**Group A5 Report**

UP879389 - UP821837 - UP918156 - UP822718

# Introduction

In this coursework, the group undertook the building of an Order System for a box selling company. This task was carried out using the Java programming language in the Netbeans environment.

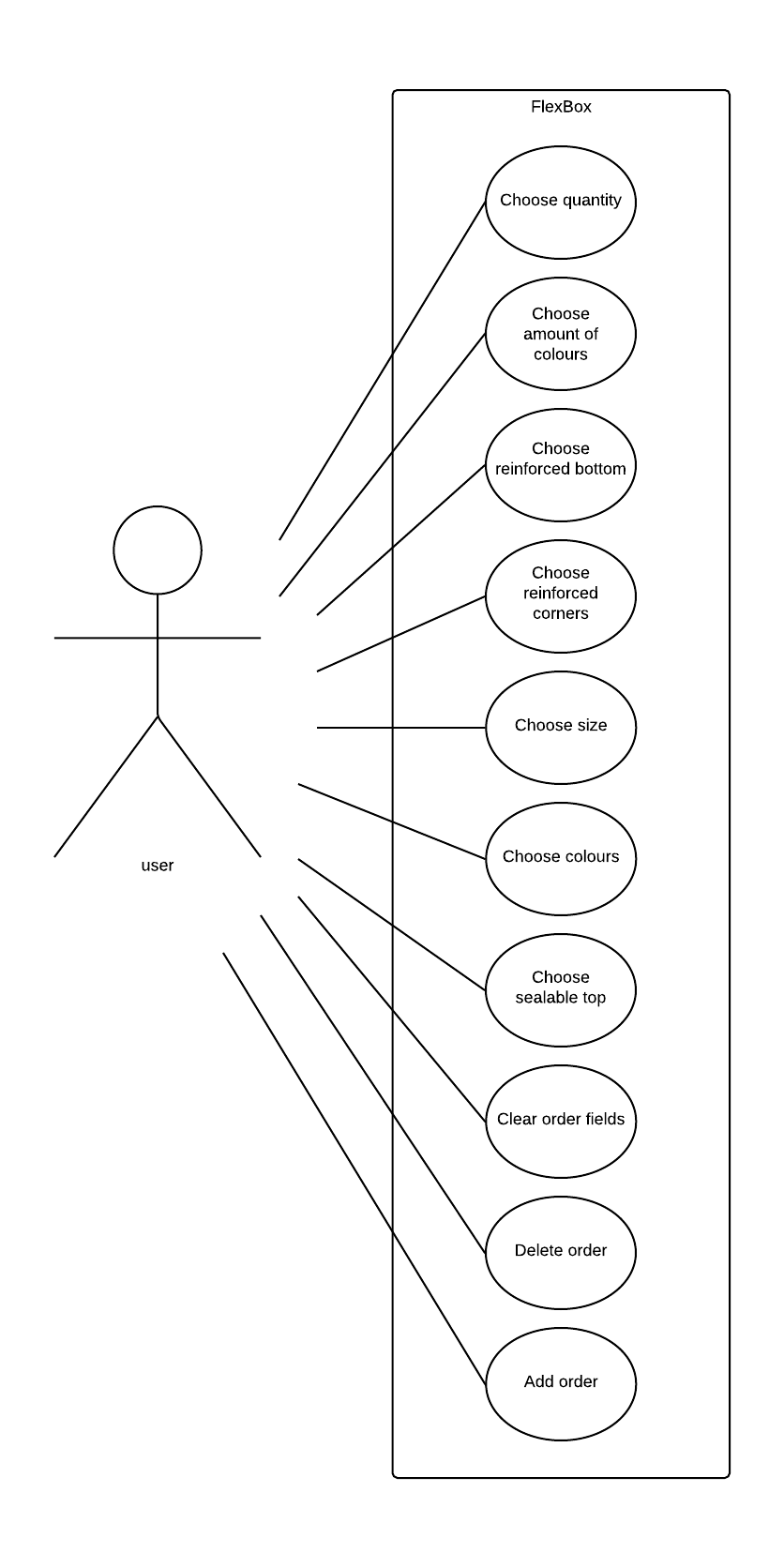
The order system had to follow the various rules the box company had set out, regarding the various types of cardboard and the additional extras the boxes could have such as: reinforced bottom, sealable top and a choice of one or two colours.

The ordering system had to be in a user-friendly Graphical User Interface, thus meaning the GUI had to have a clear and easy layout, allowing users to select the boxes they wanted, the measurements, and the price they would be paying for it.

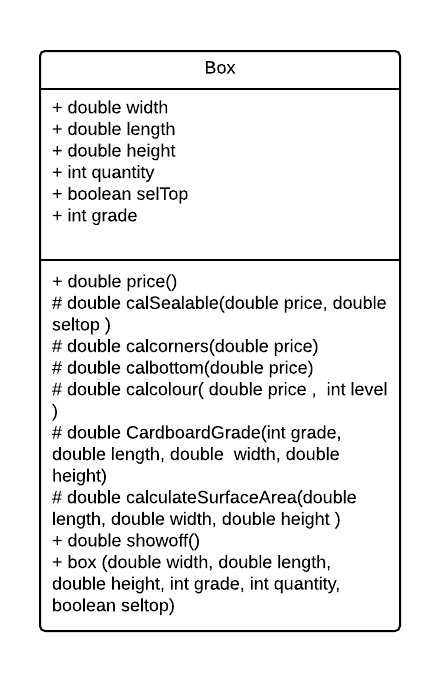
## UML diagram

* A Unified Modelling Diagram shows the relationships between the objects present in the code, using real-life examples.
* In our coursework, the Hierarchical UML helps us to plan our classes before we attempt to code in Java.
* The Hierarchical UML here shows how various subclasses have inherited from the superclass.

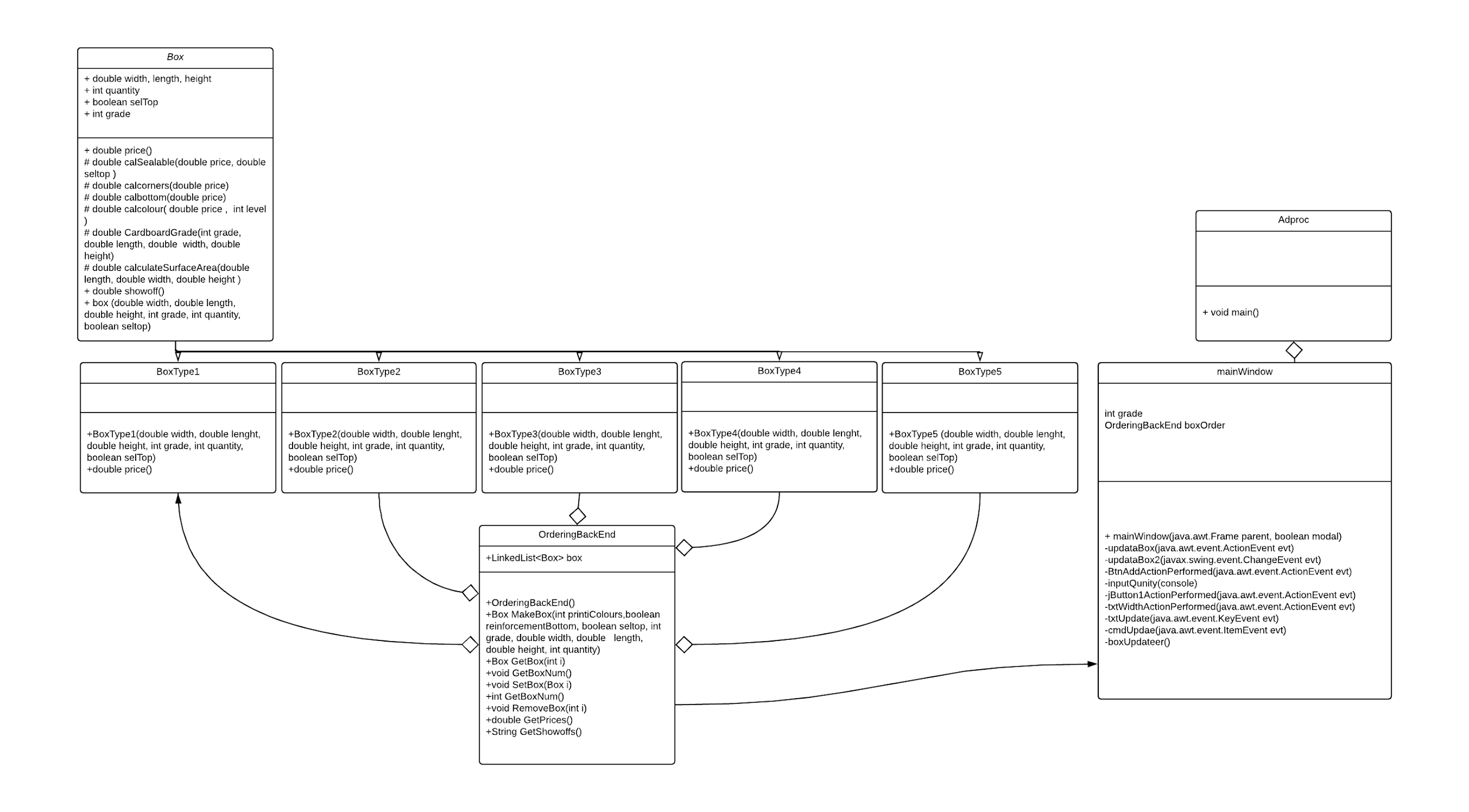
## Use case diagram

* The use case diagram shows how the user interacts with our Ordering Box System.
* The diagram shows the user interacting with the FlexBox system. All these interactions happen at the GUI of our system.
* For example: the user is able to choose a quantity of boxes needed, starting from zero, up to 900.

