

[UP777611 UP790487 UP781212 UP788777]

[Advanced Programming Concepts (U21266)]

[2017]

Contents

Introduction: page 2

* Front end page 2
* Back end page 2

Assumptions: page 3

Limitations: page 3

Unified Modelling Language (UML): page 4

* Use Case Diagram page 4 - 5
* Hierarchy Diagram page 6
* Class Diagram page 7
* Instance Diagram page 8 - 9

Application: page 10

Features: page 11 - 13

Sample input and output: page 14 - 17

Testing and Screenshots: page 18 - 48

Testing schedule: page 49

Appendix: page 50

* Source code page 50 - 98
* Group Contribution Form page 99
* Assignment Brief page 100 - 104

**Introduction:**

Within this document we will be talking about the application we were asked to build for LongPipe alongside any assumptions that we had to make in order to build it, shortly after that we will also be discussing the limitations of the implementation of the actual application itself.

A basic order placing system that might be used by a Pipe selling company (that manufactures various kinds of Pipes) was implemented as per the requirements of the coursework.

LongPipes Company manufactures a variety of pipes for use in heating, plumbing, drainage, conduit, and conveyance of fuel and gas. The pipes are made of plastic of specified grade. The pipes can have zero to three colours. Depending on the application, the pipes may have outer metallic reinforcement and improved chemical resistance.  They are sold in length not exceeding 6 meters. Customers have to specify the length and diameter of the pipes when placing a new order. The proposed system aims at developing an order system for the company’s sales department.

**Front End:**

It consists of a customer-facing front-end GUI consisting of a form where the user is required to fill in the specifications of the Pipe that the customer needs to purchase along with the quantities and then place the order with the click of a button. If the order is successful (meaning the company can supply Pipes of the demanded specification), then the order is added as a row to a table along with the cost of the order. If not, the user is informed that his specification is not available. The user may continue to place as many orders as he/she wishes.

**Back end:**

The back-end of the system implements an application modeled using object oriented programming principles. Since the company can produce a number of different types of Pipes, those attributes that are independent of the specific pipe type are compressed in an abstract Pipe class. The various (concrete) types of pipe (Pipe 1 through Pipe 5) are derived from this abstract class. Each pipe type has certain specifications (minimum and maximum plastics grade, number of colours, whether it has an inner insulation and outer reinforcement).

**Assumptions:**

The first assumption is that the company makes standard pipes of 6 meters in length for the various diameters. Upon receiving an order for pipes of a particular length and diameter, the company will first issue pipes of 6 meters in length. Only when the order requires pipes of less than 6 meters will the shorter pieces be used first. The system will give priority to the 6-meter pipes and resort to the smaller pieces to fill up any additional units needed to match the customer requirements.

The second assumption is that customers place an order of a minimum 1-meter pipe and maximum of 4-inches. Customers cannot order a pipe that is less than 1 meter long but can order any unit longer than 1.0 meter.

Finally the third assumption is that the company will round the order to the nearest 2 decimal places per pipe ordered. So for example a type 4 pipe of 1m length and 1in diameter would be 6.47 instead of 6.4742…. etc.

**Limitations:**

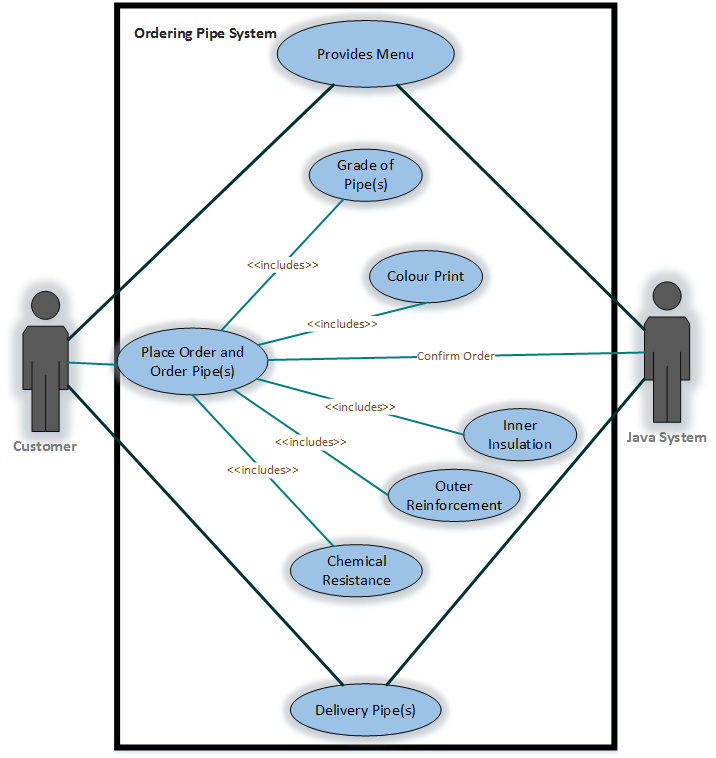
There are a few limitations when it comes to the application that we have built for LongPipe, one of which is the fact that:

