ADPROC Coursework

# Introduction

Our program accepts the width, length, height and amount from the user, as well as lets the user select the grade of card, the colour print, whether the box will have reinforced bottom or corners and whether the box will have a sealable top. The program will the check if the details input match with a certain box type. If it does not, then an error is displayed and if it does then it shows the cost of the order. It also allows the user to make more than one order and will show him the total cost of all orders.

Assumptions that we made include;

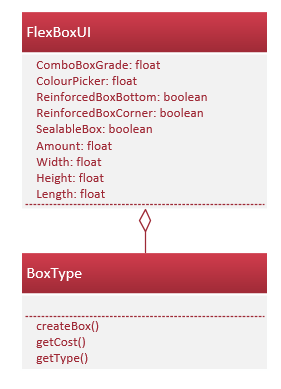
* That the user will always input the sizes (Width, Height, Length) in meters.

Limitations in our application include;

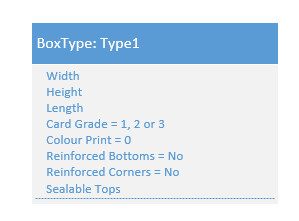
* The order and total cost only goes up to around 2 billion and then sets itself as a negative number.
* The order and total cost does not show value(Pound/Pence) and therefore the user must guess.
* The number is always rounded to the nearest whole number.

