Criterion C: Development

List of Advanced Techniques

	Complexities	Justification	Evidence (in pages)
1	Access to develop a backend database	This was done in two instances. First of which was to develop a database to collect the details of customers. Another database was also formed to store the customer's order.	3-6
2	Validation checks	Several validation checks were incorporated in order to ensure quality data.	6-8
3	SQL commands	Data is fetched from the database and is displayed in a table in the Jframe to produce a bill.	8
4	Graphical User Interface	Jframe's were created in order to provide a user friendly experience.	8
5	Techniques to enable easy navigation	Various buttons were added to ensure a smooth transfer from one JFrame to another.	9

6	Object definition	New objects are created to connect different classes.	10
7	Polymorphism	Method overloading is implemented in order to satisfy different circumstances when connecting Jframe's.	10-11
8	Integrating media	Images of food are added to enhance customer knowledge.	11-12
9	Subroutines	Accessor and Mutator methods are used in order to transfer information.	12-13
10	Exception Handling	This was implemented in order to ensure smooth running of the execution of the program.	13
11	Calculated fields	In order to calculate the total amount due at the end of the meal.	14

1] Access to develop a backend database

a) To store customer details

Code to connect software to database.

Connected using JDBC-ODBC Driver

```
17
18      public homepage() {
19          initComponents();
20          try{
21              String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
22              Class.forName(driver);
23              String login = "jdbc:odbc:23";
24              con = DriverManager.getConnection(login);
25              System.out.print("connected");
26          }
            catch(Exception ex) {}
28      }
```

Database called 23

Once connection is established "connected" is printed.

Exception Handling

Code to transfer customer input into database

```
232     try{
233
234         ps = con.prepareStatement("Insert into 23(FirstName,Surname,EmailID,TableNo,Birthday)values(?,?,?,?,?)");
235         ps.setString(1, Name.getText());
236         ps.setString(2, jTextField4.getText());
237         ps.setString(3, email.getText());
238         ps.setString(4, TNO.getSelectedItem().toString());
239         ps.setString(5, date.getDate().toString());
240         ps.executeQuery();
241     }
        catch(Exception ex) {}
243
```

5 different fields

Exception handling

The customers inputs personal details which then get saved as records in the database below:

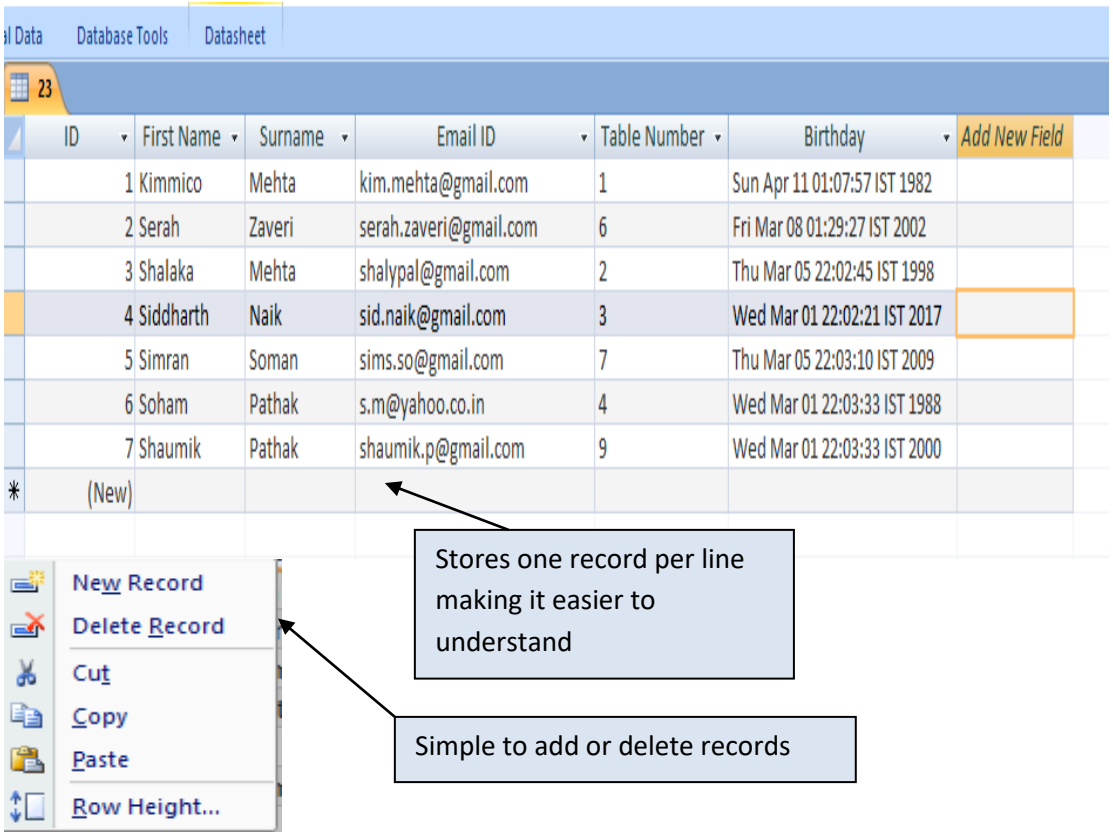

Figure 1: Records in database

In MS Access, the user can also create queries using “Query Wizard” to search for information. A query of the surname “Mehta” is shown in Figure 2.

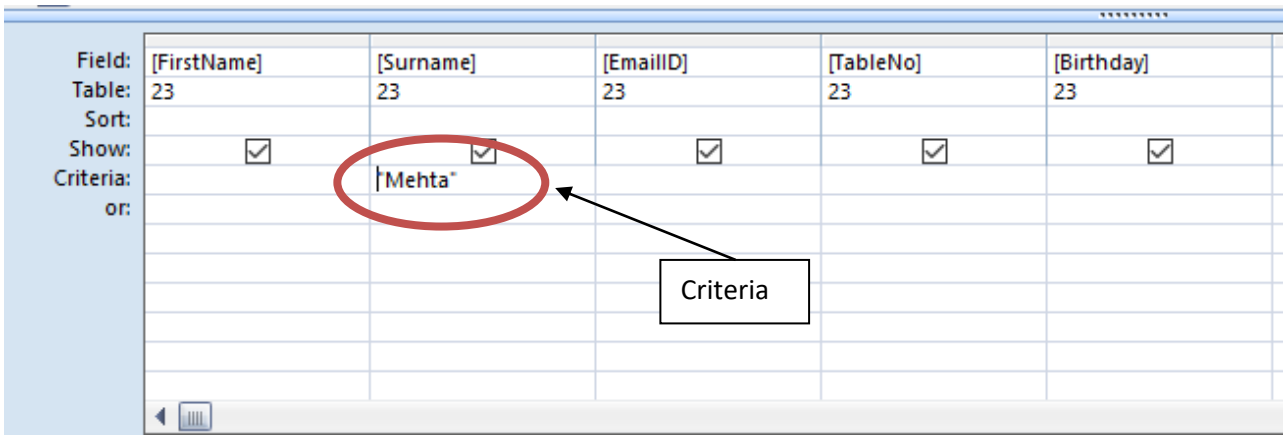

Figure 2: Creating Queries

A list of records with the surname Mehta will then be displayed.

23 | 23 Query1

FirstName	Surname	EmailID	TableNo	Birthday
Kimmico	Mehta	kim.mehta@gmail.com	1	Sun Apr 11 01:07:57 IST 1982
Shalaka	Mehta	shalypal@gmail.com	2	Thu Mar 05 22:02:45 IST 1998

Simple searching of records

Figure 3: Query Result

Microsoft Office Access is a flat file database which stores one record per line. Making it easier for my client to understand the inputted information as well as it is simple to add, search and delete records. Moreover, Access also helps in ensuring data integrity and data redundancy. Due to these reasons, I stored the customer's details in Microsoft Office Access.

b) To store customer's order

Code to connect software to second database

Connected using JDBC-ODBC Driver

```
try{
    String diver = "sun.jdbc.odbc.JdbcOdbcDriver";
    Class.forName(diver);
    String log = "jdbc:odbc:32";
    con = DriverManager.getConnection(log);
    System.out.print("connected also");
}catch(Exception ex) {}
```

Name of database

Once connection is established "connected also" is printed.