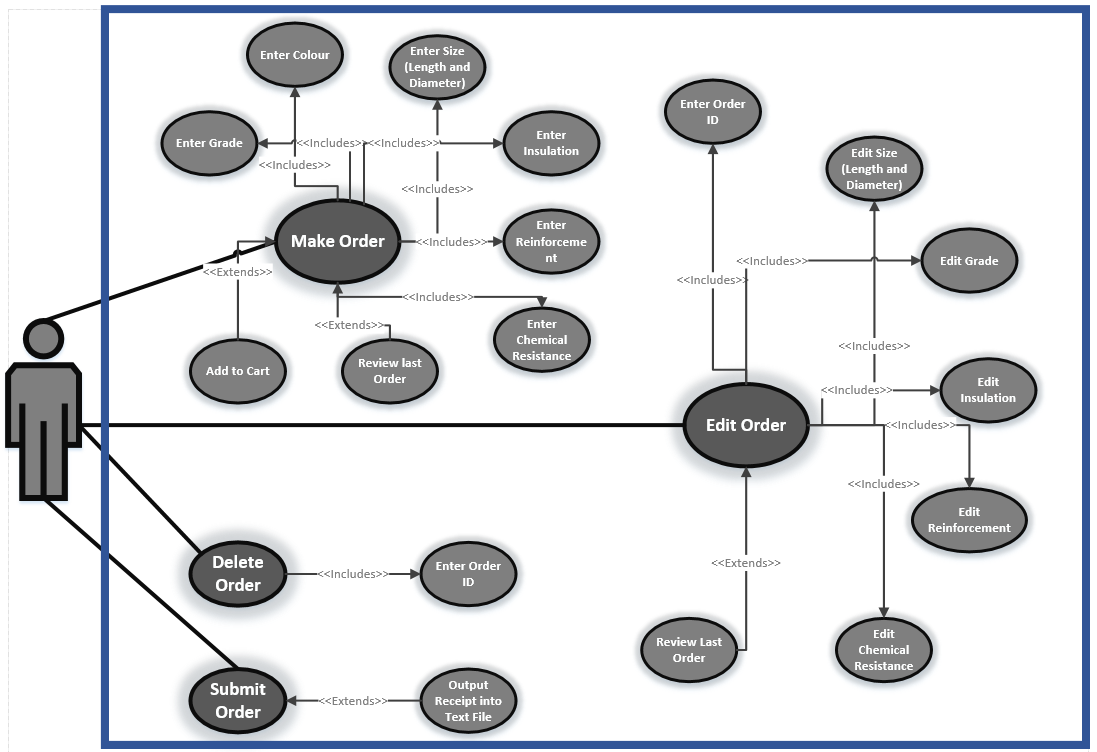
* The difficulties that we faced throughout the implementation was getting the different cost to calculate for each part.
* Creating ‘Use Case’ was hard to associate between actor (customer) and the action in order.
* The software will only be functional in a desktop.
* The maximum amount per order is 100 pipes (there is no cap on the overall order amount however to avoid potential human error a cap of 100 pipes per order was added)
* The pipe size is limited to:
  + Length is invalid: must be between 0 and 6 meters.
  + Diameter in invalid: must be between 0 and 4 inches.

Obviously some of these limitations (such as pipe size limit and order limit) are things that if LongPipe is not happy with we can easily change the software code and recompile it in order to rectify the issue.

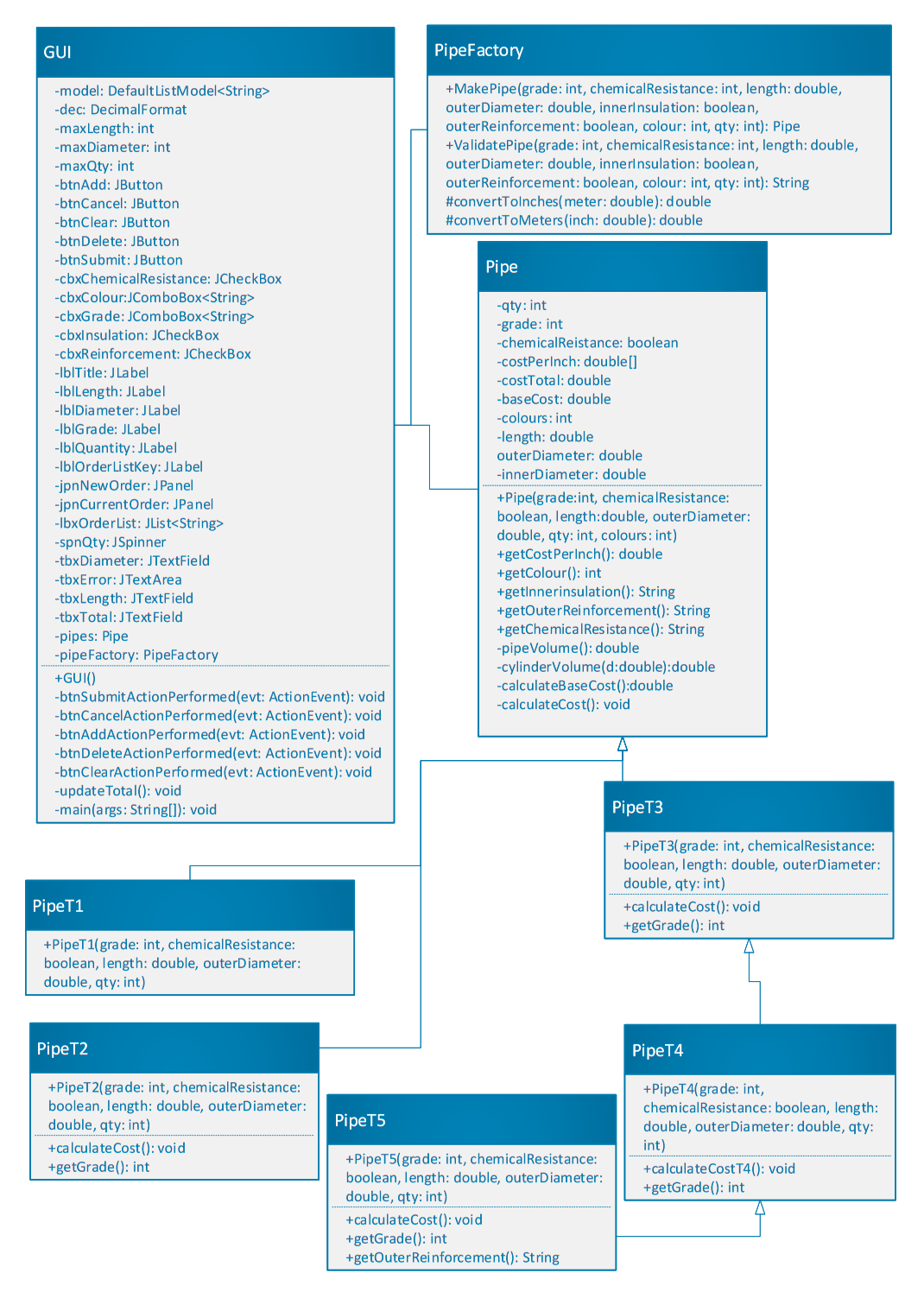
Unified Modelling Language (UML)