# UML Screenshots

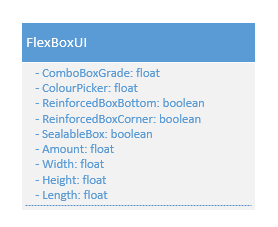
## Class Hierarchy Diagram



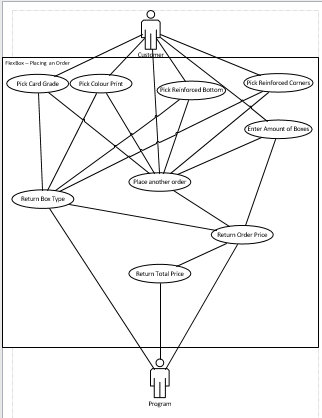
## Instance Diagram



## Single Class Diagram



## Use Case Diagram



# Source code

## FlexBoxUI

1. **package** flexbox;
3. /\*\*
4. \* @author danni
5. \* @author tom\_m
6. \*/
7. **public** **class** FlexBoxUI **extends** javax.swing.JFrame {
9. **int** total;
10. **int** result;
12. /\*\*
13. \* Creates new form FlexBoxUI
14. \*/
15. **public** FlexBoxUI() {
16. initComponents();
17. }
19. /\*\*
20. \* This method is called from within the constructor to initialise the form.
21. \* WARNING: Do NOT modify this code. The content of this method is always
22. \* regenerated by the Form Editor.
23. \*/
24. @SuppressWarnings("unchecked")
25. // <editor-fold defaultstate="collapsed" desc="Generated Code">
26. **private** **void** initComponents() {
28. jPanel1 = **new** javax.swing.JPanel();
29. FlexboxTitle = **new** javax.swing.JLabel();
30. FlexboxComboBoxCardGrades = **new** javax.swing.JComboBox<>();
31. Title2 = **new** javax.swing.JLabel();
32. GradeOfCard = **new** javax.swing.JLabel();
33. WidthTextbox = **new** javax.swing.JTextField();
34. Width = **new** javax.swing.JLabel();
35. Length = **new** javax.swing.JLabel();
36. LengthTextbox = **new** javax.swing.JTextField();
37. Height = **new** javax.swing.JLabel();
38. HeightTextbox = **new** javax.swing.JTextField();
39. FindCost = **new** javax.swing.JButton();
40. ColourPrint = **new** javax.swing.JLabel();
41. ColourBox = **new** javax.swing.JComboBox<>();
42. ReinforcedBottom = **new** javax.swing.JLabel();
43. ReinforcedB = **new** javax.swing.JComboBox<>();
44. ReinforcedCorner = **new** javax.swing.JLabel();
45. ReinforcedC = **new** javax.swing.JComboBox<>();
46. SealableTops = **new** javax.swing.JLabel();
47. SealableTopsBox = **new** javax.swing.JComboBox<>();
48. Amount = **new** javax.swing.JLabel();
49. AmountTextbox = **new** javax.swing.JTextField();
50. TotalCost = **new** javax.swing.JLabel();
51. jButton1 = **new** javax.swing.JButton();
52. jLabel1 = **new** javax.swing.JLabel();
53. jScrollPane2 = **new** javax.swing.JScrollPane();
54. Result1 = **new** javax.swing.JTextArea();
55. jScrollPane3 = **new** javax.swing.JScrollPane();
56. Result2 = **new** javax.swing.JTextArea();
57. jLabel2 = **new** javax.swing.JLabel();
59. setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);
61. jPanel1.setBackground(**new** java.awt.Color(170, 185, 237));
62. jPanel1.setCursor(**new** java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));
64. FlexboxTitle.setFont(**new** java.awt.Font("Times New Roman", 0, 24)); // NOI18N
65. FlexboxTitle.setText("Flexbox");
67. FlexboxComboBoxCardGrades.setFont(**new** java.awt.Font("Times New Roman", 0, 11)); // NOI18N
68. FlexboxComboBoxCardGrades.setModel(**new** javax.swing.DefaultComboBoxModel<>(**new** String[] { "1", "2", "3", "4", "5" }));
70. Title2.setFont(**new** java.awt.Font("Times New Roman", 0, 18)); // NOI18N
71. Title2.setText("Please specify the following details for your order:");
73. GradeOfCard.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
74. GradeOfCard.setText("Grade of card:");
76. WidthTextbox.setFont(**new** java.awt.Font("Times New Roman", 0, 11)); // NOI18N
77. WidthTextbox.addActionListener(**new** java.awt.event.ActionListener() {
78. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
79. WidthTextboxActionPerformed(evt);
80. }
81. });
83. Width.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
84. Width.setText("Width:");
86. Length.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
87. Length.setText("Length:");
89. LengthTextbox.setFont(**new** java.awt.Font("Times New Roman", 0, 11)); // NOI18N
91. Height.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
92. Height.setText("Height:");
94. HeightTextbox.setFont(**new** java.awt.Font("Times New Roman", 0, 11)); // NOI18N
96. FindCost.setText("Find Cost");
97. FindCost.addActionListener(**new** java.awt.event.ActionListener() {
98. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
99. FindCostActionPerformed(evt);
100. }
101. });
103. ColourPrint.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
104. ColourPrint.setText("Colour Print:");
106. ColourBox.setFont(**new** java.awt.Font("Times New Roman", 0, 13)); // NOI18N
107. ColourBox.setModel(**new** javax.swing.DefaultComboBoxModel<>(**new** String[] { "0", "1", "2" }));
108. ColourBox.addActionListener(**new** java.awt.event.ActionListener() {
109. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
110. ColourBoxActionPerformed(evt);
111. }
112. });
114. ReinforcedBottom.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
115. ReinforcedBottom.setText("Reinforced Bottom");
117. ReinforcedB.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