Exception handling

I created a second database to store the order information so that the staff is able to access the order. Also every time I switched the jframe the data fixed in the bill would get deleted. In order to avoid this I connected it to the database which would

display all the data of a consumer's order. These are the reasons why I created a second database in MS Access. Figure 4 shows the records of the order placed.

Primary key

ID	dish	cost	Add New Field
1	French Fries	150	
2	Eclair	280	
3	Greek Salad	280	
4	Espresso	325	
(New)			

Figure 4: Order database

2] Validation checks

In order to ensure quality input from the customer, I created validation checks.

Figure 5: Presence Check

To ensure that the customer enters details, I implemented presence validation checks into the program. As seen in figure 5, if the customer presses the continue button without entering data, a message will pop up and customer will not be able move forward. I did the same for the fields first name, email –id and table number, since they were of utmost importance.

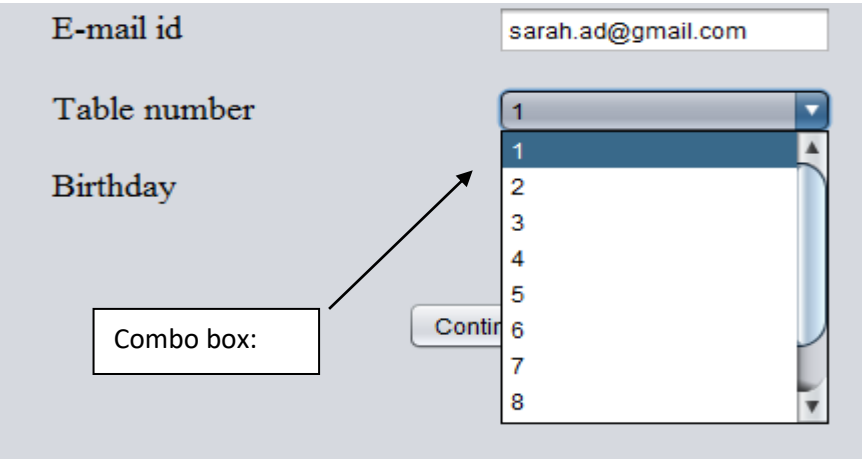

Figure 6: Range check

Table number is the most important field of data to be inputted by the customer because

without this, the waiting staff will not know which table to deliver to. In order to ensure correct data to be inputted by the customer, I created a combo box. This allow the customer to choose only from a certain set of values. Thereby not allowing abnormal data.

Moving onto the birthday field, customers could get confused in which date format to enter data. To prevent wrongful data entry I attached a calendar to the text field. This would help the customer to choose the date and it would automatically be set in the correct format as seen in Figure 7.