Use Case:

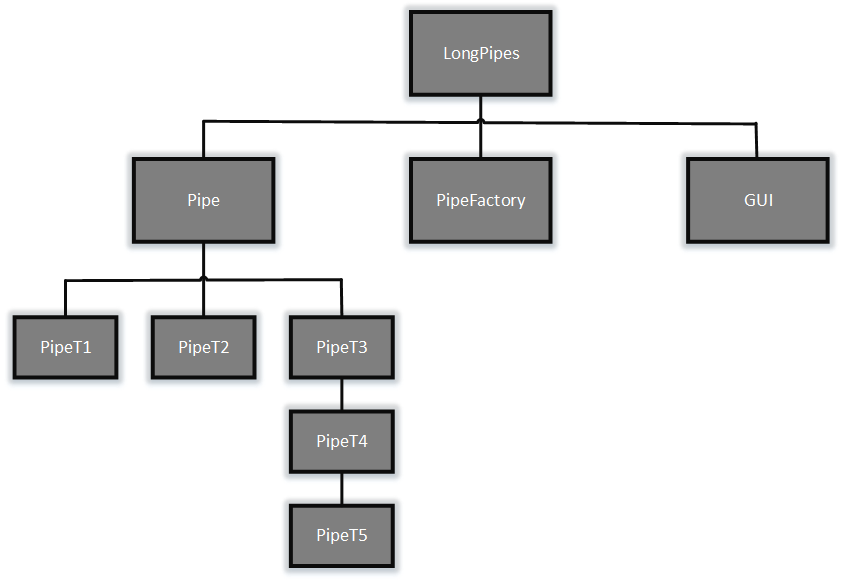




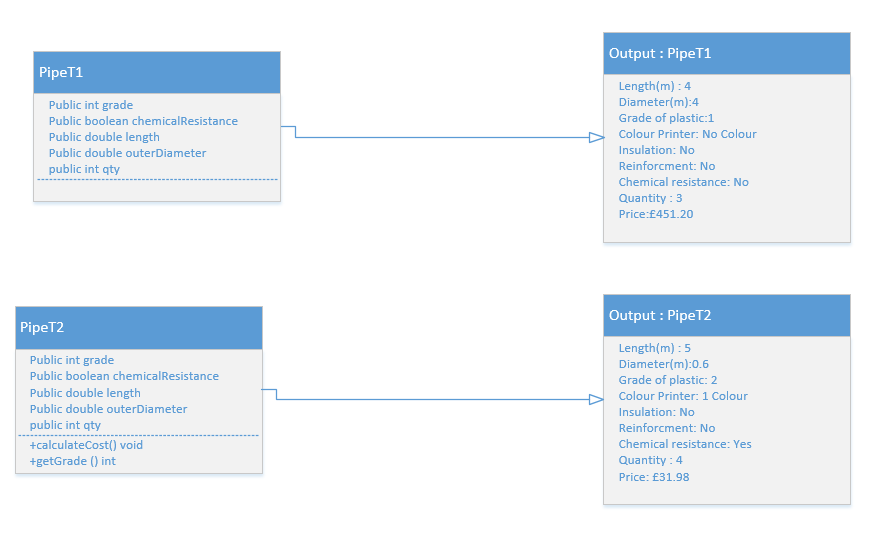
Hierarchy Diagram:

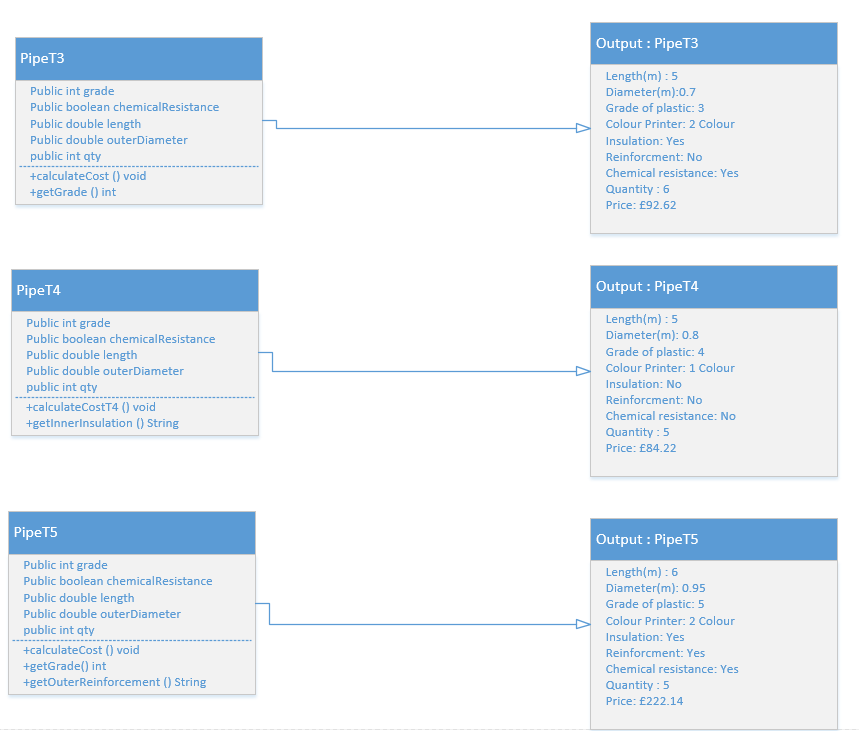


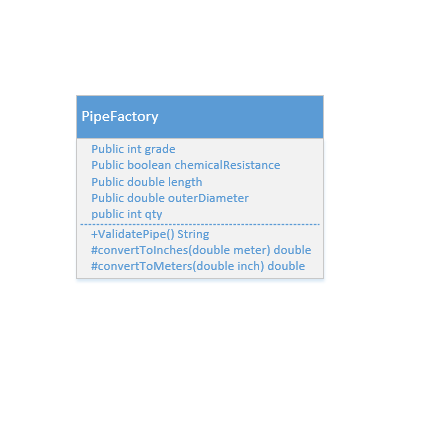
Class Diagram:



Instance Diagram:

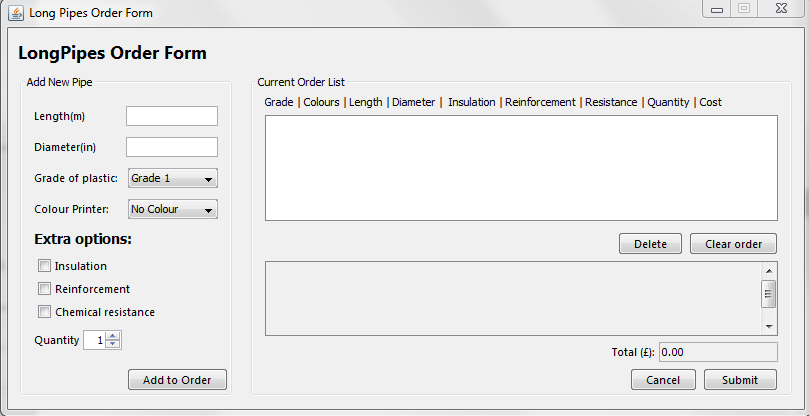






**The Application**

The overall application ended up being all in one form as shown on the screenshot below.



**The application is split into 3 general segments:**

* Input for the pipe dimensions and input for the options which the box can have (such as the quality of the box) in this segment you also have the “add to order” and edit order options.
* **Current order list:** Customer can view their order they made.
* **Confirmation box:** This area is to show the customer that their order have been submit. Also customer can confirm the purchase and delete any order they do not want it.

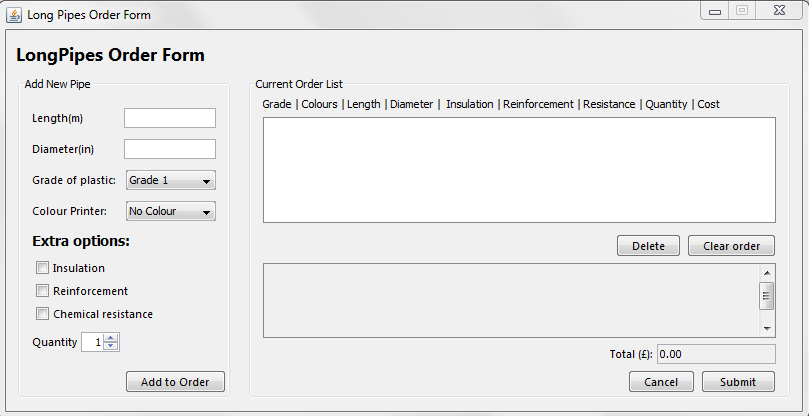
**Application Functionalities**

* The application takes the dimension of the pipe entered by the customer and will calculate the area of that box in order to calculate the price per meter/inches. (Requirement)
* The application will check which options the customer has entered and it determines what of pipe it is and check if LongPipe supplies such pipe (as shown to us by LongPipe for example, Type 1 pipe). **(Requirement)**
* The application will calculate the overall price of the pipes (which is based on the area, the grade and what type of pipe is it) and then multiply it by the quantity wanted by the customer and show the customer the overall price of the order. **(Requirement)**
* Allow for multiple orders to be placed. **(Requirement)**
* Application can edit an order placed by the customer **(Feature)**
* Application can delete an order placed by the customer **(Feature)**