**Class Diagram**



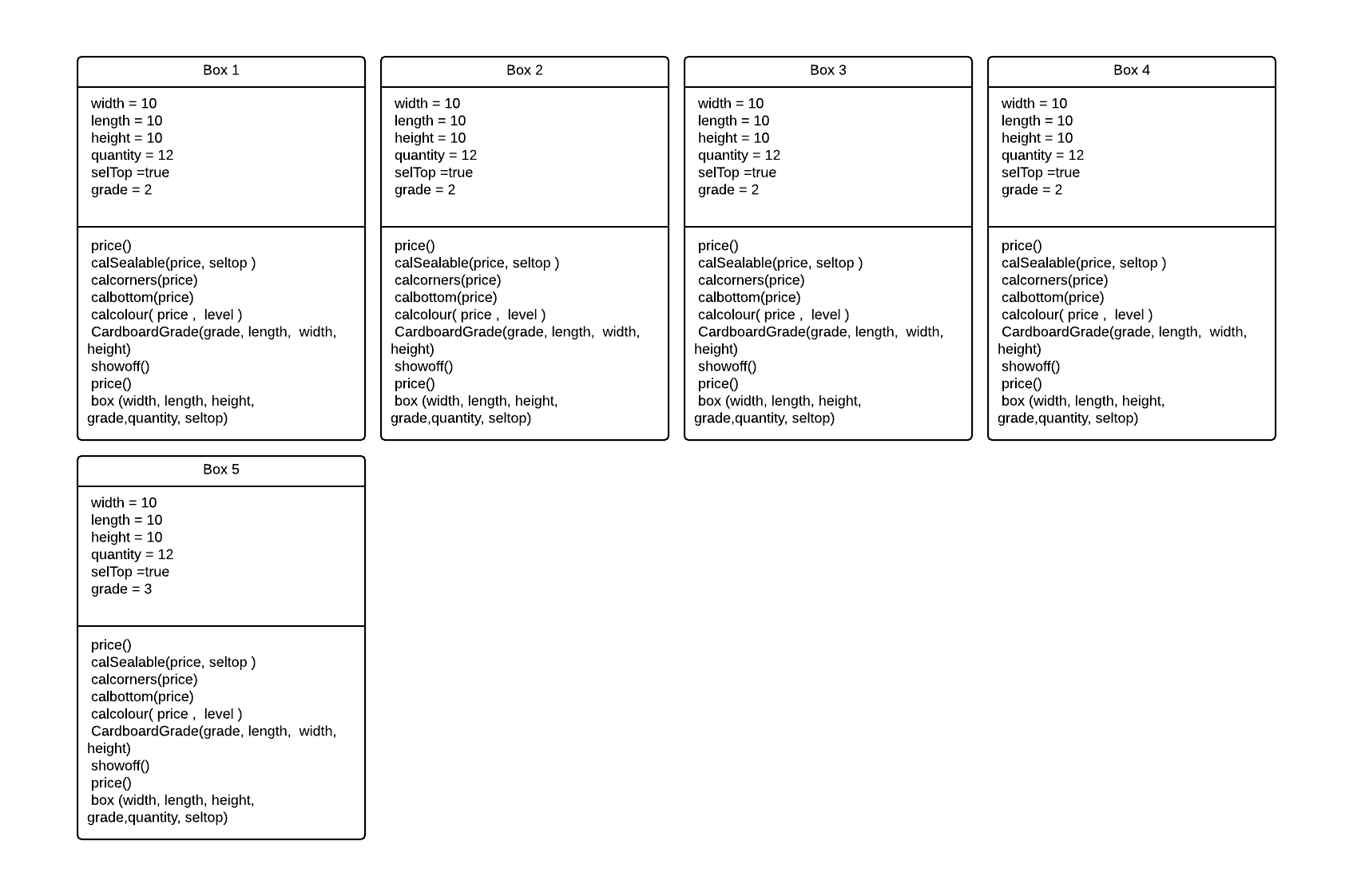
**Hierarchy**



## 

## 

## Instances



# **Assumptions and limitations**

During the making of the Box Ordering system, we had to make various assumptions based on how the boxes would be designed, and what the user would be expecting when ordering from us.

Assumption 1:

* The minimum size for a box is W: 0.01 H: 0.01 and L: 0.01. All lengths are in meters.
* This assumption was made as it would be impossible to create a box of measurements 0, furthermore, it would be a waste of money and resources to make anything smaller.

Assumption 2:

* The maximum size of a box is W: 900 H: 900 and L: 900. Again, the lengths are in meters.

Assumption 3:

* In the Graphical User Interface, a user is only able to order a maximum of 900 boxes in one purchase. Using this assumption we were able to build our GUI.

Assumption 4:

* Width Height and Length are all floating point numbers in Java.

Assumption 5:

* The maximum quantity for a single item in an order is 900.

Assumption 6:

* Inventory handling is not within the scope of the project, therefore,
* we assume infinite inventory. A reasonable improvement to this software is to implement an inventory handler.

During the coursework, there were numerous limitations that came up. Here are the following limitations:

Limitation 1:

* There is no payment service within the GUI. For examples, there is no extension to payment services such as PayPal or Visa. Upon further improvements, a consideration to enable such extensions is necessary in order to optimize the ordering system.

Limitation 2:

* A further limitation was the lack of cost to start off with.

Limitation 3:

* The software can’t update itself automatically when some values are edited, the user must make a few changes before the price reflects all the changes.

Limitation 4:

* The software does not provide web or mobile features, therefore, the program is only accessible on personal computers. This would be out of the scope of the project’s specification.

Limitation 5:

* The software does not provide a final invoice for the order. However, this is outside the scope of the project.

Limitation 6:

* The software does not register user information due to it being out of the scope of the project.

# Testing

During the programming, we ran some tests to ensure the GUI and code worked as we presumed. Below are some of the tests we ran during the creation of our program. For each test number, see appendices for corresponding screenshot evidence.

**Test group 1 (broad testing)**

1. If any/some/all of the quantity, colour, width, length, height, that is set to positive or negative infinity/NaN/negative number/text, this should cause an error of some description because of invalid input
2. Making sure the correct box was made and the final price was correct
3. making sure the Range is limited correctly W:0.1 H:0.1 L:0.1 to W:899 H:899 L:899 inclusive
4. Impossible boxes, for example, a grade 1 with reinforced sides and bottoms with two colours
5. The test showing the add box function
6. The test showing what happens if a letter is inputted
7. Test showing the input of a negative number
8. Testing that grade 3 corresponds to the table

**Test group 2 (fine grain testing)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Screenshot | W,H,L | Sealable | Bottom | corners | Grade | quantity | Colour | Expected Output | Actual Result |
| 1 | 10,10,10 | N | N | N | 1 | 1 | 0 | Type 1 £333.00 | Type 1 £333.00 |
| 2 | 10,10,10 | N | N | N | 2 | 1 | 1 | Type 2 £436.80 | Type 2 £436.80 |
| 3 | 10,10,10 | N | N | N | 2 | 1 | 2 | Type 3 £448.50 | Type 3 £448.50 |
| 4 | 10,10,10 | N | Y | N | 3 | 1 | 2 | Type 4 £629.76 | Type 4 £629.76 |
| 5 | 10,10,10 | N | Y | Y | 3 | 1 | 2 | Type 5 £688.80 | Type 5 £688.80 |
| 6 | 10,10,10 | Y | N | N | 1 | 1 | 0 | Type 1 £363.00 | Type 1 £363.00 |
| 7 | 10,10,10 | Y | N | N | 2 | 1 | 1 | Type 2 £475.80 | Type 2 £475.80 |
| 8 | 10,10,10 | Y | N | N | 2 | 1 | 2 | Type 3 £487.50 | Type 3 £487.50 |
| 9 | 10,10,10 | Y | Y | N | 3 | 1 | 2 | Type 4 £678.96 | Type 4 £678.96 |
| 10 | 10,10,10 | Y | Y | Y | 3 | 1 | 2 | Type 5 £738.00 | Type 5 £738.00 |
| 11 | 10,10,10 | N | Y | Y | 5 | 1 | 1 | invalid | invalid |
| 12 | 10,10,10 | N | N | Y | 1 | 1 | 2 | invalid | invalid |
| 13 | 10,10,10 | N | N | Y | 5 | 1 | 2 | invalid | invalid |
| 14 | 10,10,10 | N | N | Y | 4 | 1 | 2 | invalid | invalid |
| 15 | 10,10,10 | N | N | Y | 2 | 1 | 2 | invalid | invalid |
| 16 | 10,10,10 | N | N | N | 1 | 2 | 0 | Type 1 £660.00 | Type 1 £660.00 |
| 17 | NaN,10,10 | N | N | N | 1 | 2 | 0 | invalid | invalid |
| 18 | 901,10,10 | N | N | N | 1 | 2 | 0 | invalid | invalid |

# Sample input and output

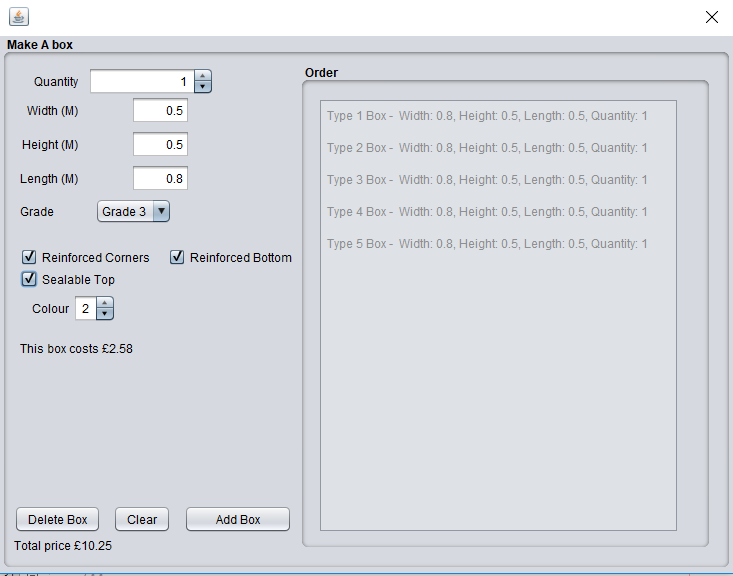
|  |  |
| --- | --- |
|  | **Input**  Quantity: 1  Width: 10  Height: 10  Length: 10  Grade: 3  Reinforced corners: Yes  Reinforced bottom: Yes  Sealable top: Yes  Colour: 2  **Output**  Type 5 Box, Cost: £738.00 |
| **Input**  Quantity: 2  Width: 10  Height: 10  Length: 10  Grade: 2  Reinforced corners: No  Reinforced bottom: No  Sealable top: No  Colour: 0  **Output**  Type 1 Box, Cost: £666.00 |  |
|  | **Input**  Quantity: 1  Width: 10  Height: 10  Length: 10  Grade: 4  Reinforced corners: Yes  Reinforced bottom: No  Sealable top: No  Colour: 2  **Output**  Invalid |
| **Input**  Quantity: 1  Width: 10  Height: 10  Length: 10  Grade: 3  Reinforced corners: Yes  Reinforced bottom: No  Sealable top: No  Colour: 0  **Output**  Invalid |  |