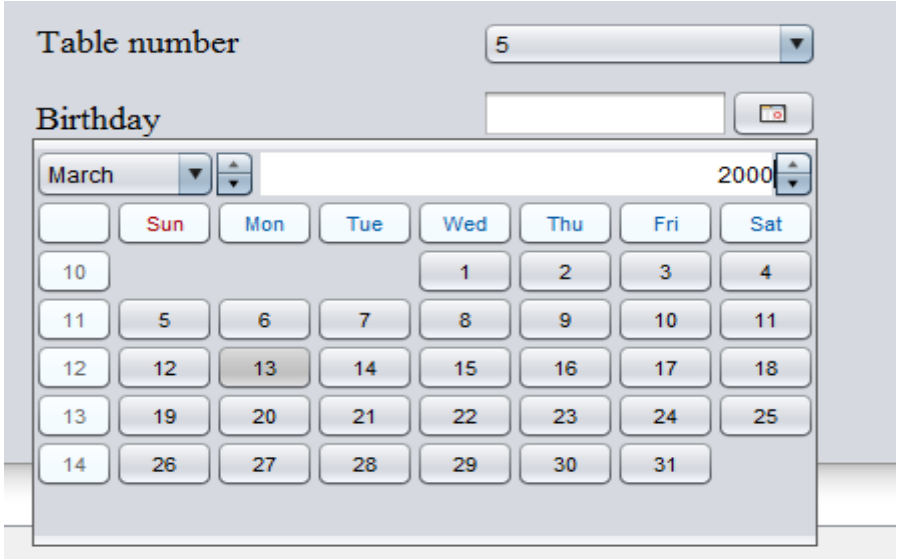

Figure 7: Calendar

3] SQL commands

In order to produce a bill in the Jtable, data needs to be fetched from the order MS Access database. To do this a method showdata() was created, which fetches the order data. This was the most adequate method since the data does not get deleted everytime the jFrame is changed. Instead the data keeps on getting added on and the entire order gets displayed in the jTable. The code of the method showdata() is shown below:

```java
private void showdata()
{
    try{
        String sql = "select Dish,Cost from 113";

        st = connection.prepareStatement(sql);
        rs = st.executeQuery(sql);
        jTable1.setModel(DbUtils.resultSetToTableModel(rs));
        jTable1.getTableHeader().setFont(new Font("Gadugi",Font.BOLD,24));

    }
    catch(Exception ex)
    {
        JOptionPane.showMessageDialog(null, ex.toString());
    }
}
```

Figure 8: fetching of data

4] Graphical User Interface

I created a graphical user interface throughout my software. This makes it easier to use especially for beginners. Some of the customers at the restaurant might not be good with technology, thus this sort of interface would be more beneficial. Also there is no need for the customer to learn confusing commands. Therefore, the software would have higher productivity and better accessibility. This is seen through the constant use of buttons in advanced technique 5.

5] Technique's to enable easy navigation

In order to transfer from one JFrame to another, I used buttons. These buttons make it more accessible for customers, since with a click they are presented with another page. Figure 9, shows the use of buttons in the main page.

