

KK14203( Object-oriented Programming)

Individual Project (Project II)

Lecturer

Dr Mohd. Shamrie Sainin

Prepared by:

|  |  |
| --- | --- |
| Name | Matric no |
| Nur Aisyah Nabilah binti Yusof | BI19110103 |

Due date is on

6th August 2020

1.0 Introduction

This individual project is given with the objective of building my very own system/application by using Graphical User Interface(GUI). The purpose of this project is to see the level or skill that I achieved in understanding the concept of Java programming. Other than that, it also helps to develop my knowledge in building a programming system/application from scratch. The application the I have created is a booking badminton court application. It displays the user name, phone number, booking hour and also the date of the week.

* 1. Java Code

1 import java.awt.\*;  
 2 import java.awt.event.\*;  
 3 import javax.swing.\*;  
 4 import javax.swing.event.\*;  
 5 //required for border  
 6 import javax.swing.BorderFactory;  
 7 import javax.swing.border.Border;  
 8 //required for file IO  
 9 import java.io.File;  
 10 import java.io.FileReader;  
 11 import java.io.FileWriter;  
 12 import java.io.PrintWriter;  
 13 import java.io.BufferedWriter;  
 14 import java.io.BufferedReader;  
 15 //required for exception  
 16 import java.io.IOException;  
 17   
 18 //Header panel  
 19 class HeaderPanel extends JPanel{  
 20 private JLabel header;  
 21 public HeaderPanel(){  
 22 header = new JLabel("Booking Badminton Court");  
 23 add(header);  
 24 }  
 25 }  
 26   
 27 //Form Panel  
 28 class FormPanel extends JPanel implements ActionListener, ItemListener{  
 29 //list all UI components for the panel  
 30 JLabel lbl\_name;  
 31 JTextField name;  
 32 JLabel lbl\_phone;  
 33 JTextField phone;  
 34 JLabel lbl\_court;  
 35 JComboBox<String> court;  
 36 JLabel lbl\_hours;  
 37 JLabel lbl\_outcome;  
 38 JButton b\_submit;  
 39 JButton b\_clear;  
 40 JButton addButton;   
 41 JCheckBox c1;  
 42 JCheckBox c2;  
 43 JCheckBox c3;  
 44 JLabel lbl\_output;   
 45 JScrollPane jsp;  
 46 Border border = BorderFactory.createLineBorder(Color.BLACK, 1);  
 47   
 48 //global variable   
 49 String output="";  
 50 String court\_selection="";  
 51 String rb\_selection="";  
 52 String cb\_selection="";  
 53 String filePath="data.txt"; //in the same directory  
 54   
 55 public FormPanel(){   
 56 setLayout(new FlowLayout(FlowLayout.LEFT));   
 57   
 58 lbl\_name = new JLabel("Name");  
 59 lbl\_name.setPreferredSize(new Dimension(200, 20));  
 60 lbl\_name.setBorder(border);  
 61   
 62 add(lbl\_name);  
 63 name = new JTextField(20);  
 64 add(name);  
 65   
 66 lbl\_phone = new JLabel("Phone No.");  
 67 lbl\_phone.setPreferredSize(new Dimension(200, 30));  
 68 add(lbl\_phone);  
 69 phone = new JTextField(11);  
 70 add(phone);   
 71   
 72 String[] Court={"[Select]", "court 1", "court 2", "court 3"};  
 73   
 74 lbl\_court = new JLabel("Court book");  
 75 lbl\_court.setPreferredSize(new Dimension(200, 20));  
 76 add(lbl\_court);  
 77 court = new JComboBox<String>(Court);  
 78 add(court);   
 79   
 80 //JComboBox action listener  
 81 court.addActionListener(new ActionListener(){  
 82 public void actionPerformed(ActionEvent ae){  