# Appendices

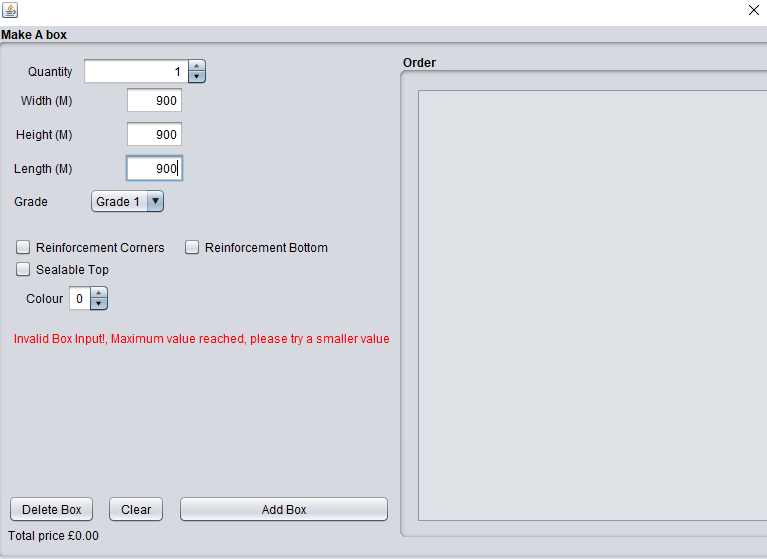
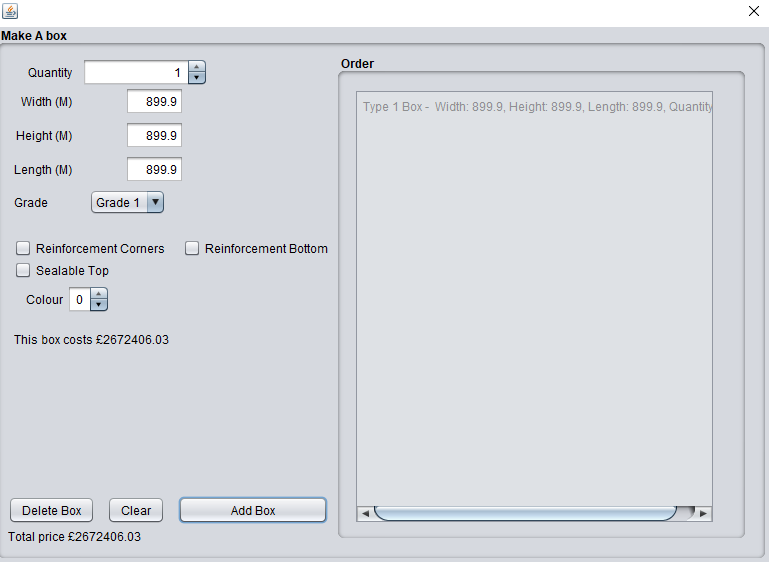
**Testing**

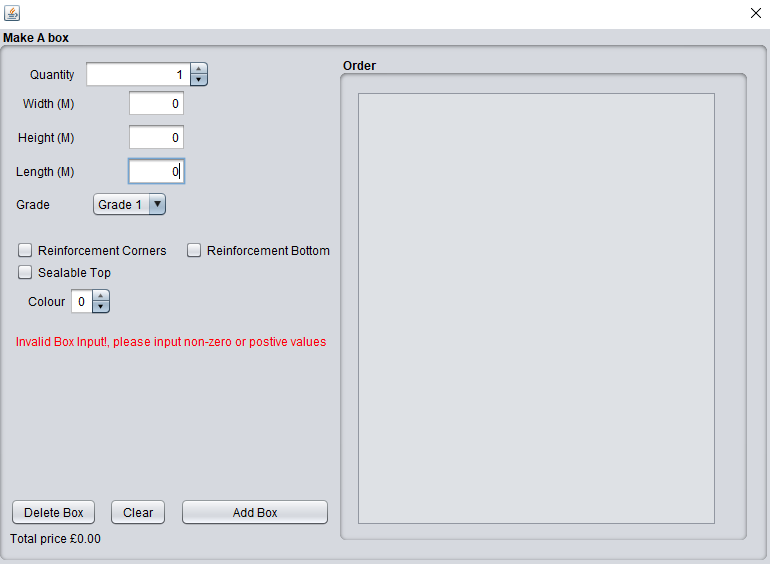
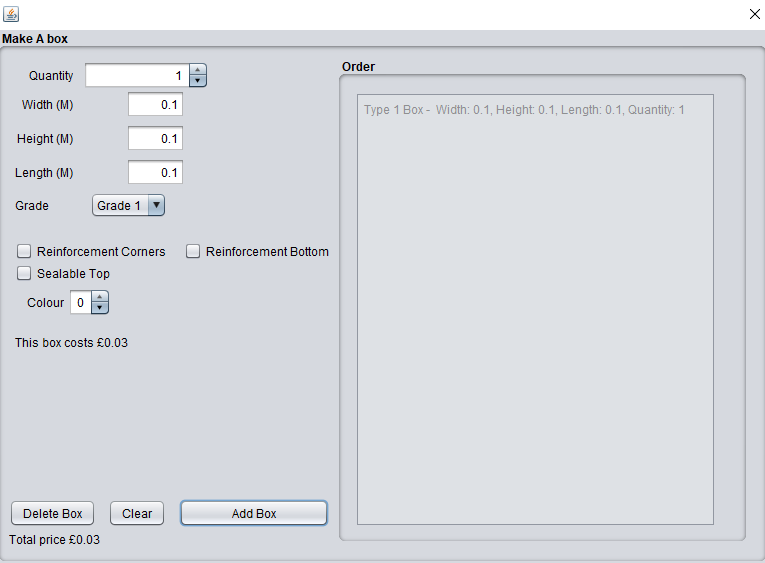
**Test group 1**