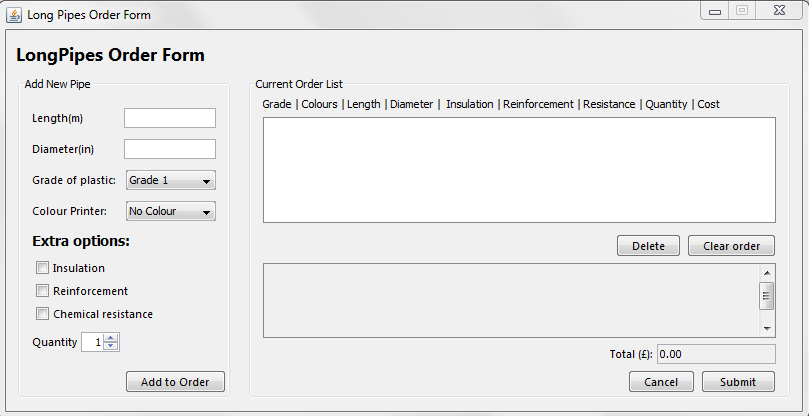
**Features**

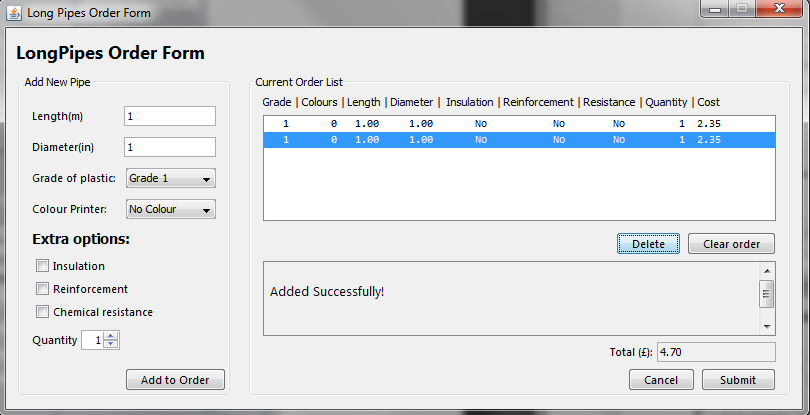
The application has a few features that were added in order to make a few “quality of life improvements” to anyone who uses this application, such features are listed below:

* **Being able to edit an order:** this allows for customers to edit current orders in the receipt area. In order to use this customers will need to input the dimension of the pipe alongside the options of the pipe (grade, quantity, colour etc.), and also enter the number of the order they wish to delete.

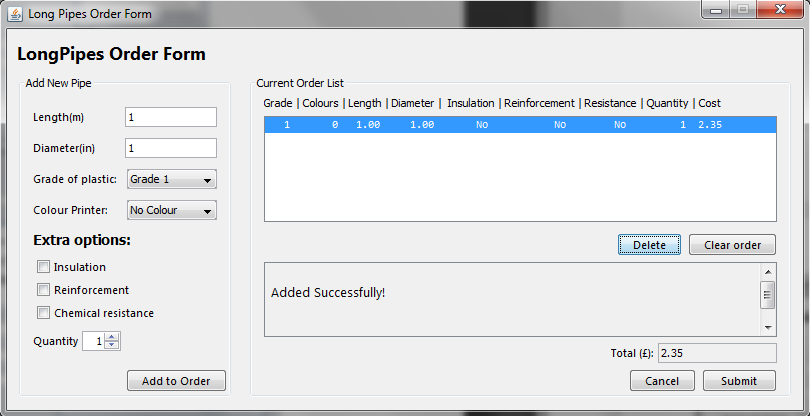


* **Being able to delete an order:** this allows for customers to delete current orders in their receipt area. In order to use this the customer will need to enter the order number they wish to delete and then click on the delete button.



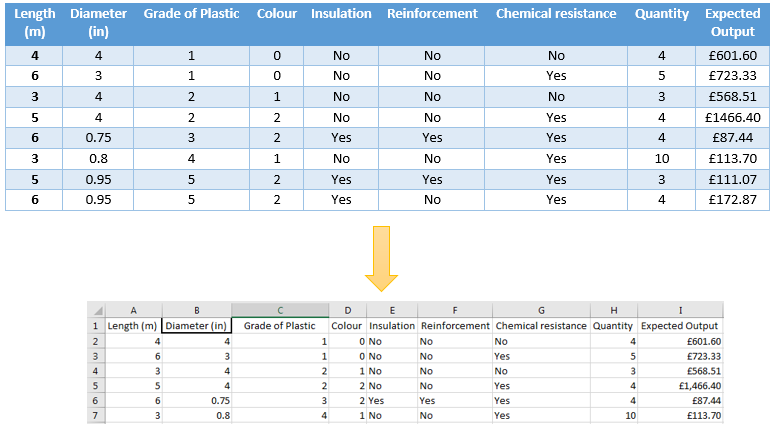


* **Create a message and total cost for the order:** this just creates message and total cost for the placed by the customer once the submit button has been clicked in order to finalise the order

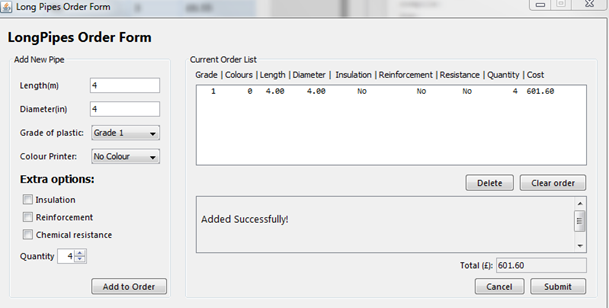


**Sample Input and outputs**

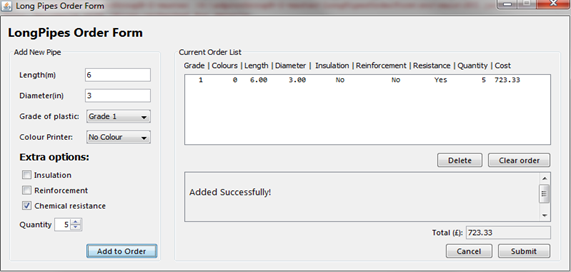
We used the spreadsheet shown below to test the output of the system based on certain inputs, of this list we have shown screenshot below we took a smaller sample in order to prove the system’s functionality.