 83 //get selected item  
 84 court\_selection = (String) court.getSelectedItem();  
 85 }  
 86 });   
 87   
 88 lbl\_hours = new JLabel("Booking hour");  
 89 lbl\_hours.setPreferredSize(new Dimension(200, 30));  
 90 add(lbl\_hours);  
 91   
 92 //Radio buttons and action listener  
 93 JRadioButton rb1 = new JRadioButton("1");  
 94 rb1.addActionListener(this);  
 95 JRadioButton rb2 = new JRadioButton("2");  
 96 rb2.addActionListener(this);  
 97 JRadioButton rb3 = new JRadioButton("3");  
 98 rb3.addActionListener(this);  
 99 JRadioButton rb4 = new JRadioButton("4");  
100 rb4.addActionListener(this);  
101 add(rb1);  
102 add(rb2);  
103 add(rb3);  
104 add(rb4);  
105   
106   
107 //define button group  
108 ButtonGroup bg = new ButtonGroup();  
109 bg.add(rb1);  
110 bg.add(rb2);  
111 bg.add(rb3);  
112 bg.add(rb4);  
113   
114 lbl\_outcome = new JLabel("Status");  
115 lbl\_outcome.setPreferredSize(new Dimension(200, 20));  
116 add(lbl\_outcome);  
117   
118 //checkbox and item listener  
119 c1 = new JCheckBox("Student");  
120 c1.addItemListener(this);  
121 c2 = new JCheckBox("Staff");  
122 c2.addItemListener(this);  
123 c3 = new JCheckBox("Guest");  
124 c3.addItemListener(this);  
125 add(c1);  
126 add(c2);  
127 add(c3);  
128   
129 b\_submit = new JButton("Submit");  
130 add(b\_submit);  
131 b\_clear = new JButton("Clear");  
132 add(b\_clear);  
133 addButton = new JButton("Add");  
134 add(addButton);  
135   
136 //handle button submit action listener  
137 //view the input to output label  
138 //and write to file  
139 b\_submit.addActionListener(new ActionListener(){   
140 public void actionPerformed(ActionEvent e){   
141 //call method   
142 if(printOutput())  
143 writeInput();   
144 }   
145 });  
146   
147 //handle button clear action listener  
148 b\_clear.addActionListener(new ActionListener(){   
149 public void actionPerformed(ActionEvent e){   
150 lbl\_output.setText("Output");   
151 name.setText("");  
152 phone.setText("");   
153 court.setSelectedIndex(0);  
154 bg.clearSelection();  
155 c1.setSelected(false);  
156 c2.setSelected(false);  
157 c3.setSelected(false);  
158 }   
159 });  
160   
161 addButton.addActionListener(new ActionListener(){   
162 public void actionPerformed(ActionEvent e){  
163 name.setText("");  
164 phone.setText("");  
165   
166 }  
167 });  
168   
169 lbl\_output = new JLabel("Output");  
170 lbl\_output.setBorder(border);  
171 lbl\_output.setVerticalAlignment(JLabel.TOP);  
172   
173 //add output label to scrollpane  
174 jsp = new JScrollPane(lbl\_output);  
175 jsp.setPreferredSize(new Dimension(410,120));  
176 add(jsp);   
177 }  
178   
179 //handle radio button selection  
180 public void actionPerformed(ActionEvent ae) {  
181 rb\_selection = ae.getActionCommand();   
182 }  
183   
184 //handle item listener for checkbox  
185 public void itemStateChanged(ItemEvent ie) {  
186 JCheckBox check = (JCheckBox)ie.getSource();  
187 cb\_selection += check.getText() + " ";   
188 }  
189   
190 //method to print output to lbl\_output  
191 public boolean printOutput(){  
192 output = "<html>";  
193 output += "Thank you for your booking<br><br>";   
194 output += "Name: " + name.getText() + "<br>";  