**2.**

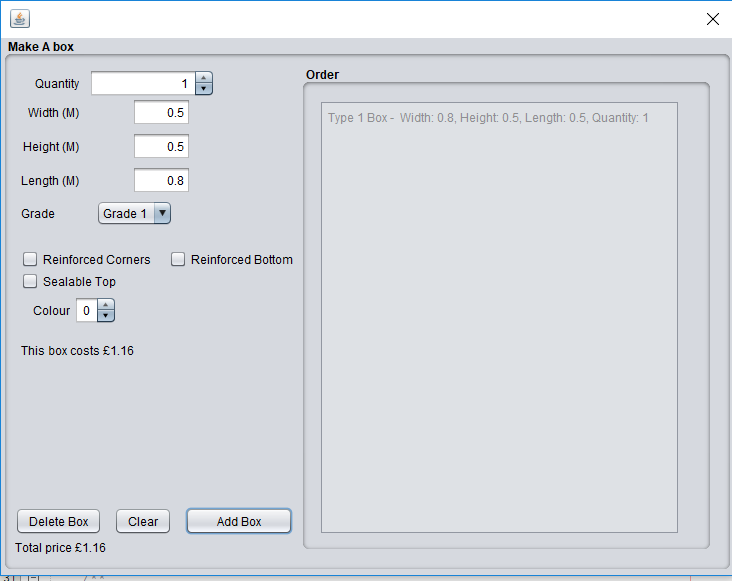


**3.**

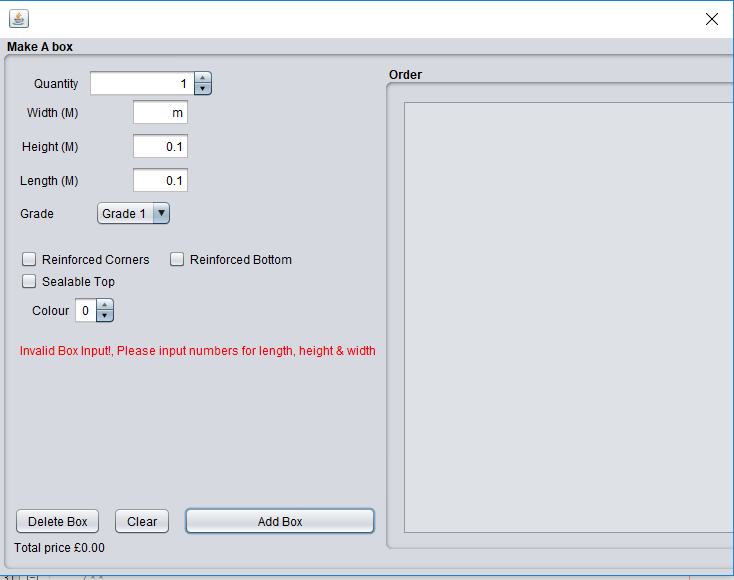
****

****

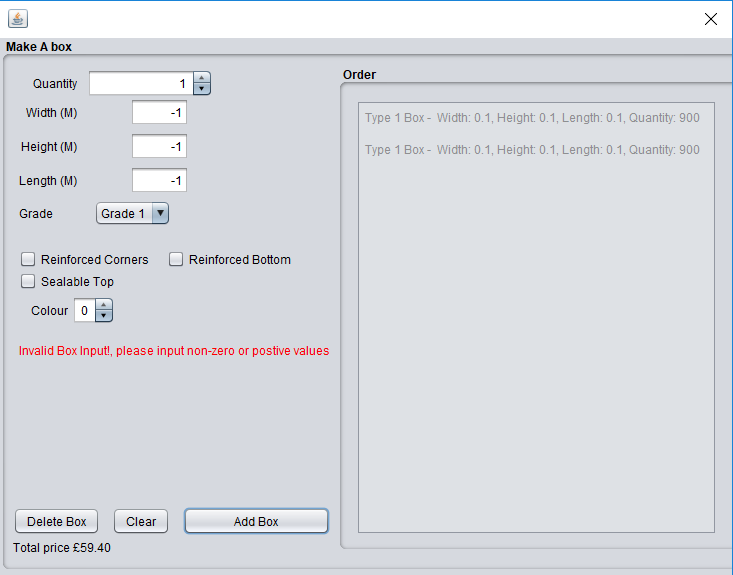
**5.**



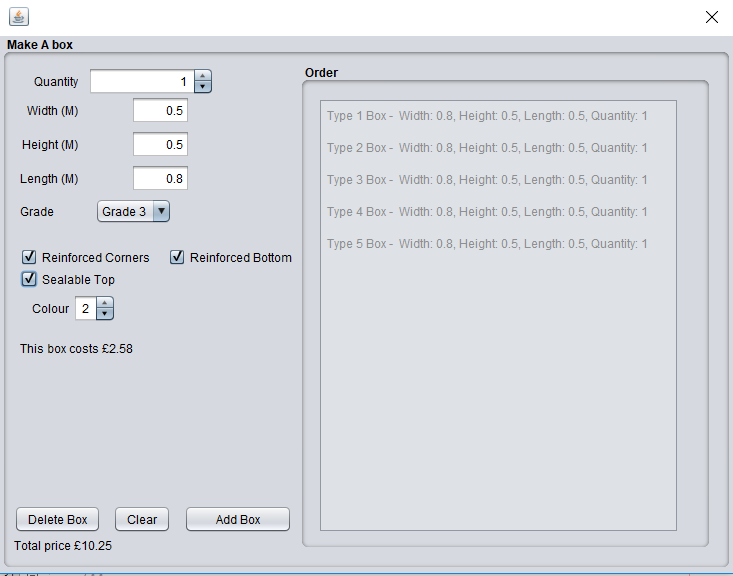
**6.**



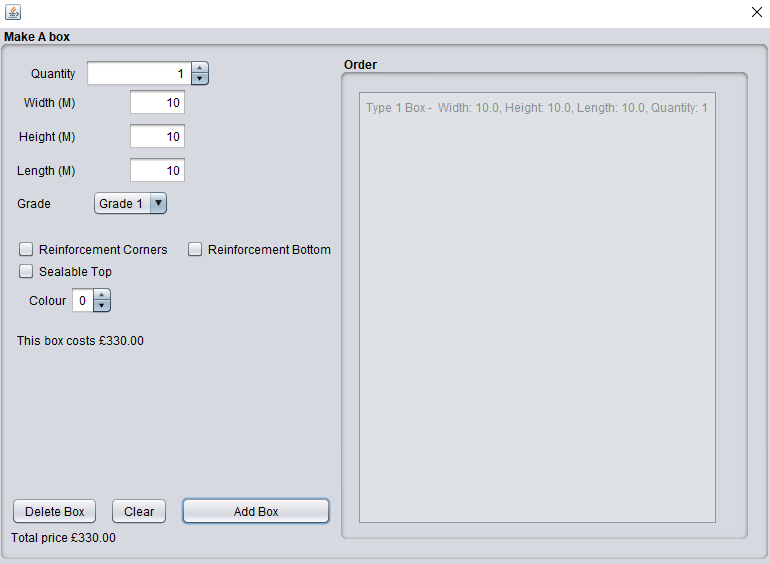
**7.**

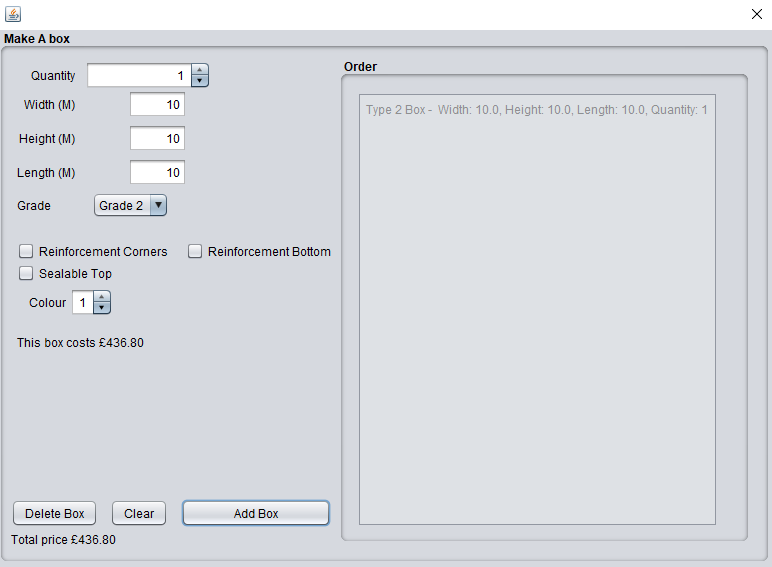


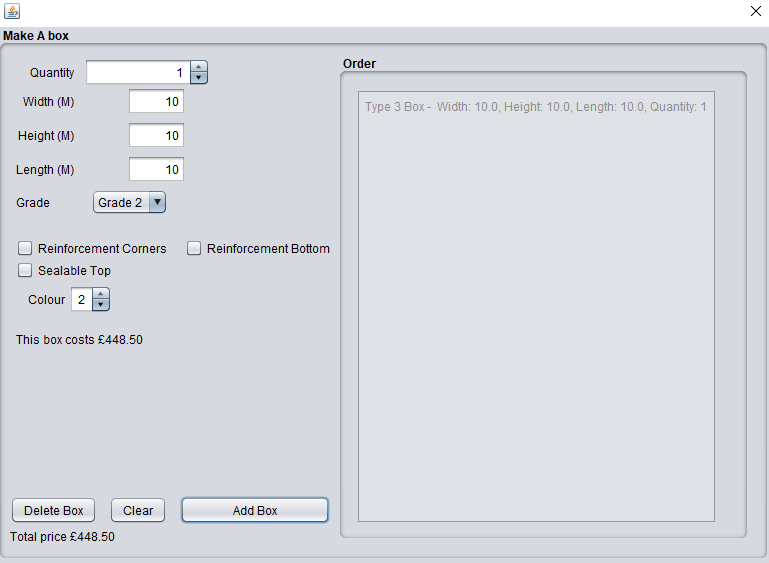
**8.**

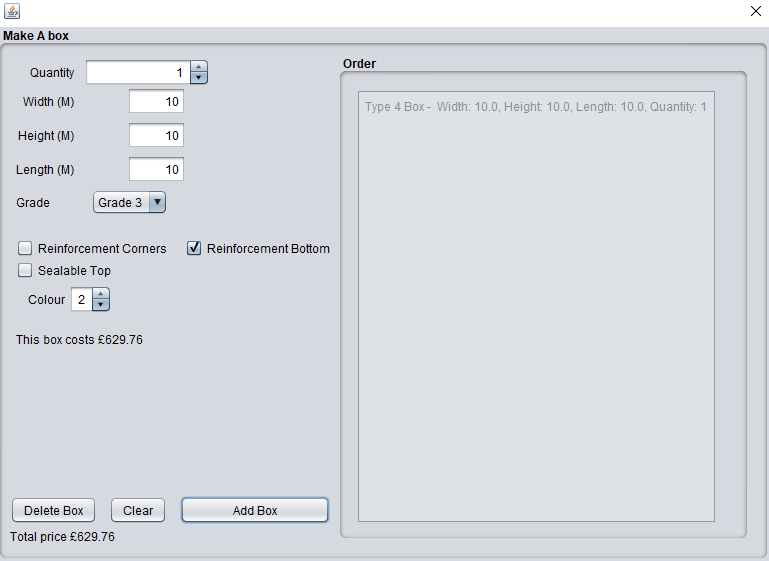


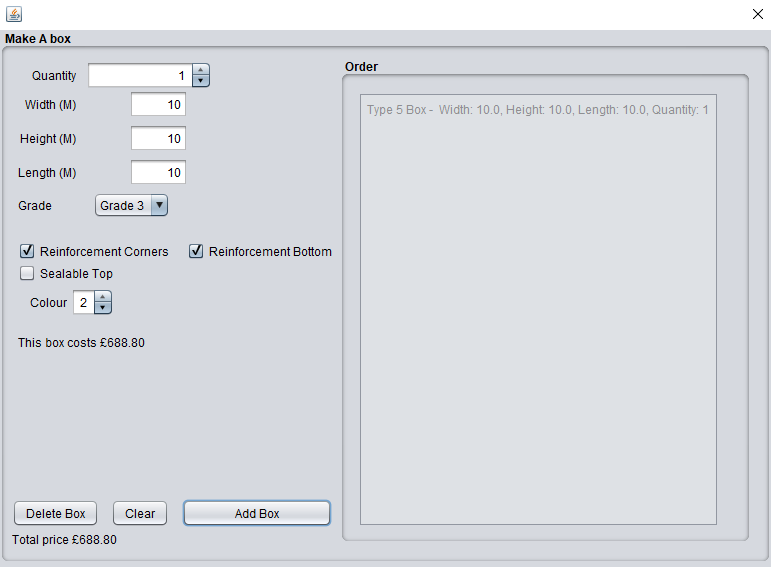
**Testing Group 2**

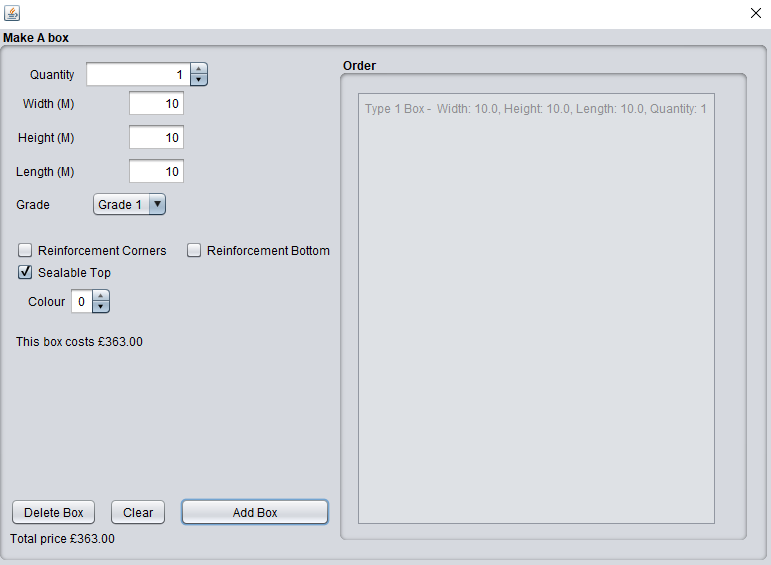
**1.**

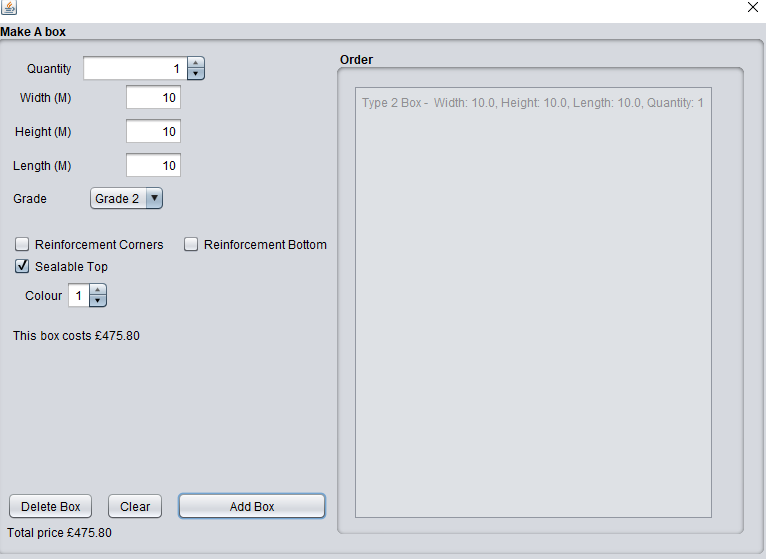
**2.**

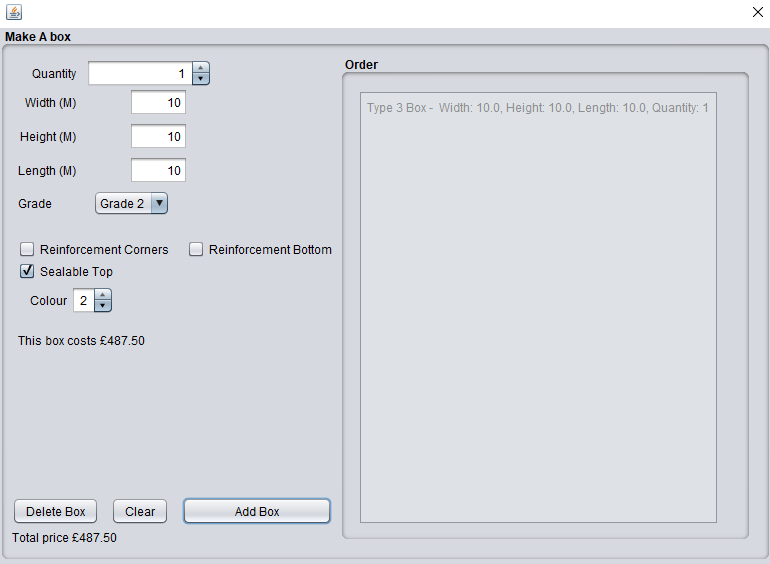
**3.**

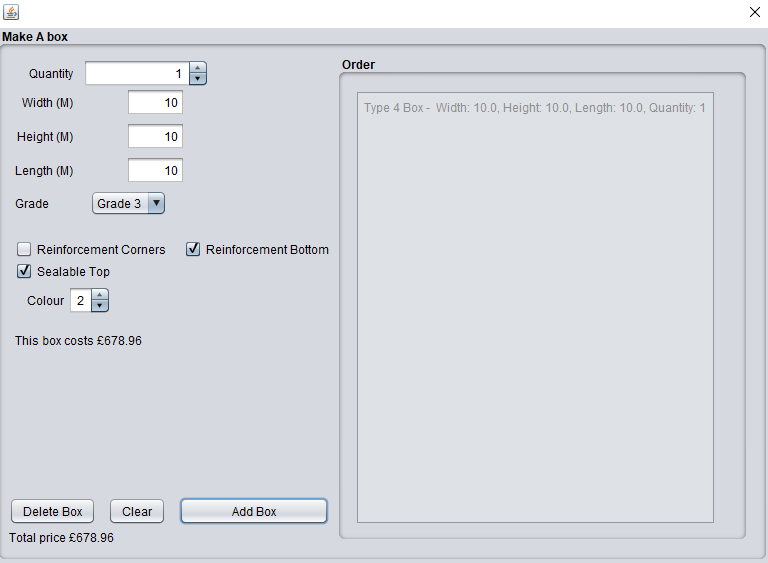
**4.**

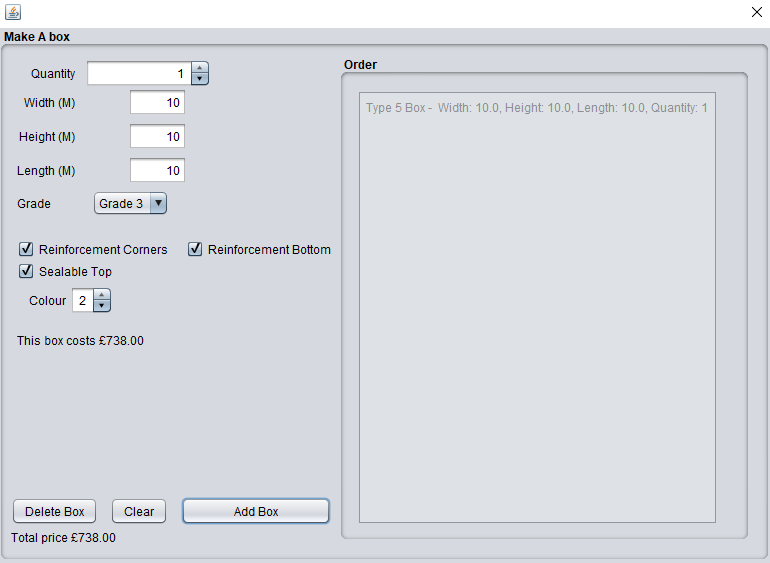
**5.**

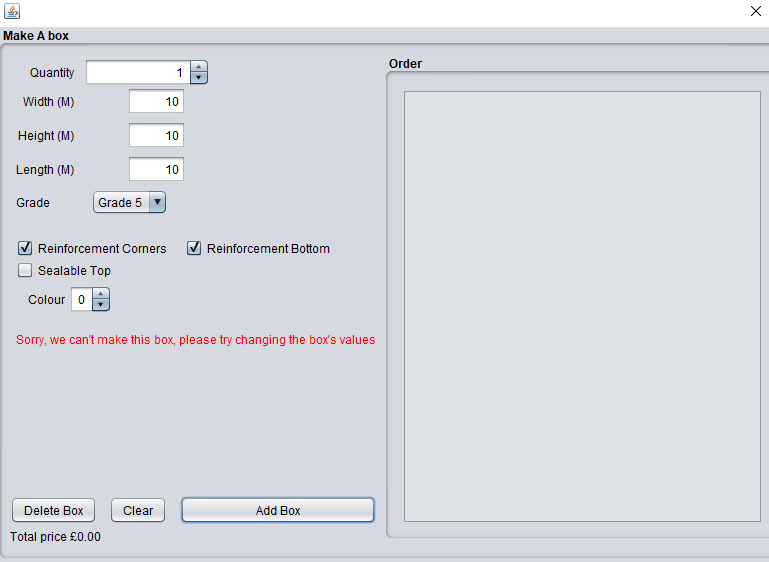
**6.**

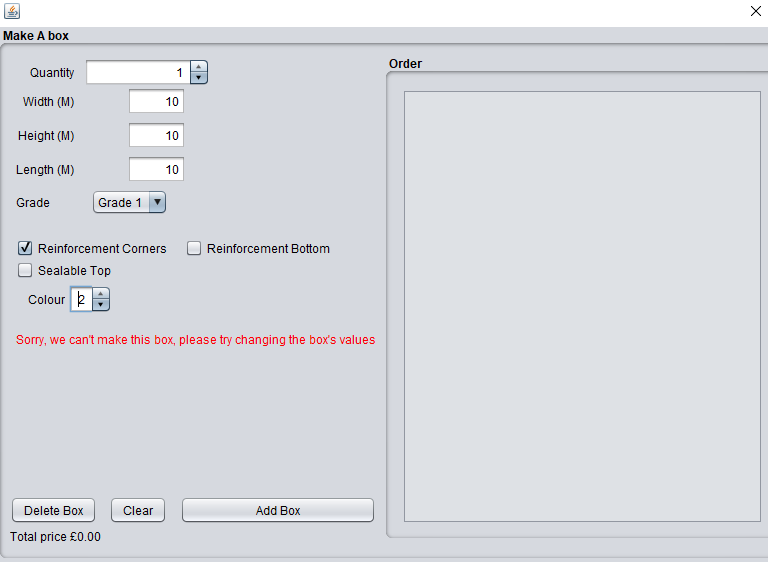
**7.**

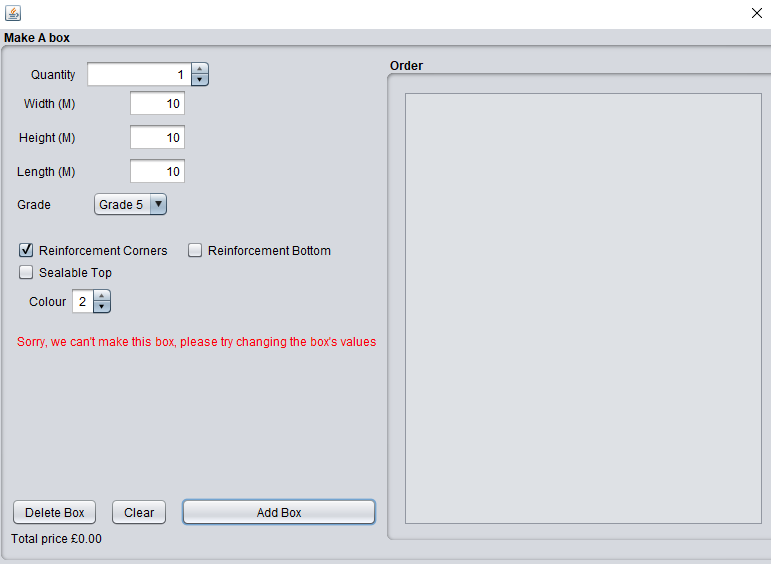
**8.**

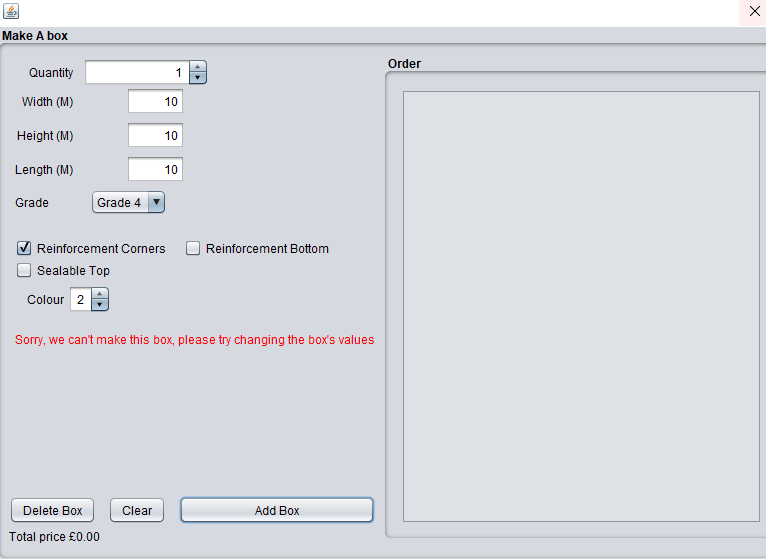
**9.**

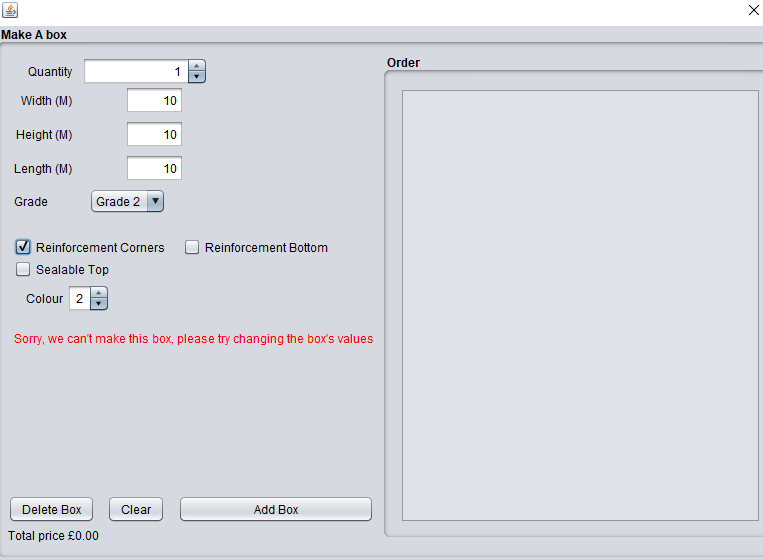
**10.**

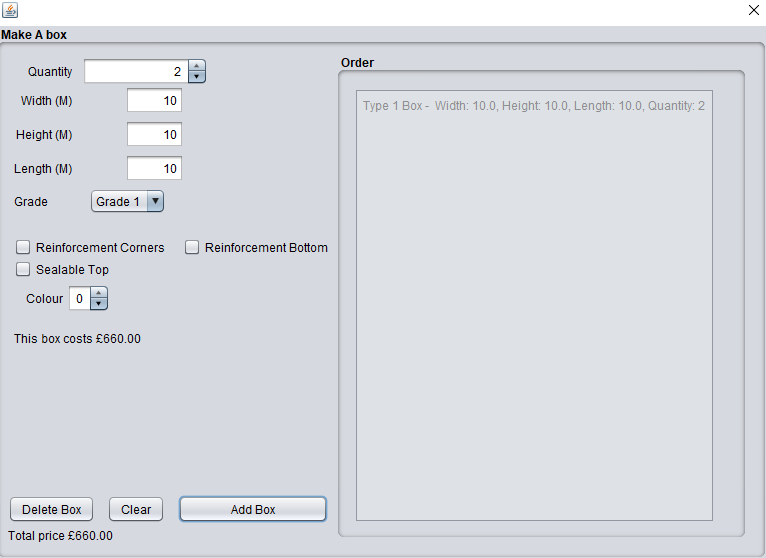
**11.**

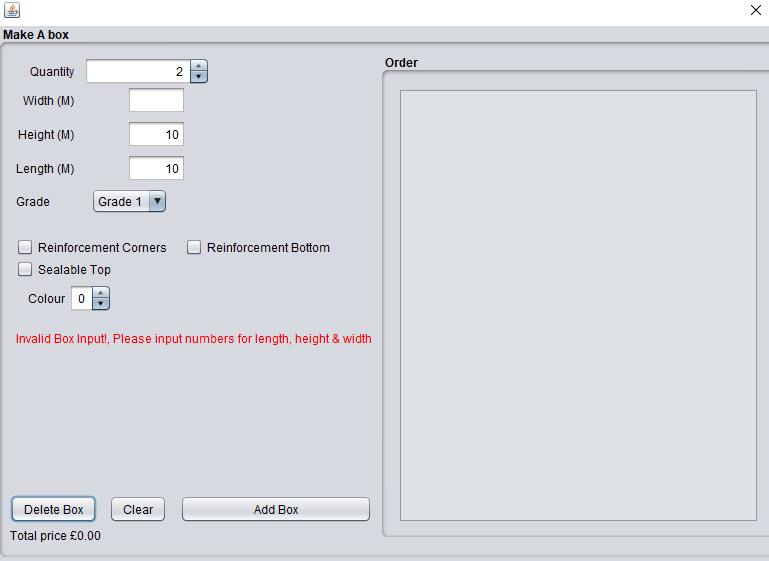
**12.**

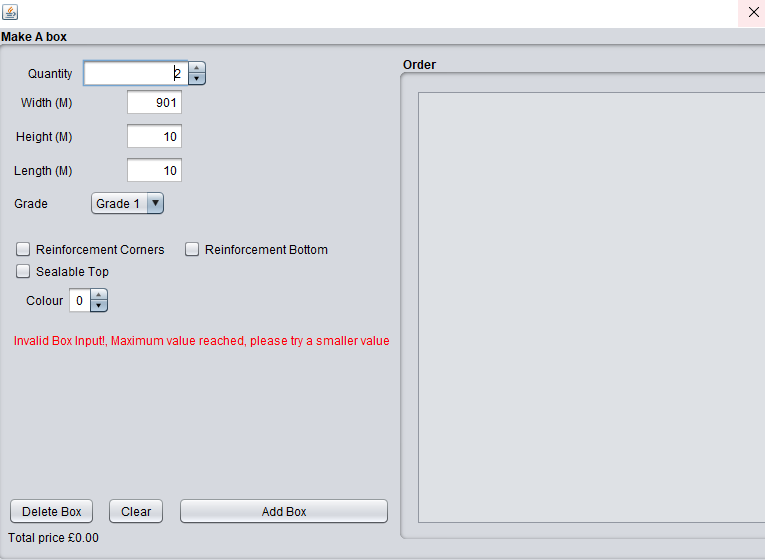
**13.**

**14.**

**15.**

**16.**

**17.**

**18.**

**Source code**

### 

### **Adproc.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/ package adproc;  import java.util.ArrayList; import java.util.Scanner;  /\*\*  \*application entry point class  \* @author william  \*/ public class Adproc{   /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args)  {  /\* Set the Nimbus look and feel \*/  //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html   \*/  try {  for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  if ("Nimbus".equals(info.getName())) {  javax.swing.UIManager.setLookAndFeel(info.getClassName());  break;  }  }  } catch (ClassNotFoundException ex) {  java.util.logging.Logger.getLogger(mainWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (InstantiationException ex) {  java.util.logging.Logger.getLogger(mainWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (IllegalAccessException ex) {  java.util.logging.Logger.getLogger(mainWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (javax.swing.UnsupportedLookAndFeelException ex) {  java.util.logging.Logger.getLogger(mainWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  }  //</editor-fold>   /\* Create and display the dialog \*/  java.awt.EventQueue.invokeLater(new Runnable() {  public void run() {  mainWindow dialog = new mainWindow(new javax.swing.JFrame(), true);  dialog.addWindowListener(new java.awt.event.WindowAdapter() {  @Override  public void windowClosing(java.awt.event.WindowEvent e) {  System.exit(0);  }  });  dialog.setVisible(true);  }  });  }   } |

### 

### **Box.java**

|  |
| --- |