118. ReinforcedB.setModel(**new** javax.swing.DefaultComboBoxModel<>(**new** String[] { "No", "Yes" }));
119. ReinforcedB.addActionListener(**new** java.awt.event.ActionListener() {
120. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
121. ReinforcedBActionPerformed(evt);
122. }
123. });
125. ReinforcedCorner.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
126. ReinforcedCorner.setText("Reinforced Corner");
128. ReinforcedC.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
129. ReinforcedC.setModel(**new** javax.swing.DefaultComboBoxModel<>(**new** String[] { "No", "Yes" }));
130. ReinforcedC.addActionListener(**new** java.awt.event.ActionListener() {
131. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
132. ReinforcedCActionPerformed(evt);
133. }
134. });
136. SealableTops.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
137. SealableTops.setText("Sealable tops");
139. SealableTopsBox.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
140. SealableTopsBox.setModel(**new** javax.swing.DefaultComboBoxModel<>(**new** String[] { "No", "Yes" }));
141. SealableTopsBox.addActionListener(**new** java.awt.event.ActionListener() {
142. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
143. SealableTopsBoxActionPerformed(evt);
144. }
145. });
147. Amount.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
148. Amount.setText("Amount ");
150. AmountTextbox.setFont(**new** java.awt.Font("Times New Roman", 0, 11)); // NOI18N
152. TotalCost.setFont(**new** java.awt.Font("Times New Roman", 0, 24)); // NOI18N
153. TotalCost.setText("Total Cost:");
155. jButton1.setText("Add another order");
156. jButton1.addActionListener(**new** java.awt.event.ActionListener() {
157. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
158. jButton1ActionPerformed(evt);
159. }
160. });
162. jLabel1.setFont(**new** java.awt.Font("Times New Roman", 0, 24)); // NOI18N
163. jLabel1.setText("Cost of order:");
165. Result1.setEditable(**false**);
166. Result1.setBackground(**new** java.awt.Color(170, 185, 237));
167. Result1.setColumns(20);
168. Result1.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
169. Result1.setLineWrap(**true**);
170. Result1.setRows(5);
171. jScrollPane2.setViewportView(Result1);
173. Result2.setEditable(**false**);
174. Result2.setBackground(**new** java.awt.Color(170, 185, 237));
175. Result2.setColumns(20);
176. Result2.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
177. Result2.setLineWrap(**true**);
178. Result2.setRows(5);
179. jScrollPane3.setViewportView(Result2);
181. jLabel2.setFont(**new** java.awt.Font("Times New Roman", 0, 14)); // NOI18N
182. jLabel2.setText("Automatically adds to ->");
184. javax.swing.GroupLayout jPanel1Layout = **new** javax.swing.GroupLayout(jPanel1);
185. jPanel1.setLayout(jPanel1Layout);
186. jPanel1Layout.setHorizontalGroup(
187. jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
188. .addGroup(jPanel1Layout.createSequentialGroup()
189. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
190. .addGroup(jPanel1Layout.createSequentialGroup()
191. .addContainerGap()
192. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
193. .addGroup(jPanel1Layout.createSequentialGroup()
194. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
195. .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
196. .addGroup(jPanel1Layout.createSequentialGroup()
197. .addGap(99, 99, 99)
198. .addComponent(FindCost)))
199. .addGap(84, 84, 84)
200. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
201. .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
202. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()
203. .addComponent(Height)
204. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
205. .addComponent(HeightTextbox, javax.swing.GroupLayout.PREFERRED\_SIZE, 72, javax.swing.GroupLayout.PREFERRED\_SIZE)
206. .addGap(33, 33, 33))))
207. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, **false**)
208. .addGroup(jPanel1Layout.createSequentialGroup()
209. .addGap(37, 37, 37)
210. .addComponent(jLabel1)
211. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)
212. .addComponent(TotalCost))
213. .addGroup(javax.swing.GroupLayout.Alignment.LEADING, jPanel1Layout.createSequentialGroup()
214. .addGap(334, 334, 334)
215. .addComponent(jButton1)))
216. .addGroup(jPanel1Layout.createSequentialGroup()
217. .addGap(14, 14, 14)
218. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
219. .addGroup(jPanel1Layout.createSequentialGroup()
220. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
221. .addGroup(jPanel1Layout.createSequentialGroup()
222. .addGap(28, 28, 28)
223. .addComponent(Width)
224. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
225. .addComponent(WidthTextbox, javax.swing.GroupLayout.PREFERRED\_SIZE, 72, javax.swing.GroupLayout.PREFERRED\_SIZE)