195 output += "Phone no.: " + phone.getText() + "<br>";  
196 if(court\_selection.equals("[Select]") || court\_selection.equals("")){  
197 lbl\_output.setText("Court is not selected.");  
198 return false;  
199 }  
200 output += "Court: " + court\_selection + "<br>";  
201 output += "Hours: " + rb\_selection + " hour/s" + "<br>";  
202 output += "Outcome: " + cb\_selection + "<br>";  
203 output += "</html>";   
204 lbl\_output.setText(output);  
205 jsp.getViewport().revalidate();  
206 return true;  
207 }  
208   
209 //write to file  
210 public void writeInput(){  
211 File file = new File(filePath);  
212 FileWriter fr = null;  
213 BufferedWriter br = null;  
214 PrintWriter pr = null;  
215   
216 String input = name.getText() + ", " + phone.getText() + ", " + court\_selection + ", " + rb\_selection + ", " + cb\_selection;  
217   
218 //exception implementation  
219 try {  
220 // to append to file, you need to initialize FileWriter using below constructor  
221 fr = new FileWriter(file, true);  
222 br = new BufferedWriter(fr);  
223 pr = new PrintWriter(br);  
224 pr.println(input);  
225 } catch (IOException e) {   
226 lbl\_output.setText(e.toString());  
227 } finally {  
228 try {  
229 pr.close();  
230 br.close();  
231 fr.close();  
232 } catch (IOException e) {  
233 lbl\_output.setText(e.toString());  
234 }  
235 }  
236 }  
237 }  
238   
239 class MenuActionListener implements ActionListener {  
240 FormPanel fp;  
241 //receive FormPanel class to this constructor  
242 public MenuActionListener(FormPanel p){  
243 fp = p;  
244 }  
245   
246 public void actionPerformed(ActionEvent e) {  
247 BufferedReader reader;  
248 try {  
249 reader = new BufferedReader(new FileReader(fp.filePath));  
250 String line = reader.readLine();  
251 String output="<html>";  
252 while (line != null) {  
253 output += line + "<br>";  
254 // read next line  
255 line = reader.readLine();  
256 }  
257 output += "<br>";  
258 fp.lbl\_output.setText(output);  
259 reader.close();  
260 } catch (IOException io) {  
261 fp.lbl\_output.setText(io.toString());  
262 }  
263   
264 }  
265 }  
266   
267 //run the application using this main  
268 public class BookingBadminton {   
269 public static void main(String[] args) {   
270 JFrame f = new JFrame("Booking");  
271 f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
272 //load panels  
273 HeaderPanel h = new HeaderPanel();  
274 HeaderPanel h2 = new HeaderPanel();  
275 FormPanel fp = new FormPanel();  
276   
277 JMenuBar mb = new JMenuBar();   
278 // create a menu   
279 JMenu x = new JMenu("Menu");   
280   
281 // create menuitems   
282 JMenuItem m1 = new JMenuItem("View Data");   
283 // attach listener and send FormPanel class  
284 m1.addActionListener(new MenuActionListener(fp));  
285   
286 JMenuItem m2 = new JMenuItem("Exit");   
287 m2.addActionListener((event) -> System.exit(0));  
288 // add menu items to menu   
289 x.add(m1);   
290 x.add(m2);  
291   
292 // add menu to menu bar   
293 mb.add(x);   
294 // add menubar to frame   
295 f.setJMenuBar(mb);   
296   
297 //add panels to frame   
298 f.add(h,BorderLayout.NORTH);  
299 f.add(fp, BorderLayout.CENTER);  
300 f.setSize(460,400);  
301 f.setVisible(true);  
302 }   
303 }