| package adproc;  import sun.reflect.generics.reflectiveObjects.NotImplementedException;  /\*\*  \*a model for the generic box  \* @author william  \*/ public abstract class Box {  //Fields  double length;  double width;  double height;  int grade;  int quantity;  boolean seltop; //Sealed top  //Constructor     /\*\*  \*constructor to make a box  \* @param width the width of the box  \* @param length the length of the box  \* @param height the height of the box  \* @param grade the grade of the box  \* @param quantity the quantity of the box  \* @param seltop the seltop of the box  \*/  public Box (double width, double length, double height, int grade,int quantity,boolean seltop)  {  this.width = width;  this.length = length;  this.height = height;  this.grade = grade;  this.quantity = quantity;  this.grade =grade;  this.seltop = seltop;  }  /\*\*  \*Get the price of the box  \* @return return the price   \*/  public abstract double price();    /\*\*  \*return a string describes a box  \* @return describes a box   \*/  public abstract String showoff();    /\*\*  \*works out the the great cross times the size of the cardboard box  \* @param length original length of box  \* @param width original width of box  \* @param height original height of box  \* @param price original level of grade  \* @return the great cross times the size of the cardboard box   \*/  protected double CardboardGrade(int grade,double length, double width, double height)  {  double size = (width\*length\*2) + (height\*length\*2) + (height\*width\*2);  switch (grade)  {  case 1:  return 0.55\*size;  case 2:  return 0.65\*size;  case 3:  return 0.82\*size;  case 4:  return 0.98\*size;  case 5:  return 1.5\*size;  default:  throw new AssertionError();  }  }  /\*\*  \*calculate the additional cost of the colour print  \* @param price original price of box  \* @param level original level of coloer  \* @return colour print cost   \*/  protected double calcolour(double price , int level )  {  switch (level) {  case 0:  return 0 \* price;  case 1:  return 0.12 \* price;  case 2:  return 0.15 \* price;  default:  throw new AssertionError();  }  }  /\*\*  \*calculate the cost of having a reinforced   \* @param price original price of box  \* @return cost reinforced bottom  \*/  protected double calbottom(double price)  {  return 0.13 \* price;   }  /\*\*  \*calculate the cost of having a corners   \* @param price original price of box  \* @return cost corners bottom  \*/  protected double calcorners(double price)  {  return 0.12 \* price;   }  /\*\*  \*calculate the cost of having a Sealable top   \* @param price price of box  \* @param seltop to determine if it has a sealable top or not  \* @return cost corners Sealable top  \*/  protected double calSealable(double price, boolean seltop )  {  if (seltop)  {  return price\*0.10;  };  return 0;  } } |

### 

### **BoxType1.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/ package adproc;  import sun.reflect.generics.reflectiveObjects.NotImplementedException;  /\*\*  \* a model for creating a box Type 1  \* @author william  \*/ public class BoxType1 extends Box {  //Constructor of Box Type 1    /\*\*  \*constructor to make a box  \* @param width the width of the box  \* @param length the length of the box  \* @param height the height of the box  \* @param grade the grade of the box  \* @param quantity the quantity of the box  \* @param seltop the seltop of