateChanged method.***
Create the mainCourseJComboBoxItemStateChanged method (which immediately follows appetizerJComboBoxItemStateChanged) and insert code that adds the String representation of the selected item to the ArrayList billItems. [*Hint*: Use the ItemEvent.SELECTED constant to determine whether an item is selected.]

***Adding code to the dessertJComboBoxItemStateChanged method.***
Create the dessertJComboBoxItemStateChanged method (which immediately follows main-CourseJComboBoxItemStateChanged) and insert code that adds the String representation of the selected item to the ArrayList billItems. [*Hint*: Use the ItemEvent.SELECTED constant to determine whether an item is selected.]

***Adding code to the calculateBillJButtonActionPerformed method.*** Create the calculateBillJButtonActionPerformed method (which immediately follows dessertJComboBoxItemStateChanged) and add code to ensure that a table number (tableNumberJTextField) and waiter name (waiterNameJTextField) have been entered. If one of these fields is empty, display a JOptionPane informing the user that both fields must contain information. Otherwise, call the calculateSubtotal method, which you implement in the next step, to calculate the subtotal of the bill. The calculateSubtotal method takes no arguments and returns a double containing the subtotal, which you should display in subtotalJTextField. Calculate and display the tax and the total of the bill in JTextFields taxJTextField and totalJTextField, respectively. The tax rate is specified in a constant TAX_RATE.

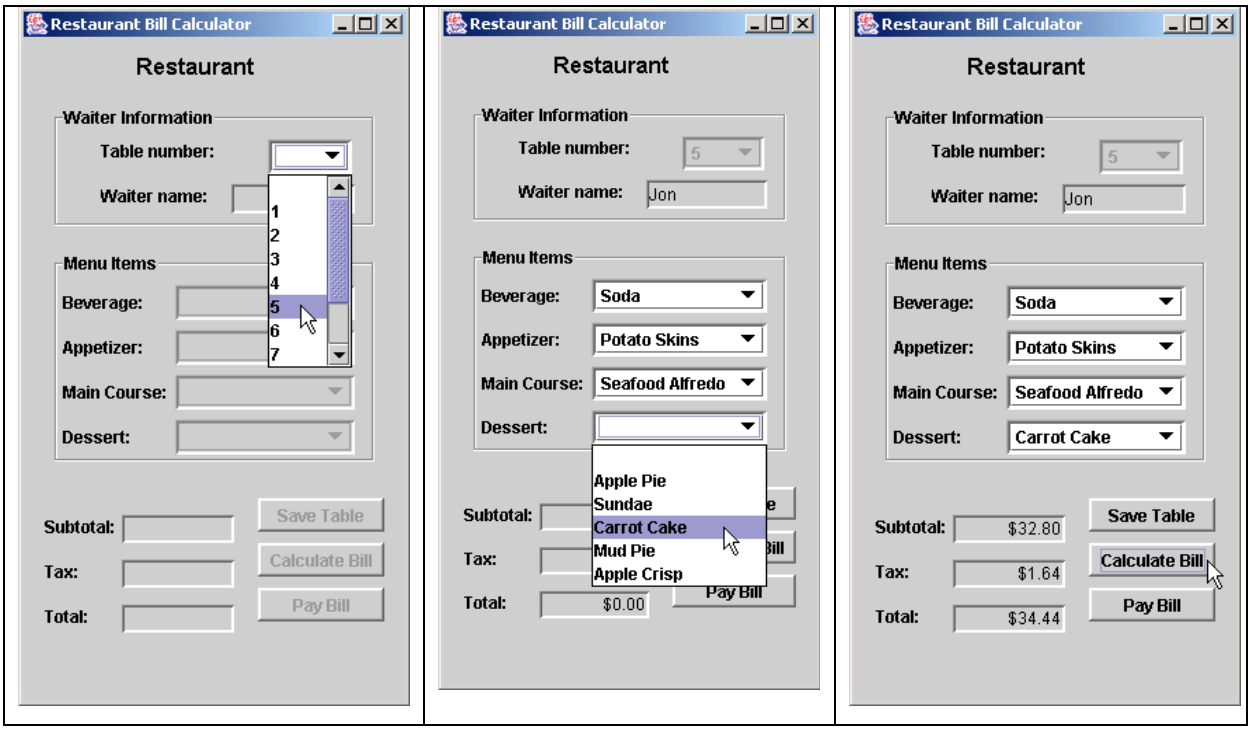
***Adding code to the calculateSubtotal method.***
Create the calculateSubtotal method (which immediately follows calculateBillJButtonActionPerformed) and add code that queries the database and retrieves the price column for all the menu items in the billItems ArrayList. This method should then calculate the total price of all the items in the ArrayList and return this value as a double.