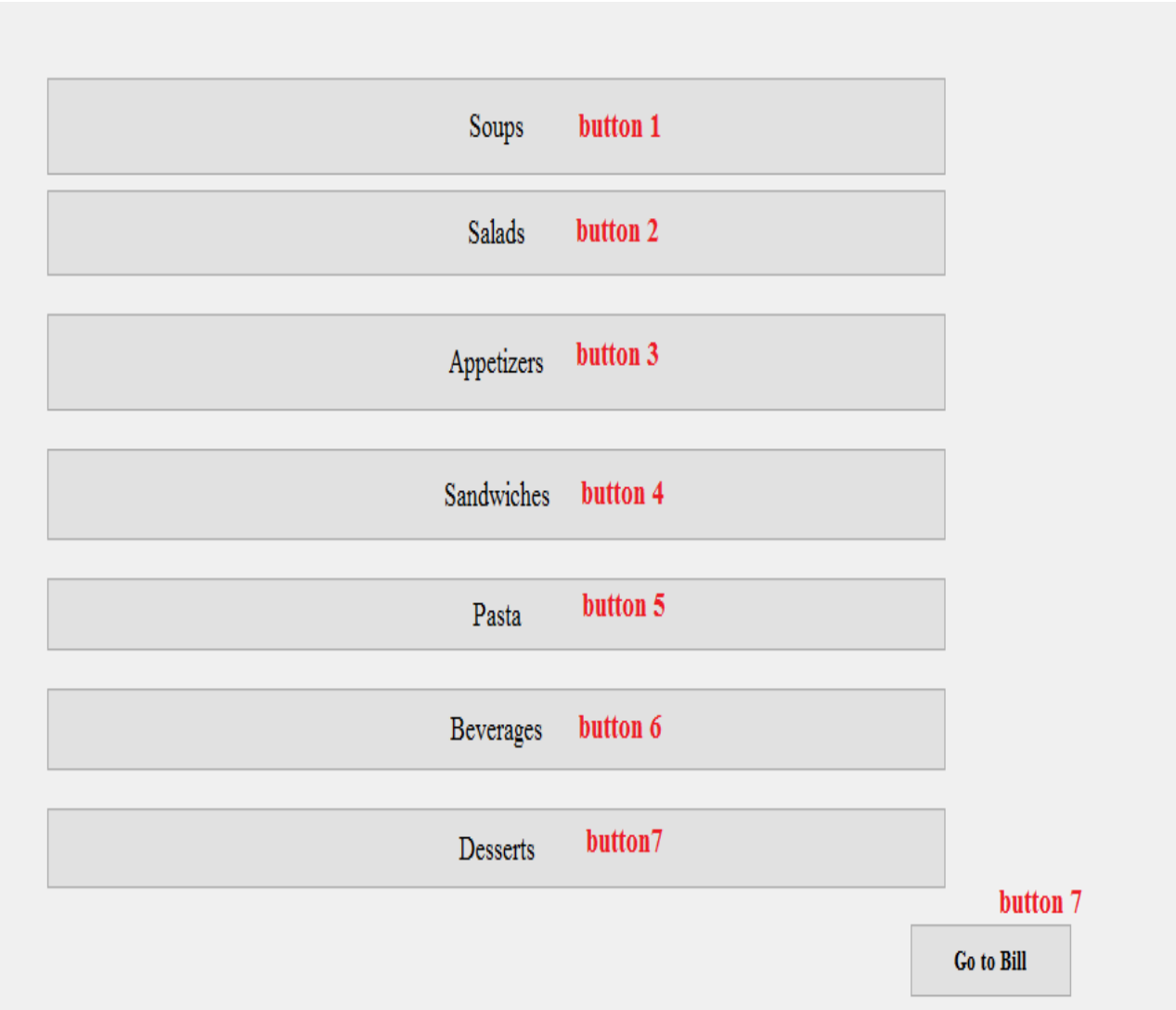

Figure 9: use of buttons

6] Object Definition

To use the buttons and transfer from one jframe to another, objects are created within the event-action performed for every button. Objects are constantly created in order to traverse through the software. This is seen in the code below:

```
188     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
189         // soups button
190         close();
191         soups s = new soups();
192         s.setVisible(true);
193     }
194
195     private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
196         // salads button
197         close();
198         salads a = new salads();
199         a.setVisible(true);
200     }
```

Object Creation

Figure 10: code of buttons

7] Polymorphism

I used the concept of polymorphism, in order to satisfy different circumstances when connecting Jframe's. There are two circumstances, when a customer pressed the "go to bill" button: (a) When no data is being transferred. (b) When name and cost have to be transferred. In order to do this I have created a second constructor with two parameters which accepts this data. Figure 11 shows these constructors:

```
25
26      public myorder() {
27          initComponents();
28          oorder.setEditable(false);
29          gugu.setEditable(false);
30      }
31      public myorder(String para, String c){
32          try{
33              String diver = "sun.jdbc.odbc.JdbcOdbcDriver";
34              Class.forName(diver);
35              String log = "jdbc:odbc:32";
36              con = DriverManager.getConnection(log);
37              System.out.print("connected also");
            }catch(Exception ex) {}
39          initComponents();
40          oorder.setEditable(false);
41          gugu.setEditable(false);
42          oorder.setText(para);
43          gugu.setText(c);
44      }
```

Constructor overloading

Figure 11: polymorphism

8] Integrating media

In order to provide maximum customer knowledge every dish in the software has its own Jframe consisting of various details including an image of the dish. One such Jframe can be seen in Figure 12:

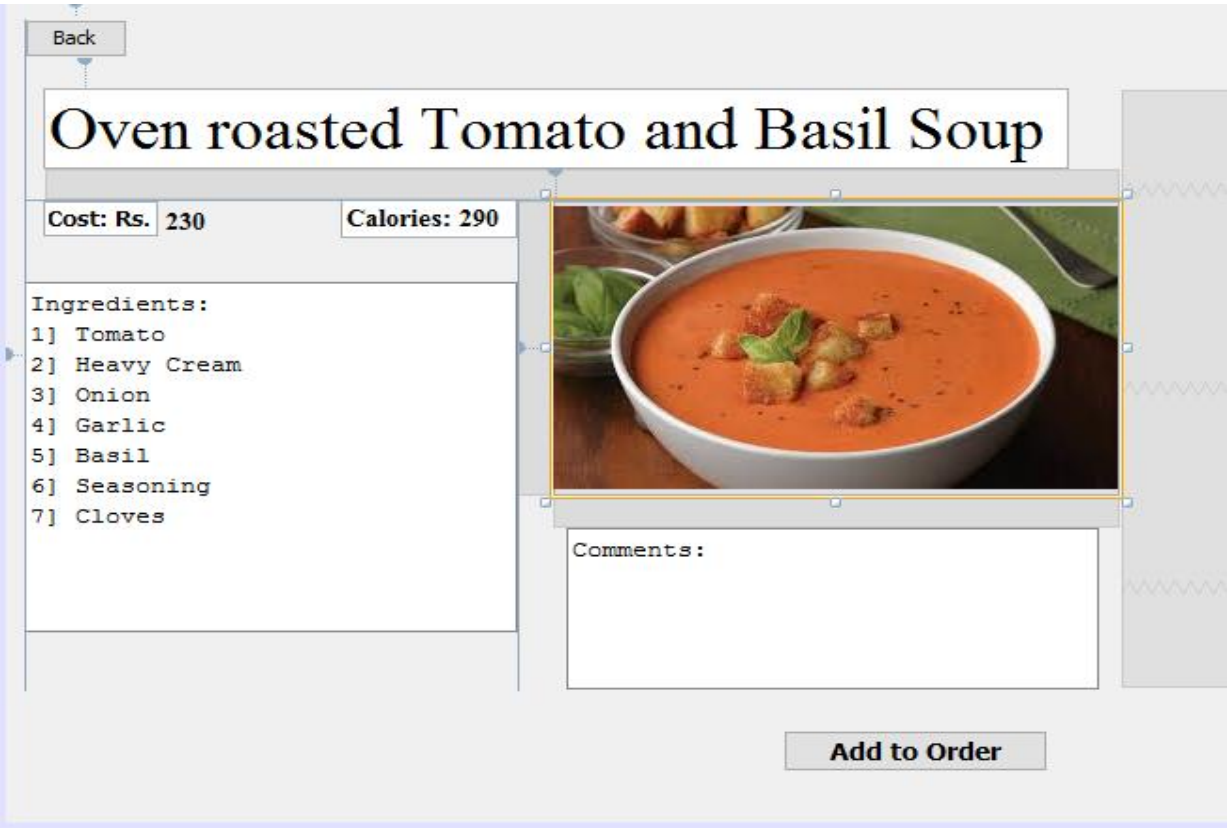

Figure 12: Integrating media

9] Subroutines

In order to set information in another class from data entered by the customer, I have used the accessor and mutator methods. Accessor methods getText() and getSelectedItem() are used and seen in Figure 13. This data is then transferred to another class through constructor overloading and is then displayed in a label through the mutator methods seen in Figure 14.

```
206        String d = Name.getText();
207        String e = TNO.getSelectedItem().toString();
```

Accessor Methods

Figure 13: Accessor Methods

This data gets transferred to another class through constructor overloading

```
22     public first(String d,String e)
23     {
24
25         String a = d;
26         initComponents();
27         yuyu.setText("HI! "+d);
28
29         mumu.setText("Table Number: "+e);
30
31
32
33     }
```