226. .addGap(39, 39, 39)
227. .addComponent(Length)
228. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
229. .addComponent(LengthTextbox, javax.swing.GroupLayout.PREFERRED\_SIZE, 72, javax.swing.GroupLayout.PREFERRED\_SIZE))
230. .addGroup(jPanel1Layout.createSequentialGroup()
231. .addComponent(SealableTops)
232. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
233. .addComponent(SealableTopsBox, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
234. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
235. .addComponent(ReinforcedCorner)
236. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
237. .addComponent(ReinforcedC, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))
238. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
239. .addComponent(Amount)
240. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
241. .addComponent(AmountTextbox, javax.swing.GroupLayout.PREFERRED\_SIZE, 49, javax.swing.GroupLayout.PREFERRED\_SIZE))
242. .addGroup(jPanel1Layout.createSequentialGroup()
243. .addComponent(GradeOfCard)
244. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
245. .addComponent(FlexboxComboBoxCardGrades, javax.swing.GroupLayout.PREFERRED\_SIZE, 43, javax.swing.GroupLayout.PREFERRED\_SIZE)
246. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
247. .addComponent(ColourPrint)
248. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
249. .addComponent(ColourBox, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
250. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
251. .addComponent(ReinforcedBottom)
252. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
253. .addComponent(ReinforcedB, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))
254. .addGroup(jPanel1Layout.createSequentialGroup()
255. .addGap(182, 182, 182)
256. .addComponent(jLabel2))))))
257. .addGroup(jPanel1Layout.createSequentialGroup()
258. .addGap(221, 221, 221)
259. .addComponent(FlexboxTitle))
260. .addGroup(jPanel1Layout.createSequentialGroup()
261. .addGap(90, 90, 90)
262. .addComponent(Title2)))
263. .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))
264. );
265. jPanel1Layout.setVerticalGroup(
266. jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
267. .addGroup(jPanel1Layout.createSequentialGroup()
268. .addContainerGap()
269. .addComponent(FlexboxTitle)
270. .addGap(18, 18, 18)
271. .addComponent(Title2)
272. .addGap(33, 33, 33)
273. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
274. .addComponent(GradeOfCard)
275. .addComponent(ColourPrint)
276. .addComponent(FlexboxComboBoxCardGrades, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
277. .addComponent(ColourBox, javax.swing.GroupLayout.PREFERRED\_SIZE, 19, javax.swing.GroupLayout.PREFERRED\_SIZE)
278. .addComponent(ReinforcedBottom)
279. .addComponent(ReinforcedB, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))
280. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
281. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
282. .addComponent(ReinforcedCorner)
283. .addComponent(ReinforcedC, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
284. .addComponent(SealableTops)
285. .addComponent(SealableTopsBox, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
286. .addComponent(Amount)
287. .addComponent(AmountTextbox, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))
288. .addGap(51, 51, 51)
289. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
290. .addComponent(Width)
291. .addComponent(WidthTextbox, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
292. .addComponent(Length)
293. .addComponent(LengthTextbox, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
294. .addComponent(Height)
295. .addComponent(HeightTextbox, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))
296. .addGap(35, 35, 35)
297. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
298. .addComponent(FindCost)
299. .addComponent(jButton1))
300. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
301. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
302. .addComponent(jLabel1)
303. .addComponent(TotalCost)
304. .addComponent(jLabel2))
305. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 26, Short.MAX\_VALUE)
306. .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
307. .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
308. .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))
309. .addContainerGap())
310. );
312. Title2.getAccessibleContext().setAccessibleName("");
314. javax.swing.GroupLayout layout = **new** javax.swing.GroupLayout(getContentPane());
315. getContentPane().setLayout(layout);
316. layout.setHorizontalGroup(
317. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
318. .addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)
319. );
320. layout.setVerticalGroup(
321. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