the box  \*/  public BoxType1(double width, double length, double height, int grade,int quantity,boolean seltop)  {  super ( width, length, height, grade, quantity, seltop);  }  /\*\*  \*Get the price of the box  \* @return return the price   \*/  public double price()  {  double price = CardboardGrade(grade,length,width, height);  price += calSealable(price, seltop );  return price \* quantity;  }  /\*\*  \*return a string describes a box  \* @return describes a box   \*/  public String showoff()  {  return "Type 1 Box " + " width:" + width+ " height:" + height + " length:" + length + "\n" ;   }   } |

### 

### **BoxType2.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/ package adproc;  import sun.reflect.generics.reflectiveObjects.NotImplementedException;  /\*\*  \* a model for creating a box Type 2  \* @author william  \*/ public class BoxType2 extends Box  {  //Constructor of box Type 2  /\*\*  \*constructor to make a box  \* @param width the width of the box  \* @param length the length of the box  \* @param height the height of the box  \* @param grade the grade of the box  \* @param quantity the quantity of the box  \* @param seltop the seltop of the box  \*/  public BoxType2(double width, double length, double height, int grade,int quantity,boolean seltop)  {  super(width, length, height, grade, quantity,seltop);  }  /\*\*  \*Get the price of the box  \* @return return the price   \*/  public double price()  {  double price = CardboardGrade(grade,length,width, height);  double colour = calcolour( price , 1);  double sealable = calSealable(price, seltop );  return (colour+price+sealable) \* quantity;  }  /\*\*  \*return a string describes a box  \* @return describes a box   \*/  public String showoff()  {  return "Type 2 Box " + " width:" + width+ " height:" + height + " length:" + length + "\n" ;   }  } |

### 

### **BoxType3.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/ package adproc;  import sun.reflect.generics.reflectiveObjects.NotImplementedException;  /\*\*  \* a model for creating a box Type 3  \* @author william  \*/ public class BoxType3 extends Box {  /\*\*  \*constructor to make a box  \* @param width the width of the box  \* @param length the length of the box  \* @param height the height of the box  \* @param grade the grade of the box  \* @param quantity the quantity of the box  \* @param seltop the seltop of the box  \*/  public BoxType3(double width, double length, double height, int grade,int quantity,boolean seltop)  {  super ( width, length, height, grade, quantity, seltop);  }  /\*\*  \*Get the price of the box  \* @return return the price   \*/  public double price()  {  double price = CardboardGrade(grade,length,width, height);  double colour = calcolour( price , 2);  double sealable = calSealable(price, seltop );  return (colour+price+sealable) \* quantity;  }   /\*\*  \*return a string describes a box  \* @return describes a box   \*/  public String showoff()  {  return "Type 3 Box " + " width:" + width+ " height:" + height + " length:" + length + "\n" ;   }   } |