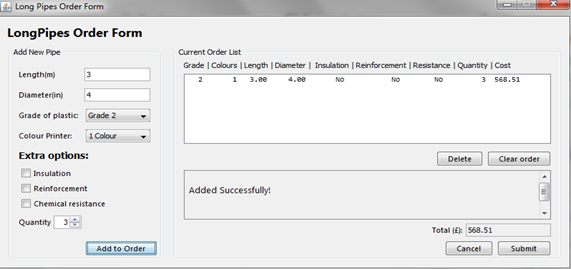
**Row2:**

****

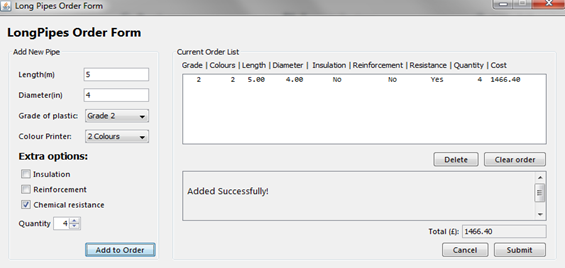
**Row3:**



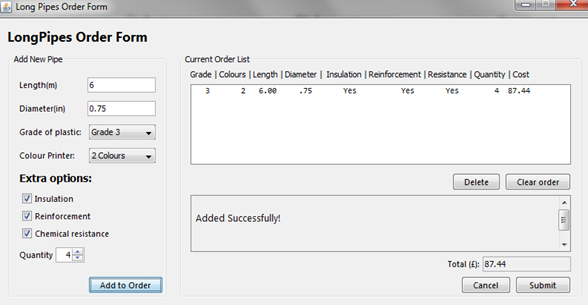
**Row4:**



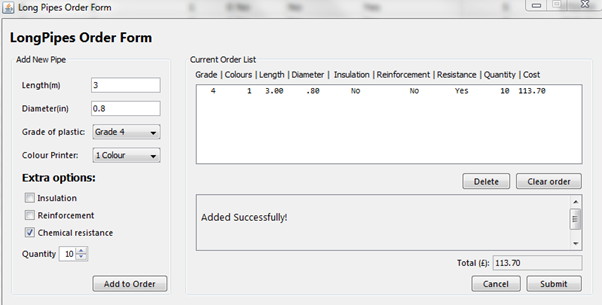
**Row5**:



**Row6:**



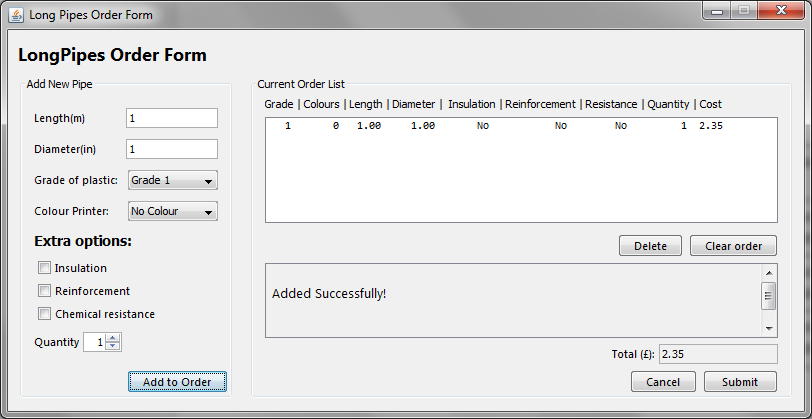
**Row7**:



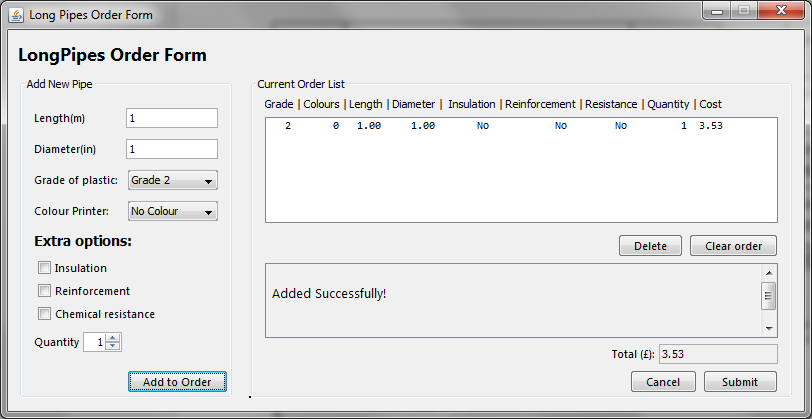
Testing & Screenshots

Test I - Pipe of Type 1:

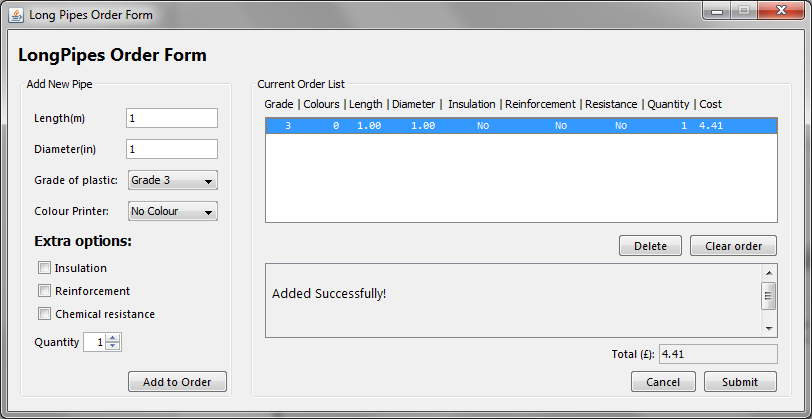
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| I1 | Type 1 pipe created using grade 1 plastic, length and diameter of 1 meter and 1 inches. | Value will be 2.35 | Pass | Not Applicable |
| I2 | Type 1 pipe created using grade 2 plastic, length and diameter of 1 meter and 1 inches. | Value will be 3.53 | Pass | Not Applicable |
| I3 | Type 1 pipe created using grade 3 plastic, length and diameter of 1 meter and 1 inches. | Value will be  4.41 | Pass | Not Applicable |



I1 correct output



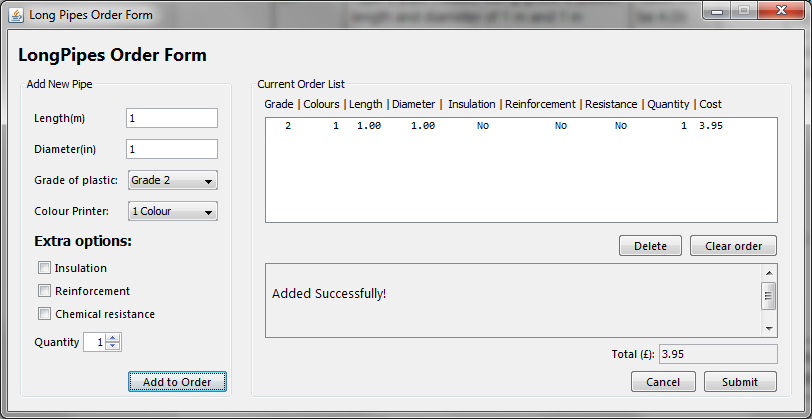
I2 correct output



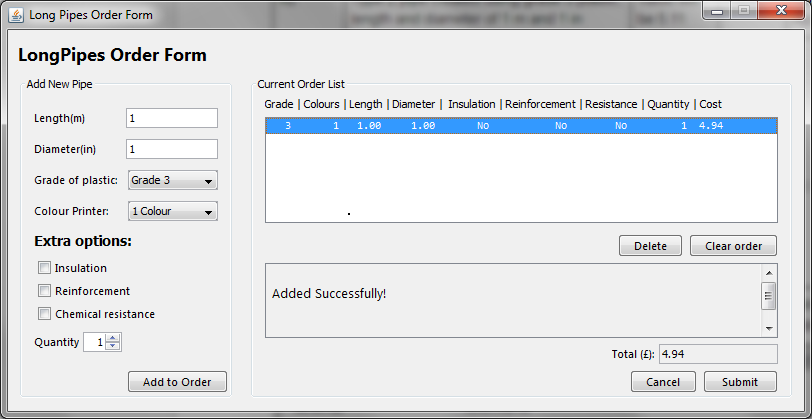
I3 correct output

Test II - Pipe of Type 2:

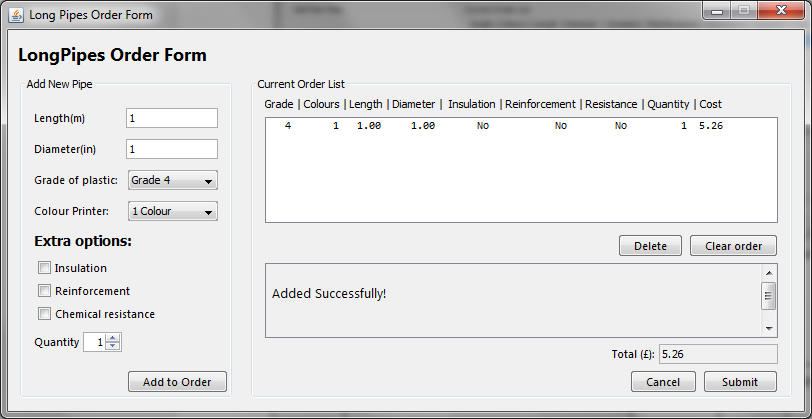
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| II1 | Type 2 pipe created using grade 2 plastic, length and diameter of 1 meter and 1 inches. | Value will be 3.95 | Pass | Not Applicable |
| II2 | Type 2 pipe created using grade 3 plastic, length and diameter of 1 meter and 1 inches. | Value will be 4.94 | Pass | Not Applicable |
| II3 | Type 2 pipe created using grade 4 plastic, length and diameter of 1 meter and 1 inches. | Value will be  5.26 | Pass | Not Applicable |