Mutator Methods

Figure 14: Mutator Methods

10] Exception handling

In order to maintain the normal flow during the execution of the program, I decided to use exception handling. The try blocks contains the code which might throw an exception which is followed by the catch block which handles the exception. This sort of exception handling is seen in Figure 15:

```
174        try{
175
176            ps = con.prepareStatement("Insert into 32(dish,cost)values(?,?)");
177            ps.setString(1, oorder.getText());
178            ps.setString(2, gugu.getText());
179
180            ps.executeQuery();
181        }
           catch(Exception ex) {}
183        JOptionPane.showMessageDialog(null, "Order Placed");
```

Figure 15: Exception Handling

11] Calculated fields

In order to calculate the total amount due while ordering, a calculation will need to take place in the bill table. This is done using the function getRowCount() and the condition i<rowsCount. The sum is calculated and displayed in the textfield of the bill class. The code of this calculation is shown in Figure 16:

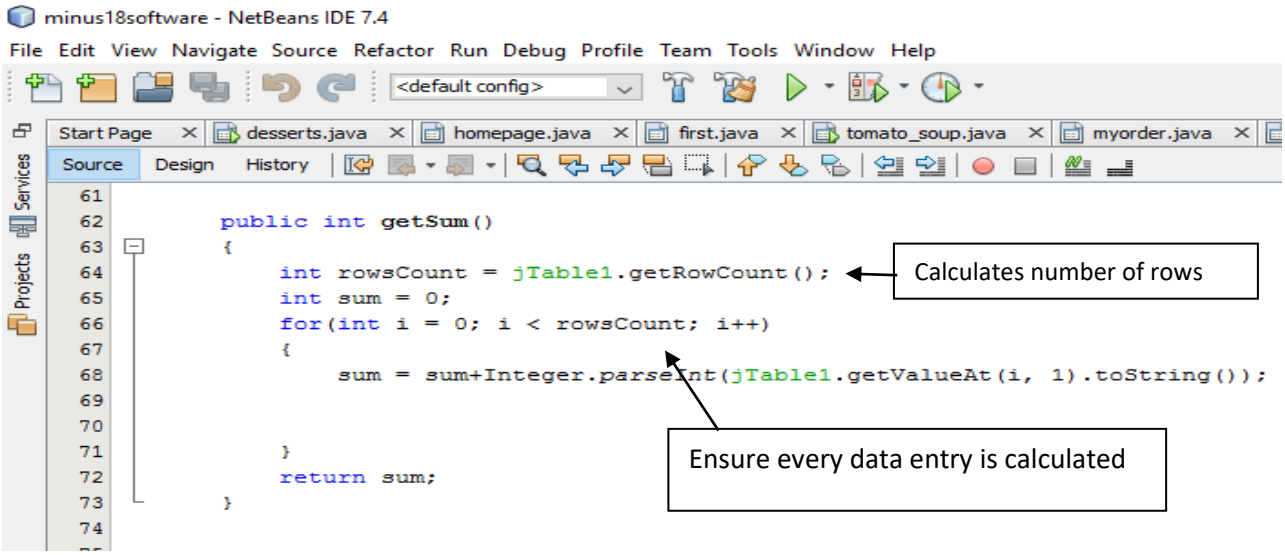

Figure 16: Sum calculation

(994 words)

Bibliography

1. https://www.youtube.com/watch?v=gM3y-sgGxkQ
2. https://www.youtube.com/watch?v=CEFzNs6UP4I
3. https://www.youtube.com/watch?v=QlrnavzizGw
4. https://www.youtube.com/watch?v=IiE3B5xkn2I&t=537s
5. https://www.youtube.com/watch?v=7GZppdccFfs&t=52s
6. https://www.youtube.com/watch?v=zvRTjIAFmXI
7. https://www.youtube.com/watch?v=r5S8TI5W2Q8
8. https://www.youtube.com/watch?v=uZFgiqM0udA
9. https://www.youtube.com/watch?v=5nfTfgQyXs8
10. https://www.youtube.com/watch?v=5m0zzr98k50