* 1. Object Oriented Concept Implementation

1.2.1 Abstraction:

Abstraction aims to hide complexity from the users and show them only the relevant information.

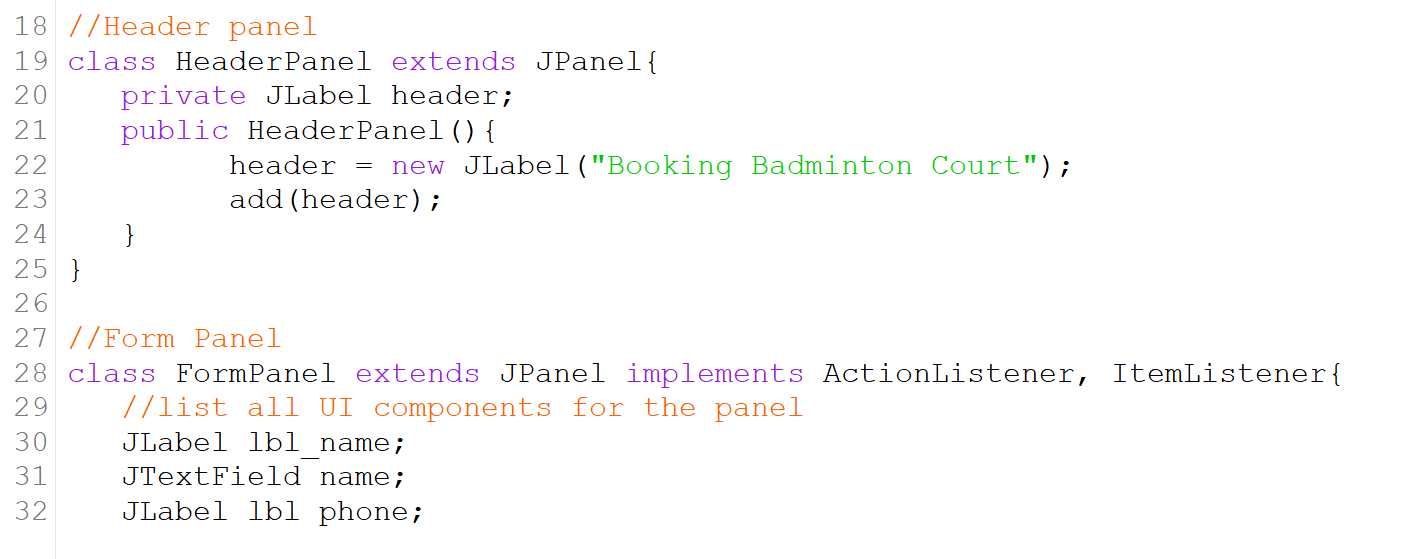


Figure 1.0

In Figure 1.0, shows multiple inheritance. Class HeadPanel extends from JPanel and class FormPanel extends JPanel that implements ActionListener. This shows that there are codes that can be reuse and flexibility of implementation. The result shows that the header exists in the panel.

1.2.2 Encapsulation

Encapsulation binds data and its related methods together within a class. It also protect the data by making fields private and giving access to them only through their related method.

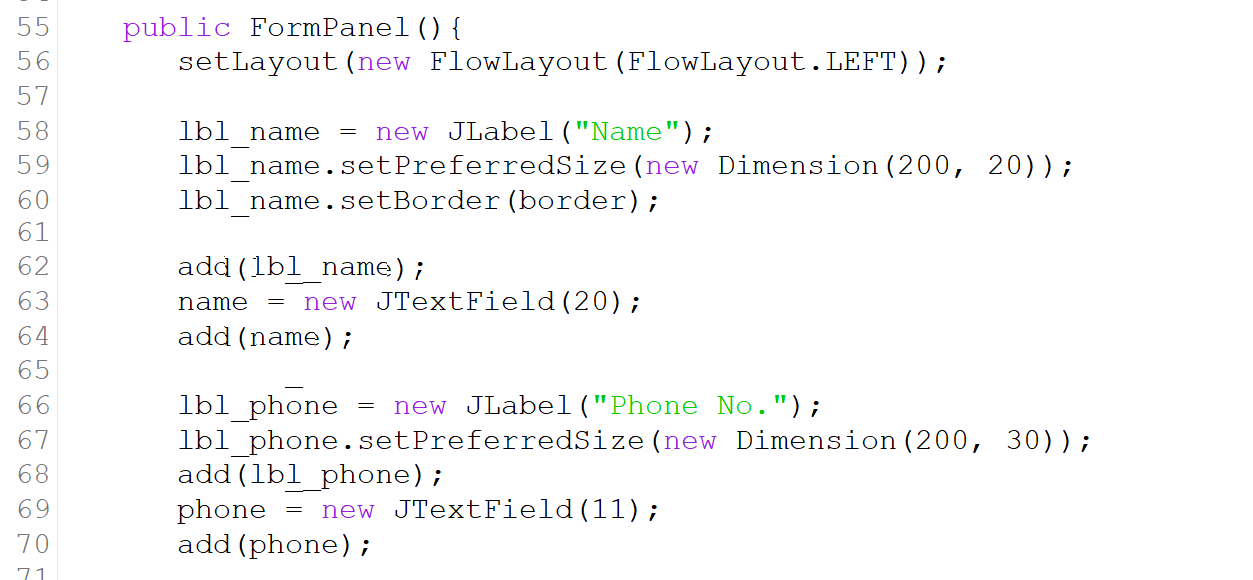


Figure 1.1

Based on the figure above, FormPanel is set to be public because it is needed for user input. As it shows in the figure, the lbl\_name and lbl\_phone has a setPreferredSize. It is function to set the size of the label for name and phone number. The variables in this method can be use by other methods because it is set to be public not private.

1.2.3 Inheritance

Inheritance allows a class(child class) to inherit the feature of another class.