322. .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)
323. );
325. pack();
326. }// </editor-fold>
328. **private** **void** WidthTextboxActionPerformed(java.awt.event.ActionEvent evt) {
329. // TODO add your handling code here:
330. }
332. **private** **void** FindCostActionPerformed(java.awt.event.ActionEvent evt) {
333. String ComboBoxGrade = (String) FlexboxComboBoxCardGrades.getSelectedItem();
334. String ColourPicker = (String) ColourBox.getSelectedItem();
335. String ReinforcedBoxBottom = (String) ReinforcedB.getSelectedItem();
336. String ReinforcedBoxCorner = (String) ReinforcedC.getSelectedItem();
337. String SealableBox = (String) SealableTopsBox.getSelectedItem();
339. **try** {
340. **float** height, length, width, Area;
341. height = Float.parseFloat(HeightTextbox.getText());
342. length = Float.parseFloat(LengthTextbox.getText());
343. width = Float.parseFloat(WidthTextbox.getText());
344. Area = ((2 \* (height \* length)) + (2 \* (length \* width)) + (2 \* (height \* width)));
345. Result1.setText(String.valueOf(Area));
347. BoxType box = **new** BoxType(Area, ComboBoxGrade, ColourPicker, ReinforcedBoxBottom, ReinforcedBoxCorner, SealableBox);
348. box.createBox();
349. result = (**int**) box.getCost();
350. **int** boxType = box.getType();
352. **float** amount;
353. amount = Float.parseFloat(AmountTextbox.getText());
354. result = (**int**) (result \* amount);
356. **if** (boxType >= 1 && boxType <= 5) {
357. total = total + result;
358. Result1.setText(String.valueOf((result)));
359. Result2.setText(String.valueOf(total));
360. } **else** {
361. Result1.setText(String.valueOf("Invalid Box Type, Please change order"));
362. }
363. } **catch** (NumberFormatException e) {
364. Result1.setText(String.valueOf("Please insert a number and don't keep the textbox empty."));
365. }
366. }
368. **private** **void** ReinforcedBActionPerformed(java.awt.event.ActionEvent evt) {
369. // TODO add your handling code here:
370. }
372. **private** **void** ReinforcedCActionPerformed(java.awt.event.ActionEvent evt) {
373. // TODO add your handling code here:
374. }
376. **private** **void** SealableTopsBoxActionPerformed(java.awt.event.ActionEvent evt) {
377. // TODO add your handling code here:
378. }
380. **private** **void** ColourBoxActionPerformed(java.awt.event.ActionEvent evt) {
381. // TODO add your handling code here:
382. }
384. /\*\*
385. \* @param args
386. \* Runs the program. Initialises the gui.
387. \*/
388. **public** **static** **void** main(String[] args) {
389. FlexBoxUI gui = **new** FlexBoxUI();
390. **new** FlexBoxUI().setVisible(**true**);
391. }
393. **private** **void** jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
395. result = (**int**) Float.parseFloat(Result1.getText());
396. Result2.setText(String.valueOf((total)));
398. WidthTextbox.setText("");
399. LengthTextbox.setText("");
400. HeightTextbox.setText("");
401. AmountTextbox.setText("");
402. FlexboxComboBoxCardGrades.setSelectedIndex(0);
403. ColourBox.setSelectedIndex(0);
404. ReinforcedB.setSelectedIndex(0);
405. ReinforcedC.setSelectedIndex(0);
406. SealableTopsBox.setSelectedIndex(0);
407. Result1.setText("");
408. }
409. // Variables declaration - do not modify
410. **private** javax.swing.JLabel Amount;
411. **private** javax.swing.JTextField AmountTextbox;
412. **private** javax.swing.JComboBox<String> ColourBox;
413. **private** javax.swing.JLabel ColourPrint;
414. **private** javax.swing.JButton FindCost;
415. **private** javax.swing.JComboBox<String> FlexboxComboBoxCardGrades;
416. **private** javax.swing.JLabel FlexboxTitle;
417. **private** javax.swing.JLabel GradeOfCard;
418. **private** javax.swing.JLabel Height;
419. **private** javax.swing.JTextField HeightTextbox;
420. **private** javax.swing.JLabel Length;
421. **private** javax.swing.JTextField LengthTextbox;
422. **private** javax.swing.JComboBox<String> ReinforcedB;
423. **private** javax.swing.JLabel ReinforcedBottom;
424. **private** javax.swing.JComboBox<String> ReinforcedC;
425. **private** javax.swing.JLabel ReinforcedCorner;
426. **private** javax.swing.JTextArea Result1;
427. **private** javax.swing.JTextArea Result2;
428. **private** javax.swing.JLabel SealableTops;
429. **private** javax.swing.JComboBox<String> SealableTopsBox;
430. **private** javax.swing.JLabel Title2;
431. **private** javax.swing.JLabel TotalCost;
432. **private** javax.swing.JLabel Width;
433. **private** javax.swing.JTextField WidthTextbox;
434. **private** javax.swing.JButton jButton1;
435. **private** javax.swing.JLabel jLabel1;
436. **private** javax.swing.JLabel jLabel2;
437. **private** javax.swing.JPanel jPanel1;
438. **private** javax.swing.JScrollPane jScrollPane2;
439. **private** javax.swing.JScrollPane jScrollPane3;
440. // End of variables declaration
441. }