### 

### **BoxType4.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/ package adproc;  import sun.reflect.generics.reflectiveObjects.NotImplementedException;  /\*\*  \* a model for creating a box Type 4  \* @author william  \*/ public class BoxType4 extends Box {  boolean botton ;  /\*\*  \*constructor to make a box  \* @param width the width of the box  \* @param length the length of the box  \* @param height the height of the box  \* @param grade the grade of the box  \* @param quantity the quantity of the box  \* @param seltop the seltop of the box  \*/  public BoxType4(double width, double length, double height, int grade,int quantity,boolean seltop)  {  super ( width, length, height, grade, quantity, seltop);  }  /\*\*  \*Get the price of the box  \* @return return the price   \*/  public double price()  {  double price = CardboardGrade(grade,length,width, height);  double colour = calcolour( price , 2);  double bottom = calbottom(price);  double sealable = calSealable(price, seltop );  return (colour+price+bottom +sealable ) \* quantity;  }   /\*\*  \*return a string describes a box  \* @return describes a box   \*/  public String showoff()  {  return "Type 4 Box " + " width:" + width+ " height:" + height + " length:" + length + "\n" ;   } } |

### 

### **BoxType5.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/ package adproc;  import static com.sun.javafx.geom.BaseBounds.BoundsType.BOX; import sun.reflect.generics.reflectiveObjects.NotImplementedException;  /\*\*  \* a model for creating a box Type 5  \* @author william  \*/ public class BoxType5 extends Box {  boolean corner;  /\*\*  \*constructor to make a box  \* @param width the width of the box  \* @param length the length of the box  \* @param height the height of the box  \* @param grade the grade of the box  \* @param quantity the quantity of the box  \* @param seltop the seltop of the box  \*/  public BoxType5(double width, double length, double height, int grade,int quantity,boolean seltop)  {  super ( width, length, height, grade, quantity,seltop);  }  public double price()  {  double price = CardboardGrade(grade,length,width, height);  double corners = calcorners(price);  double colour = calcolour( price , 2);  double bottom = calbottom(price);  double sealable = calSealable(price, seltop );  return (colour+price+bottom+corners + sealable) \* quantity;  }  /\*\*  \*return a string describes a box  \* @return describes a box   \*/  public String showoff()  {  return "Type 5 Box " + " width:" + width+ " height:" + height + " length:" + length + "\n" ;   } } |

### 

### **OrderingBackEnd.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/ package adproc;  import java.util.LinkedList; import java.util.List;  /\*\*  \* a model for the the order backend  \* @author User  \*/ public class OrderingBackEnd {    LinkedList<Box> box;  OrderingBackEnd ()  {  box = new LinkedList<Box>();   }  /\*\*  \* make the correct box  \* @param printiColers the number of colours printed on the box   \* @param reinforcementBottom if the box has a reinforced bottom or not  \* @param reinforcementConres if the box has a reinforced corners or not  \* @param seltop if the box has a reinforced sealable top or not  \* @param grade the grade of the cardboard  \* @param width the width of the blocks  \* @param length length of the box  \* @param height the height of the box  \* @param quantity the quantity of the box created  \* @return then new box   \*/  public Box MakeBox(int printiColers,boolean reinforcementBottom,boolean reinforcementConres,  boolean seltop, int grade, double width, double length, double height, int quantity)  {  if(printiColers == 0 & !reinforcementBottom & !reinforcementConres & grade >=1 & grade <=3 ) //I  {  return (new BoxType1( width, length, height, grade,quantity, seltop));  }  else if(printiColers == 1& !reinforcementBottom & !reinforcementConres & grade >=2 & grade <=4) //II  {  return (new BoxType2( width, length, height, grade,quantity, seltop));  }  else if(printiColers == 2 & !reinforcementBottom & !reinforcementConres & grade >=2 & grade <=5 ) //III  {  return (new BoxType3( width, length, height, grade,quantity, seltop));  }  else if(printiColers == 2 & !reinforcementBottom & reinforcementConres & grade >=2 & grade <=5 ) //IV  {  return(new BoxType4 (width,length,height,grade,quantity, seltop));  }  else if(printiColers == 2 & reinforcementBottom & reinforcementConres & grade >=3 & grade <=5 ) //V  {  return ( new BoxType5( width, length, height, grade,quantity, seltop));  }    return null;  }  /\*\*  \*gat a box form the order  \* @param i the index to get the box at  \* @return the box at index  \*/  public Box GetBox(int i)  {  return box.get(i);  }  /\*\*  \*Adds a box to the order  \* @param i the box to be added to the order  \*/  public void SetBox(Box i)  {  box.add(i);  }  /\*\*  \*get the number of boxes  \* @return get the number of boxes  \*/  public int GetBoxNum()  {  return box.size();   }  /\*\*  \*remove the box at index  \* @param i the index to remove the box at  \*/  public void RemoveBox(int i)  {  box.remove(i);  }  /\*\*  \*get all the Prices  \* @return get all the Prices  \*/  public double GetPrices()  {  double cost = 0;  for (int i = 0; i < box.size(); i++)  {  cost += box.get(i).price();  }  return cost;  }  /\*\*  \*get all the show of strings  \* @return get all the show of strings  \*/  public String GetShowoffs()  {  String cost = "";  for (int i = 0; i < box.size(); i++)  {  cost += box.get(i).showoff() +"\n";  }  return cost;  } } |