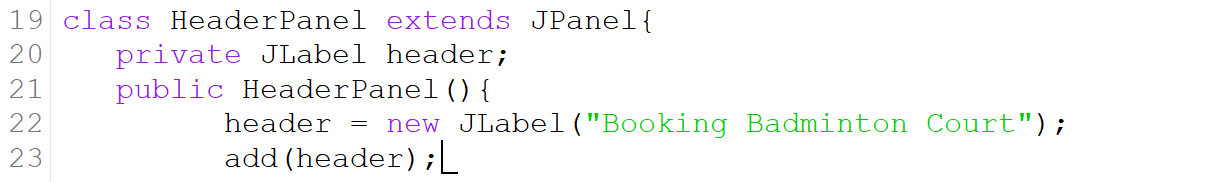


Figure 1.2

Figure above shows that the class HeaderPanel extends Jpanel. It means that HeaderPanel wants to inherit the feature that exists in Jpanel to be use in its very own method in its class.

1.2.4 Polymorphism

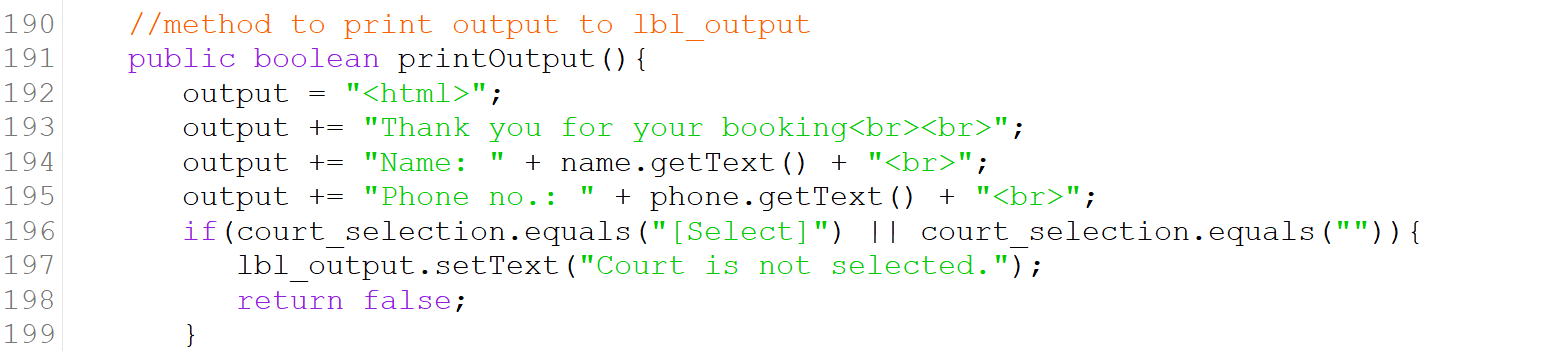


Figure 1.3

As shown in figure 1.3, method to print output is in boolean. In Java, classes can have various versions of the same method if their parameter structures are different. In this case, printOutput does not have any parameter. So there is not problem for this method to function because it did not take other methods’ parameter.