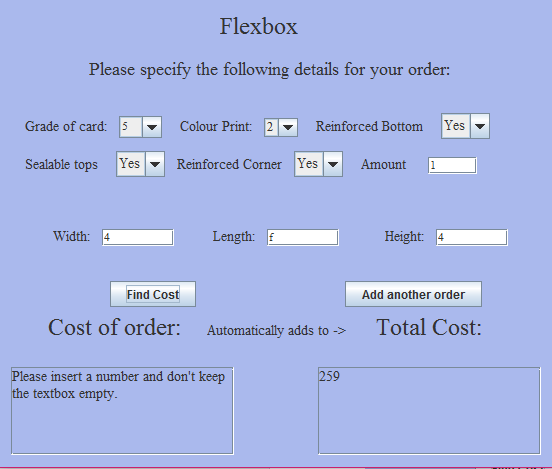
## BoxType

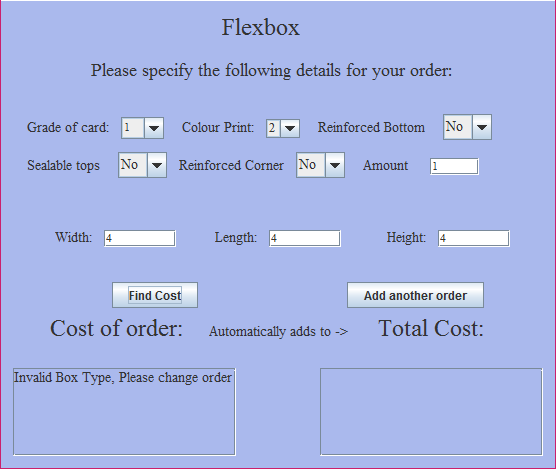
1. **package** flexbox;
3. /\*\*
4. \* @author tom\_m
5. \* @author danni
6. \*/
7. **public** **class** BoxType {
9. **private** **double** cost = 0.0;
10. **private** String ComboBoxGrade = "0";
11. **private** **float** area = (**float**) 0.0;
12. **private** String ColourPicker = "0";
13. **private** **int** boxType = 0;
14. **private** String ReinforcedBoxBottom = "No";
15. **private** String ReinforcedBoxCorner = "No";
16. **private** String SealableBox = "No";
18. /\*\*
19. \* Constructor.
20. \*/
21. **public** BoxType() {
22. area = (**float**) 10.0;
23. ComboBoxGrade = "1";
25. }
27. /\*\*
28. \* Initialises the parameters
29. \*
30. \* @param area
31. \* @param cmg
32. \* @param cp
33. \* @param rbb
34. \* @param rbc
35. \* @param sb
36. \*/
37. **public** BoxType(**float** area, String cmg, String cp, String rbb, String rbc, String sb) {
38. **this**.area = area;
39. **this**.ComboBoxGrade = cmg;
40. **this**.ColourPicker = cp;
41. **this**.ReinforcedBoxBottom = rbb;
42. **this**.ReinforcedBoxCorner = rbc;
43. **this**.SealableBox = sb;
44. }
46. /\*\*
47. \* Method to calculate the boxType
48. \*/
49. **public** **void** createBox() {
50. **switch** (ComboBoxGrade) {
51. **case** "1":
52. cost = (area \* 0.5);
53. **break**;
54. **case** "2":
55. cost = (area \* 0.6);
56. **break**;
57. **case** "3":
58. cost = (area \* 0.72);
59. **break**;
60. **case** "4":
61. cost = (area \* 0.9);
62. **break**;
63. **case** "5":
64. cost = (area \* 1.4);
65. **break**;
66. }
68. **switch** (ColourPicker) {
69. **case** "0":
70. **if** ("1".equals(ComboBoxGrade) || "2".equals(ComboBoxGrade) || "3".equals(ComboBoxGrade)) {
71. boxType = 1;
72. } **else** {
73. boxType = 6; //Setting it to 6 returns an error.
74. }
75. **break**;
76. **case** "1":
77. **if** ("2".equals(ComboBoxGrade) || "3".equals(ComboBoxGrade) || "4".equals(ComboBoxGrade)) {
78. boxType = 2;
79. } **else** {
80. boxType = 6;
81. }
82. cost = (cost \* 1.13);
83. **break**;
84. **case** "2":
85. **if** ("2".equals(ComboBoxGrade)) {
86. boxType = 3;
87. }
88. **if** ("3".equals(ComboBoxGrade) || "4".equals(ComboBoxGrade) || "5".equals(ComboBoxGrade)) {
89. boxType = 5;
90. }
91. cost = (cost \* 1.16);
92. **break**;
93. }
95. **switch** (ReinforcedBoxBottom) {
96. **case** "Yes":
97. **if** (boxType == 1) {
98. boxType = 6;
99. }
100. **if** (boxType == 2) {
101. boxType = 6;
102. }
103. **if** (boxType == 3) {
104. boxType = 4;
105. }
106. cost = (cost \* 1.14);
107. **break**;
108. **case** "No":
109. **if** (boxType == 3) {
110. boxType = 3;
111. }
112. **if** (boxType == 4 || boxType == 5) {
113. boxType = 6;
114. }
115. **break**;
116. }
118. **switch** (ReinforcedBoxCorner) {
119. **case** "Yes":
120. **if** (boxType != 5) {
121. boxType = 6;
122. }
124. cost = (cost \* 1.10);
125. **break**;
126. **case** "No":
127. **if** (boxType == 5) {
128. boxType = 6;
129. }
130. **break**;
131. }
133. **switch** (SealableBox) {
134. **case** "Yes":
135. cost = (cost \* 1.08);
136. **break**;
137. **case** "No":
138. **break**;
139. }
140. }
142. /\*\*
143. \* @return cost
144. \*/
145. **public** **double** getCost() {
146. **return** **this**.cost;
147. }
149. /\*\*
150. \* @return boxType
151. \*/
152. **public** **int** getType() {
153. **return** **this**.boxType;
154. }
155. }