***Closing the database connection.***
Create the frameWindowClosing method (which is located just before main at the end of the class) and add code that closes myStatement and myConnection.

# Application Overview Phase 2

Create an ***Enhanced* Restaurant Bill Calculator *Application***

Modify the application you developed in Part 1 and Part 2 to keep track of multiple table bills at the same time. Sample outputs are shown in the screen shots below. The user should be able to calculate a bill for a table and save that table's subtotal and waiter's name. The user should also be able to retrieve that information at a later time. [*Hint*: The restaurant database contains two tables, one for the menu items, as before, and another (restaurantTables) for all the tables in the restaurant. The restaurantTables table has three columns—tableNumber, subtotal and waiterName. The values in the tableNumber column are ints. The values in the subtotal column are doubles. The values in the waiterName column are Strings.]



# Project PART 3 (50 Marks)

### Part 3 Modifications to the Graphical User Interface

Replace the JTextField which is currently used for the table number with a JComboBox. This new combo box will eventually be populated by the database. Do not add any functionality yet.

Add a **Save Table** Button which will Save the Waiter Name and Total to the Database. Do not add any functionality yet.

Add a **Pay Bill** Button which will mark the bill as paid in the database. Do not add any functionality yet.

All three command buttons should be disabled by default.

Create method ***resetFrame*** which will do the following

- reset instance variable billItems
- reset and disable menuItemsJPanel
- reset and enable waiterJPanel
- clear JTextFields
- disable JButtons

Instructions for using this method are given later.

# Project PART 4 (150 Marks)

### Part 4 Modifications to Database Interaction Features

### Adding code to the loadTableNumbers method.

Create the loadTableNumbers method, which immediately follows createMenuItemsJPanel. In the loadTable-Numbers method, add a statement that queries the database and retrieves the tableNumber column from the restaurantTables table. Insert a loop that processes the ResultSet and adds each table number to the tableNumberJComboBox.

### Adding code to the tableNumberJComboBoxItemStateChanged method.

Create the tableNumberJComboBoxItemStateChanged method, which immediately follows loadCategory. In the tableNumberJComboBoxItemStateChanged method, add a statement that queries the database and retrieves all the columns from the restaurantTables table for the table that is selected from the JComboBox. Process the ResultSet and display the waiter's name in waiterNameJTextField. Call the displayTotal method with the subtotal retrieved from the database, which is provided in the template, to display the subtotal, tax and total in the subtotalJTextField, taxJTextField and totalJTextField, respectively. At the end of the tableNumberJComboBoxItemStateChanged method, enable the menuItemsJPanel and all JComboBoxes in it and disable the waiterJPanel and the tableNumberJCombox in it. Finally, enable the saveTableJButton, calculateBillJButton and payBillJButton.

### Adding code to the saveTableJButtonActionPerformed method.

Create the saveTableJButtonActionPerformed method, which immediately follows dessertJComboBoxItemStateChanged. In the saveTableJButtonActionPerformed method, assign the double value returned by the calculateSubtotal method (which you must create) to instance variable subtotal. The method should then call the updateTable method (which will be created later) to update the database. Lastly the method should call the resetJFrame method (which is already declared in the program) to reset the components in the JFrame to their initial setting.

### Adding code to the payBillJButtonActionPerformed method.

Create the payBillJButtonActionPerformed method, which immediately follows saveTableJButtonActionPerformed. The payBillJButtonActionPerformed method should reset subtotal to zero, call the updateTable method (which is created in the next step) to update the database and call the resetJFrame method (which you created in Part 3) to reset the components in the JFrame to their initial setting.

### Creating the updateTable method.

Immediately following the payBillJButtonActionPerformed method, create the updateTable method, which does not take any arguments and does not return anything. In the updateTable method, add a statement that updates the subtotal column with the value stored in the instance variable subtotal for the selected table number.