* 1. Read and Write Implementation
     1. Read and Write

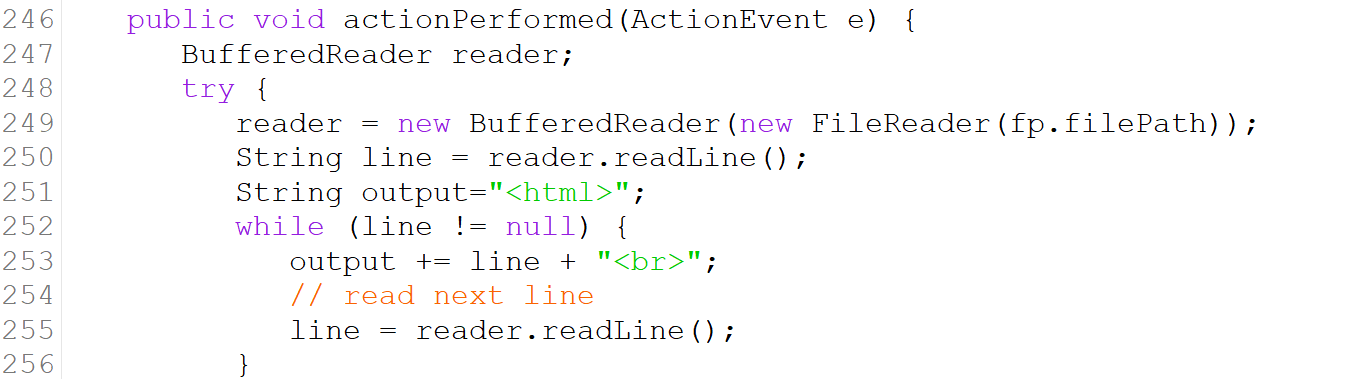


Figure 1.4

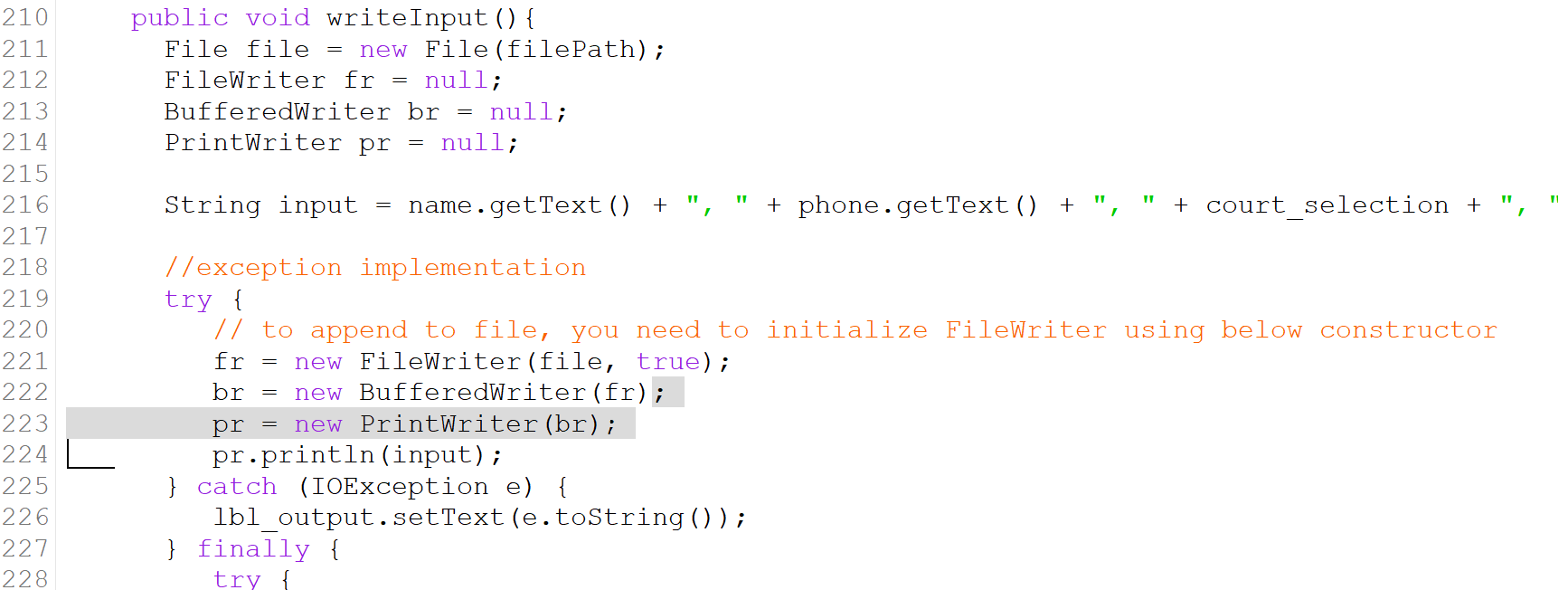
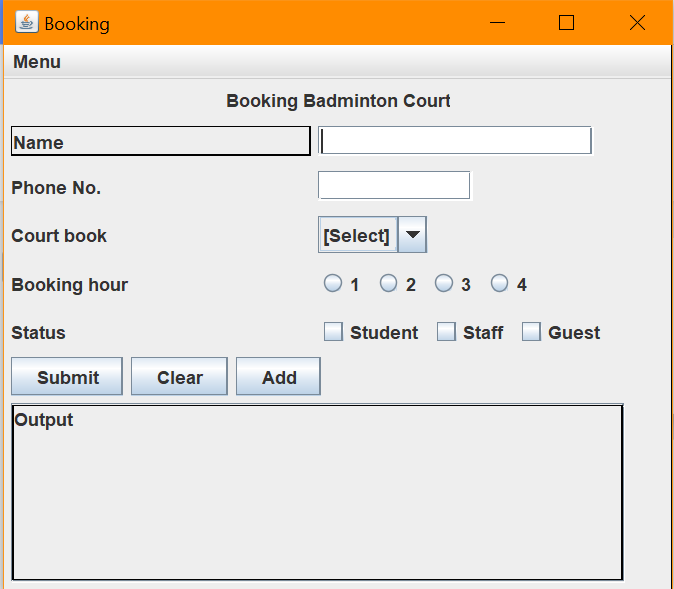