# Input/Output









# Contribution form

**ADPROC Cwk - Group Contribution**

Complete the Group Members’ Contribution to Coursework **below.** This should cover the overall contribution to the coursework of each group member (remember to include your own contribution).

|  |
| --- |
| **Group Members’ Contribution to Coursework**  Distribute 100% among all the members of your group (including yourself) to indicate each person’s relative contribution.  For example, in a group of three students Alpha, Beta, and Gamma, where all have contributed evenly you would give 33.3% each.  However, if the contributions were significantly uneven, you might mark them as follows – Alpha has done most of the work, so give her/him 50%, Beta and Gamma have completed the rest of the work and between them Beta did 20% and Gamma did 30%.  List your group members **by student number** and their scores below:  1. \_\_\_\_\_\_\_\_\_\_\_\_\_777900\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_        50 /100  2. \_\_\_\_\_\_\_\_\_\_\_\_\_763734\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_         50/100  **TOTAL 100/100** |

# ADPROC Document

## ADPROC, Advanced Programming Concepts (U21266)

## 

## Coursework

Hand out date: 25.X.2016 Hand in date: 6.XII.2016 (Demonstration: by week starting 5th Dec)

This is an assessed piece of group coursework, it is therefore essential to be completed and handed-in on time. If you are unclear about any aspect of the assignment, including the assessment criteria, please raise this at the first opportunity. The usual regulations apply to a late submission of work. The submitted application must be in Java (using Java NetBeans IDE) to be marked. During the demonstration (by week 11, in your lab session) **you have to submit a disk** with your source code and Java NetBeans project files (if not already submitted to CAM office) with your student numbers on it.

**The coursework you submit should be your group work. If your coursework includes other people's ideas and material, they must be properly referenced or acknowledged. Failing to do so intentionally or unintentionally constitutes plagiarism. The University treats plagiarism as a serious offence.**

## Order system for a box-selling company

The Chinese invented cardboard in the 1600s and the English created the first commercial cardboard box in 1817.

“*FlexBox*” is a company which makes an extensive range of boxes for packaging a wide range of goods. Due to the wide range of requirements of their customers, the variety of boxes which they have to produce is very extensive.

The boxes are all rectangular and have the following characteristics:

* They are all made of card;
* The card has a specified grade;
* The boxes may have no printing, or 1, or 2 colour printing;
* Some boxes may have reinforced bottoms;
* Some boxes may have reinforced corners;
* All boxes may have sealable tops.

The types of boxes, produced by the company, are shown in Table 1.

**Table 1.** Types of cardboard boxes available.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Grade of card | Colour print | | | Reinforced bottom | Reinforced corners |
| 0 | 1 | 2 |
| I | 1 – 3 | YES | NO | NO | NO | NO |
| II | 2 – 4 | NO | YES | NO | NO | NO |
| III | 2 – 5 | NO | NO | YES | NO | NO |
| IV | 2 – 5 | NO | NO | YES | YES | NO |
| V | 3 – 5 | NO | NO | YES | YES | YES |

**Fig. 1*.*** Simple cardboard boxes.

The costs of 1m2 of card are given in Table 2.