### 

### **mainWindow.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/ package adproc;  import java.math.BigDecimal;  /\*\*  \* the main window display  \* @author william  \*/ public class mainWindow extends javax.swing.JDialog {   /\*\*  \* Creates new form mainWindow  \*/  int grade = 1;  OrderingBackEnd boxOrder =null;    /\*\*  \*constructor to make a box  \* @param parent parent window frame  \* @param modal the window mode   \*/  public mainWindow(java.awt.Frame parent, boolean modal)  {    super(parent, modal);  initComponents();  boxOrder = new OrderingBackEnd();  boxUpdateer();  }  /\*\*  \* This method is called from within the constructor to initialize the form.  \* WARNING: Do NOT modify this code. The content of this method is always  \* regenerated by the Form Editor.  \*/  @SuppressWarnings("unchecked")  // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  private void initComponents() {   buttonGroup1 = new javax.swing.ButtonGroup();  jPanel1 = new javax.swing.JPanel();  jLabel2 = new javax.swing.JLabel();  jLabel3 = new javax.swing.JLabel();  jLabel1 = new javax.swing.JLabel();  checkConres = new javax.swing.JCheckBox();  checkBottom = new javax.swing.JCheckBox();  checkSeltop = new javax.swing.JCheckBox();  Colour = new javax.swing.JLabel();  jButton1 = new javax.swing.JButton();  BtnAdd = new javax.swing.JButton();  lblPrices = new javax.swing.JLabel();  txtTotlePrice = new javax.swing.JLabel();  spinQuantity = new javax.swing.JSpinner();  jLabel6 = new javax.swing.JLabel();  jLabel7 = new javax.swing.JLabel();  spinColour = new javax.swing.JSpinner();  txtWidth = new javax.swing.JTextField();  txtLength = new javax.swing.JTextField();  txtHeigth = new javax.swing.JTextField();  jComboBox1 = new javax.swing.JComboBox<>();  jPanel2 = new javax.swing.JPanel();  jScrollPane1 = new javax.swing.JScrollPane();  txtInvoice = new javax.swing.JTextArea();   setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE);   jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("make A box"));   jLabel2.setText("Width");   jLabel3.setText("Height");   jLabel1.setText("Length");   checkConres.setText("Reinforcement Conres");  checkConres.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  updataBox(evt);  }  });   checkBottom.setText("Reinforcement Bottom");  checkBottom.setToolTipText("");  checkBottom.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  updataBox(evt);  }  });   checkSeltop.setText("Sealable Top");  checkSeltop.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  updataBox(evt);  }  });   Colour.setText("Colour");   jButton1.setText("Clear");  jButton1.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  jButton1ActionPerformed(evt);  }  });   BtnAdd.setText("Add");  BtnAdd.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  BtnAddActionPerformed(evt);  }  });   lblPrices.setText("jLabel4");   txtTotlePrice.setText("Total Price");   spinQuantity.setModel(new javax.swing.SpinnerNumberModel(1, 1, 900, 1));  spinQuantity.addChangeListener(new javax.swing.event.ChangeListener() {  public void stateChanged(javax.swing.event.ChangeEvent evt) {  updataBox2(evt);  }  });   jLabel6.setText("Quantity");  jLabel6.setToolTipText("");   jLabel7.setText("Grade");   spinColour.setModel(new javax.swing.SpinnerNumberModel(0, 0, 2, 1));   txtWidth.setText("0.1");  txtWidth.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  txtWidthActionPerformed(evt);  }  });  txtWidth.addKeyListener(new java.awt.event.KeyAdapter() {  public void keyTyped(java.awt.event.KeyEvent evt) {  txtUpdate(evt);  }  });   txtLength.setText("0.1");  txtLength.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  txtWidthActionPerformed(evt);  }  });  txtLength.addKeyListener(new java.awt.event.KeyAdapter() {  public void keyTyped(java.awt.event.KeyEvent evt) {  txtUpdate(evt);  }  });   txtHeigth.setText("0.1");  txtHeigth.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  txtWidthActionPerformed(evt);  }  });  txtHeigth.addKeyListener(new java.awt.event.KeyAdapter() {  public void keyTyped(java.awt.event.KeyEvent evt) {  txtUpdate(evt);  }  });   jComboBox1.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Grade 1", "Grade 2", "Grade 3", "Grade 4", "Grade 5" }));  jComboBox1.addItemListener(new java.awt.event.ItemListener() {  public void itemStateChanged(java.awt.event.ItemEvent evt) {  cmdUpdae(evt);  }  });   jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("ordares"));   txtInvoice.setEditable(false);  txtInvoice.setColumns(20);  txtInvoice.setRows(5);  txtInvoice.setEnabled(false);  jScrollPane1.setViewportView(txtInvoice);   javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);  jPanel2.setLayout(jPanel2Layout);  jPanel2Layout.setHorizontalGroup(  jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel2Layout.createSequentialGroup()  .addContainerGap()  .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 361, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addContainerGap(20, Short.MAX\_VALUE))  );  jPanel2Layout.setVerticalGroup(  jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel2Layout.createSequentialGroup()  .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 435, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGap(0, 21, Short.MAX\_VALUE))  );   javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  jPanel1.setLayout(jPanel1Layout);  jPanel1Layout.setHorizontalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(txtTotlePrice)  .addGroup(jPanel1Layout.createSequentialGroup()  .addContainerGap()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(checkSeltop)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  .addComponent(Colour)  .addComponent(BtnAdd))  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(18, 18, 18)  .addComponent(jButton1))  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(4, 4, 4)  .addComponent(spinColour, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))))  .addComponent(lblPrices)  .addComponent(jLabel7)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(75, 75, 75)  .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGroup(jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  .addComponent(jLabel2)  .addComponent(jLabel3)  .addComponent(jLabel1)  .addComponent(jLabel6))  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(10, 10, 10)  .addComponent(spinQuantity, javax.swing.GroupLayout.PREFERRED\_SIZE, 126, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(txtWidth, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGap(24, 24, 24))))  .addGroup(jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  .addComponent(checkConres)  .addComponent(txtHeigth, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addComponent(txtLength, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGap(18, 18, 18)  .addComponent(checkBottom)))))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addContainerGap())  );  jPanel1Layout.setVerticalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  .addGap(3, 3, 3)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(jLabel6)  .addComponent(spinQuantity, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGap(1, 1, 1)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(jLabel2)  .addComponent(txtWidth, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(jLabel3)  .addComponent(txtHeigth, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(jLabel1)  .addComponent(txtLength, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(jLabel7)  .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGap(24, 24, 24)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(checkConres)  .addComponent(checkBottom))  .addGap(4, 4, 4)  .addComponent(checkSeltop)  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(Colour)  .addComponent(spinColour, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGap(18, 18, 18)  .addComponent(lblPrices)  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(jButton1)  .addComponent(BtnAdd))  .addGap(4, 4, 4)  .addComponent(txtTotlePrice))  .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  .addContainerGap())  );   jLabel2.getAccessibleContext().setAccessibleName("height");   javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  getContentPane().setLayout(layout);  layout.setHorizontalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  );  layout.setVerticalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  );   jPanel1.getAccessibleContext().setAccessibleName("Make a Box");   pack();  }// </editor-fold>//GEN-END:initComponents   /\*\*  \*update form ActionEvent in check boxes   \* @param evt event from update  \*/  private void updataBox(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_updataBox  boxUpdateer();  }//GEN-LAST:event\_updataBox  /\*\*  \*update form ChangeEvent in spinQuantity  \* @param evt event from update  \*/  private void updataBox2(javax.swing.event.ChangeEvent evt) {//GEN-FIRST:event\_updataBox2  boxUpdateer();  }//GEN-LAST:event\_updataBox2    /\*\*  \*allows the user to add to order form button press  \* @param evt event from update  \*/  private void BtnAddActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_BtnAddActionPerformed  Box box =boxUpdateer() ;  if (box != null)  {  boxOrder.SetBox(box);  BigDecimal value = new BigDecimal(boxOrder.GetPrices());  value = value.setScale(2, BigDecimal.ROUND\_HALF\_UP);  txtTotlePrice.setText("totle prices £"+ value);  txtInvoice.setText(boxOrder.GetShowoffs());  }      }//GEN-LAST:event\_BtnAddActionPerformed   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton1ActionPerformed  // TODO add your handling code here:   }//GEN-LAST:event\_jButton1ActionPerformed    /\*\*  \*update form ActionEvent in Text box (only after hiting enter)  \* @param evt event from update  \*/  private void txtWidthActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_txtWidthActionPerformed  boxUpdateer();  }//GEN-LAST:event\_txtWidthActionPerformed    /\*\*  \*update form ActionEvent in Text box  \* @param evt event from update  \*/  private void txtUpdate(java.awt.event.KeyEvent evt) {//GEN-FIRST:event\_txtUpdate boxUpdateer();  }//GEN-LAST:event\_txtUpdate    /\*\*  \*update form ItemEvent in Combobox  \* @param evt event from update  \*/  private void cmdUpdae(java.awt.event.ItemEvent evt) {//GEN-FIRST:event\_cmdUpdae    System.out.println("not top one");  switch(jComboBox1.getSelectedItem().toString())  {  case "Grade 1":  grade = 1 ;  break;  case "Grade 2":  grade = 2 ;  break;  case "Grade 3":  grade = 3 ;  break;  case "Grade 4":  grade = 4 ;  break;  case "Grade 5":  grade = 5 ;  break;  }  boxUpdateer();  }//GEN-LAST:event\_cmdUpdae      /\*\*  \*vaid user input after it has bean inputed and update the ui  \* @return a vaid box  \*/  private Box boxUpdateer()  {  int Quantity = (Integer) spinQuantity.getValue();  double length;  double heigth;  double width;  try   {  length = Double.parseDouble(txtLength.getText());  heigth = Double.parseDouble(txtHeigth.getText());  width = Double.parseDouble(txtWidth.getText());  }  catch (Exception e)  {  lblPrices.setText("invaided BOX");  return null;  }  if (length < 0 ||heigth < 0 || width < 0) {  lblPrices.setText("invaided BOX");  return null;    }    if (length >= 900 ||heigth >= 900 || width >= 900) {  lblPrices.setText("invaided BOX");  return null;    }  boolean Bottom = checkBottom.isSelected();  boolean Conres = checkConres.isSelected();  boolean seltop = checkSeltop.isSelected();  int NumerOfColur = (Integer) spinColour.getValue();  Box box = boxOrder.MakeBox(NumerOfColur, Bottom, Conres, seltop, grade, length, heigth, width,Quantity);  try  {  BigDecimal value = new BigDecimal(box.price());  value = value.setScale(2, BigDecimal.ROUND\_HALF\_UP);  lblPrices.setText("this box £"+value);  }  catch (Exception e)  {  lblPrices.setText("invaided BOX");  }  return box;  }    // Variables declaration - do not modify//GEN-BEGIN:variables  private javax.swing.JButton BtnAdd;  private javax.swing.JLabel Colour;  private javax.swing.ButtonGroup buttonGroup1;  private javax.swing.JCheckBox checkBottom;  private javax.swing.JCheckBox checkConres;  private javax.swing.JCheckBox checkSeltop;  private javax.swing.JButton jButton1;  private javax.swing.JComboBox<String> jComboBox1;  private javax.swing.JLabel jLabel1;  private javax.swing.JLabel jLabel2;  private javax.swing.JLabel jLabel3;  private javax.swing.JLabel jLabel6;  private javax.swing.JLabel jLabel7;  private javax.swing.JPanel jPanel1;  private javax.swing.JPanel jPanel2;  private javax.swing.JScrollPane jScrollPane1;  private javax.swing.JLabel lblPrices;  private javax.swing.JSpinner spinColour;  private javax.swing.JSpinner spinQuantity;  private javax.swing.JTextField txtHeigth;  private javax.swing.JTextArea txtInvoice;  private javax.swing.JTextField txtLength;  private javax.swing.JLabel txtTotlePrice;  private javax.swing.JTextField txtWidth;  // End of variables declaration//GEN-END:variables } |

**ADPROC Coursework -**

Complete the Group Members’ Contribution to the ADPROC Coursework **below.**

This should cover the overall contribution of each group member to the coursework.

|  |
| --- |
| **Group Members’ Contribution to the ADPROC 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 four students Alpha, Beta, Gamma, and Delta, where all have contributed evenly, you would give 25% each.  However, if the contributions were uneven, you might mark them as: e.g., Alpha has done most of the work, so he/she gets 50%; Beta, Gamma and Delta have completed the rest of the work and between them Beta did 25%, Gamma did 15% and Delta - 10%.  List your group number and group members **by student number** and their scores below:  Group No (*e.g.,* ***GrD-7***): GrA-5  Student number and contribution:  1. \_UP822718\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          /100  2. \_UP918156\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          /100  3. \_UP879389\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /100  4. \_UP821837\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          /100  **TOTAL 100/100** |

ADPROC, Advanced Programming Concepts (U21266)

## 

## Coursework

**Hand out date: 24.X.2018 Hand in date: 7.XII.2018 (Demonstration: by week starting 3rd 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 12 – week starting 3rd December, in your lab session) **you have to submit a memory stick**  with your source code and Java NetBeans project files with **your group number** 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 producing variety of boxes for packaging wide range of goods. Due to the wide range of requirements of their customers, the variety of boxes the *FlexBox* has to produce is very extensive.

The boxes are all rectangular and have the following characteristics:

* They are all made of cardboard;
* The cardboard 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.

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Grade of cardboard | 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 |



The types of boxes, produced by the company, are shown in Table1 and the costs of 1m2 of cardboard are given in Table2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cardboard Grade | 1 | 2 | 3 | 4 | 5 |
| Cost per m2 [in £] | 0.55 | 0.65 | 0.82 | 0.98 | 1.5 |

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

|  |  |
| --- | --- |
| 1 colour | 12% extra |
| 2 colours | 15% extra |
| Reinforced bottom | 13% extra |
| Reinforced corners | 12% extra |
| Sealable tops | 10% extra |

There are some additional costs depending on whether the box has printing and if there is any box reinforcement. These are shown in Table 3 and the percentage increases **are all applied using 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 cardboard;
* 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 the requested box 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 (using Table 2 and Table 3) and quoted.

The customer should be able to place several orders in one session, in which case the total cost should be prompted.

**Customers should not be asked for the type of box they want** (since this is only used within the company to calculate the cost). **It is your application that must determine** (using Table 1) **the box type, based on the ordered box attributes.**

Customers should be able to get a quote for as many pipes (of different types) as they like (within the capacity of *FlexBox*) in the same order. In such cases, the total cost of the order should be calculated and displayed.

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

## Your Task

* Write an application, which will allow the customers to enter the details of their order and subsequently prompts the cost of the order. Your application should verify that *FlexBox* can supply the type of the requested boxes (the customer should not be asked to specify the box type).
* Use OO design approach (abstraction, inheritance, aggregation, and polymorphism) and create a class hierarchy that describes the types of boxes *FlexBox* sells. Use an abstract class as well.
* 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 the coursework!
* 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 memory stick** (with **your group number on it**) with your source code and Java NetBeans project files of your software no later than week12 (week starting **3.XII.2018**), during your lab session.

On **7.XII.2018** your group should submit electronically (**by 6pm**) to Moodle a **.pdf** file with your **report. The file name should be your group name** (e.g., ***GrC-2.pdf***, or ***GrA-3.pdf***, or ***GrD-5.pdf***, etc.). **Only one file per group** should be submitted(decide in advance who is going to submit it). Your report **should be no more than 6 pages (excluding the appendices) and should include** the following:

* **A UML** use case diagram of your order system, UML class hierarchy diagram of your OO application design, and also one UML class diagram (one class of your choice), 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 (no more than one page) and screen shots to evidence the testing and evaluation;
* **The source** code that you have written as 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** with your individual contributions;
* **This document**.

The assessment criteria and allocated marks 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 of 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 |  |