Figure 1.5

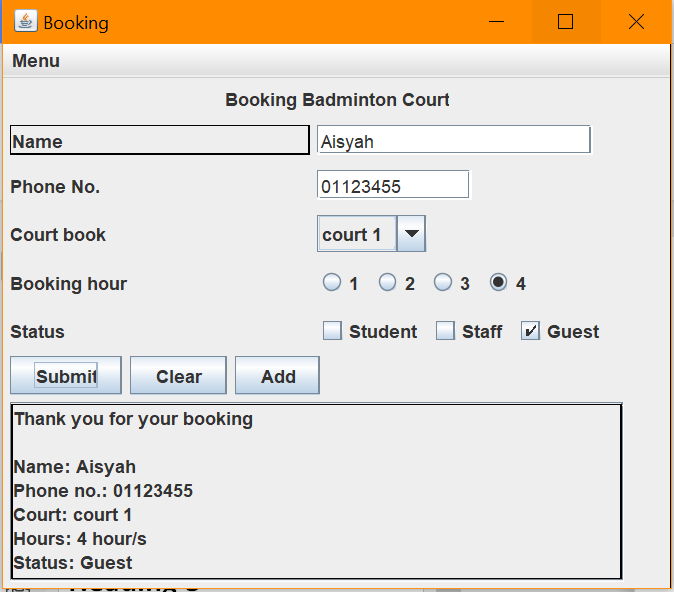
Read and write method is declared as abstract class. Read Methods cannot accept any parameter. As in this case, based on the figure 1.4, public void actionPerformed(ActionEvent e) is the read method. In Figure 1.5, public void writeInput() is the method. Data will be written in a file and the saved data are name, phone number and court selection.

* 1. User Manual

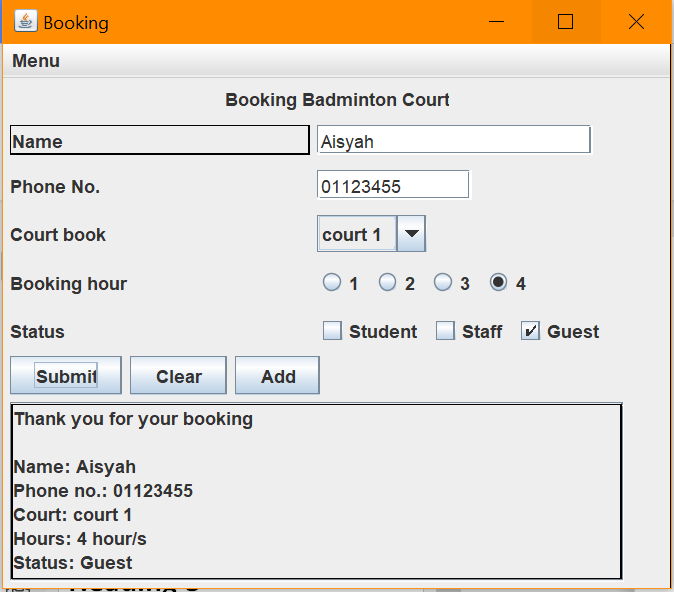
Firstly, user must filled the name and phone number. Select the court, and booking hour also provide user’s status when the application is pops up. As shown in figure below.



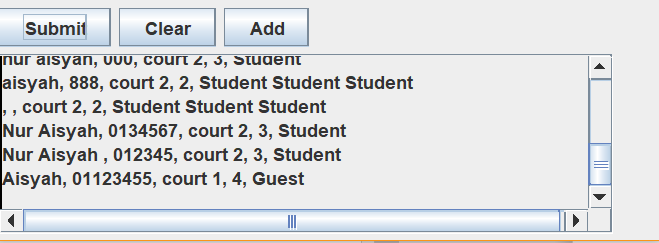
After all is complete, user must click the “Submit” button to confirm the booking. If the booking is successful, a message containing the user data will be shown in Output Texk box. As shown in the figure provided below.



Perhaps the user want to check wether or not the booking is saved. User can view the data by clicking the Menu button on the left corner of the application. Like so.