**Table 2.** Basic cost of 1 square metre of card. **Table 3.** Additional costs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Grade of card | 1 | 2 | 3 | 4 | 5 |
| Cost per m2 [in £] | 0.50 | 0.60 | 0.72 | 0.90 | 1.4 |

|  |  |
| --- | --- |
| 1 colour | 13% extra |
| 2 colours | 16% extra |
| Reinforced bottom | 14% extra |
| Reinforced corners | 10% extra |
| Sealable tops | 8% extra |

There are some additional costs depending on whether the box has printing and if there is any reinforcing. These are shown in Table 3 and the percentage increase **is based on the basic cost**.

All boxes may have sealable tops.

When a customer asks *FlexBox* to quote a price for an order, they specify the following:

* The size of the box (width, length, and height);
* The grade of the card;
* Whether they want any colour printing (no colour, or 1, or 2 colour printing);
* Whether they want any bottom and/or corner reinforcement;
* Whether the box has a sealable top;
* The quantity of boxes.

From this information, the order system should determine if the type of box requested can be supplied by *FlexBox*, if it cannot, it should display an appropriate message and reject the order. If the ordered box/boxes correspond to any of the types given in Table 1, and can be supplied by *FlexBox*, the cost of the order must be calculated and quoted.

Customers **should not be asked for the type of the box** they order, since the type is only used within the company to calculate the cost: **your application must determine the type of the ordered boxes** (using Table 1).

If a customer is placing more than one order (say one order for 5 boxes (of type I) and another order of 10 boxes (of type II)), then he/she should receive a quote with the total cost of the orders.

Your user interface should be a GUI (graphical user interface) using AWT/Swing. If no GUI is used, you will lose the marks allocated for this part of your coursework.

## Your Task

* Write an application, which will allow the customer to enter the details of his/her order and will calculate the cost of the order. Your application should verify that *FlexBox* can supply the type of box requested (the customer should not be asked to specify the box type).
* Use OO design approach (abstraction, inheritance and polymorphism) and create a class hierarchy that describes the types of boxes *FlexBox* sells. Use an abstract class if necessary.
* Give UML use case diagram, UML class hierarchy diagram, one class and one instance diagrams.
* Use proper level of abstraction, encapsulation and accessibility for the class attributes and methods. Application with no levels of abstraction will fail.
* Devise suitable test plan and data, which you can use to test the performance of your ordering system.

## Assessment Criteria

You should give **a demonstration and submit a disk** (with your group student numbers on it) with your source code and Java NetBeans project files of your software no later than week11 (starting **5.XII.2016**), during your lab session or with your report (whichever is first).

On **6.XII.2016** your group should hand in a **report** (and a disk, if not already submitted) that includes the following:

* A UML use case diagram of placing an order, UML class hierarchy diagram of your application design, and also one UML class diagram and one instance diagram;
* A brief description of the application including any assumptions you have made and any limitations in your implementation of the application;
* A test schedule and screen shots to evidence the testing and evaluation;
* The source code you have written in an Appendix (the same code that you used in your demonstration);
* Some sample input and output (screenshots) to demonstrate your application is working;
* A Group contribution form;
* This document.

The assessment criteria and marks distribution are given in Table 4.

**Table 4.** Assessment criteria and marks distribution.

|  |  |  |  |
| --- | --- | --- | --- |
| Topic/Criteria | Comments | Marks available | Marks awarded |
| Class hierarchy descriptions (UML) | How suitable is the design and the adopted hierarchy for the application? Use of abstract class? | 10 (Report) |  |
| UML class and instance diagrams | Are the UML use case, class and instance diagrams relevant to the application? | 10 (Report) |  |
| Code and functionality | How complete is the implementation? Does it perform as specified?  Does it use an OO design approach? Use of abstract class?  Are the class attributes and methods at the appropriate hierarchy level?  Is the verification and validation of input data adequate?  Is the exception handling properly done?  Are the style, indentation and comments appropriate?  Is the layout clear? | 45  (Demo(20),  Report(25)) |  |
| Using AWT/Swing | Is the layout clear?  How well designed is the interface?  How appropriate is the use of components?  How appropriate is the use of attributes?  Is it working, or just an attempt? | 15 (Demo) |  |
| Testing | How thorough is planning and testing?  Does it cover most/few possible errors? | 10 (Report) |  |
| Supporting documentation and comments. | Is the text clearly written and well presented?  Are the assumptions, limitations, problems and features of the application well documented? | 10 (Report) |  |
| OVERALL MARK |  | 100 |  |