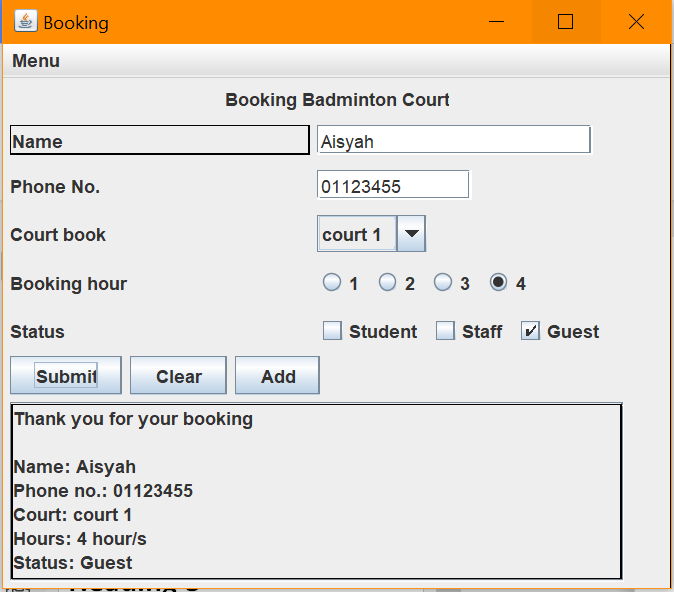


After that, user can see that there is a “view data” option. User can select “View Data” and the data will be display in the output Text box as shown in figure below.

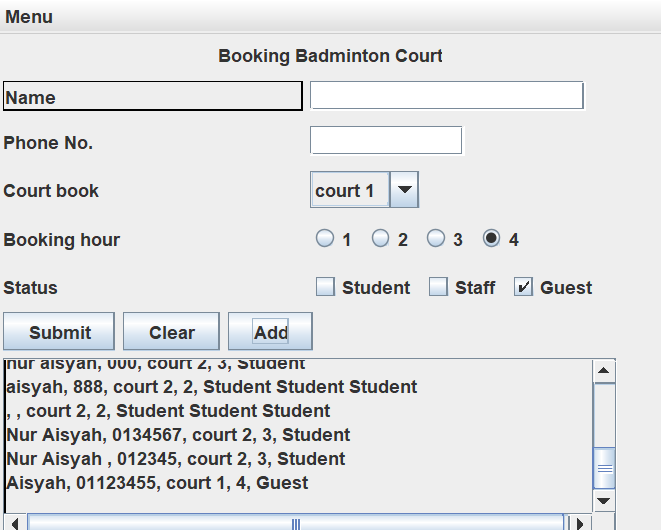


As it shows, the data is saved and it displays all the booking that was made earlier.

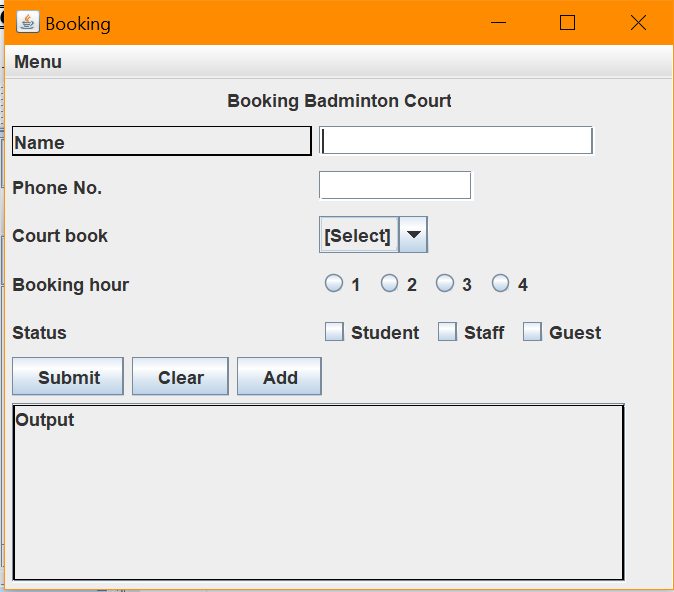
If the user wishes to add another booking, user may click the “Add” button.



The name and phone number will be empty again so that the user can enter another booking. As shown in figure below.



If user wants to clear or reset everything, user can click the “Clear” button. Everything will be reset and can be use to book another booking.



As it shows, everything has been reset for the user to make a new booking.

When everything is completed and the user wish to exit. User can go to the Menu which is on the left corner of the application and select “Exit”. Automatically the application will be closed.