II1 correct output



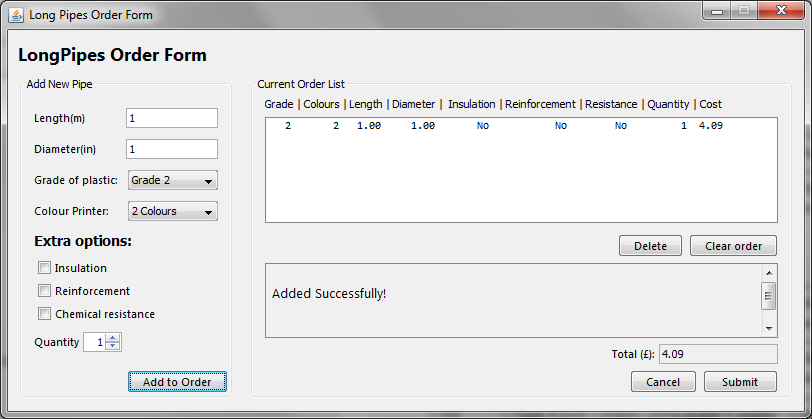
II2 correct output



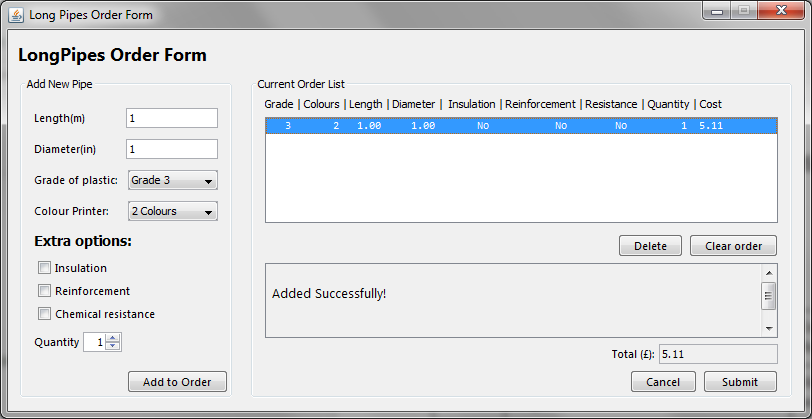
II3 correct output

Test III - Pipe of Type 3:

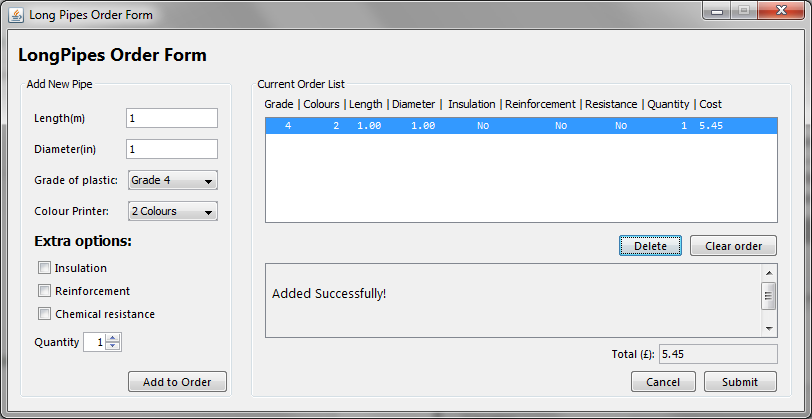
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| III1 | Type 3 pipe created using grade 2 plastic, length and diameter of 1 meter and 1 inches. | Value will be 4.09 | Pass | Not Applicable |
| III2 | Type 3 pipe created using grade 3 plastic, length and diameter of 1 meter and 1 inches. | Value will be 5.11 | Pass | Not Applicable |
| III3 | Type 3 pipe created using grade 4 plastic, length and diameter of meter and 1 inches. | Value will be  5.45 | Pass | Not Applicable |
| III4 | Type 3 pipe created using grade 5 plastic, length and diameter of meter and 1 inches. | Value will be 6.47 | Pass | Not Applicable |



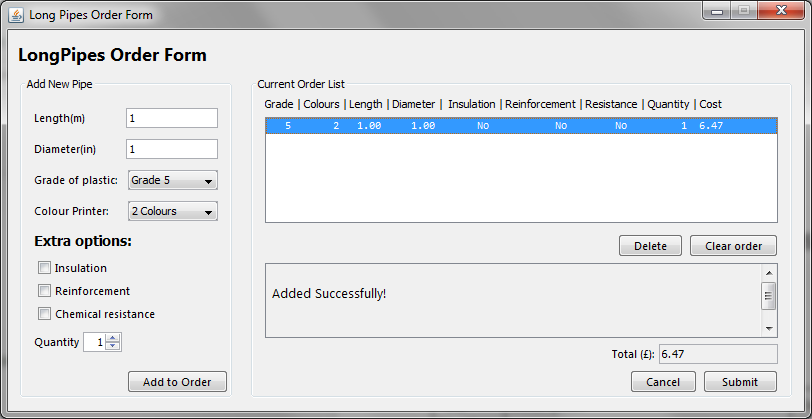
III1 correct output



III2 correct output



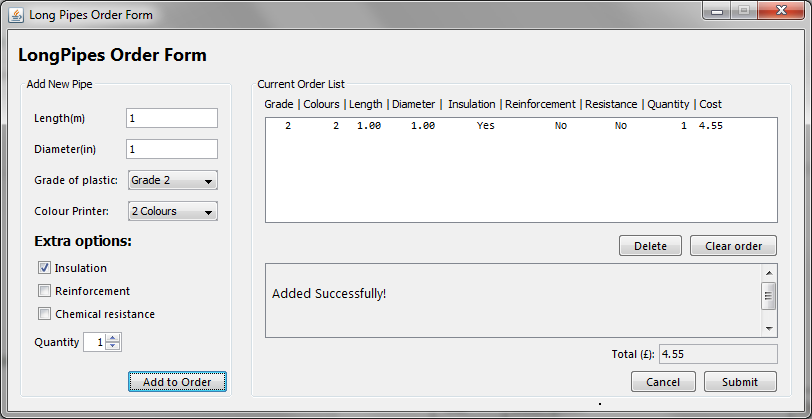
III3 correct output



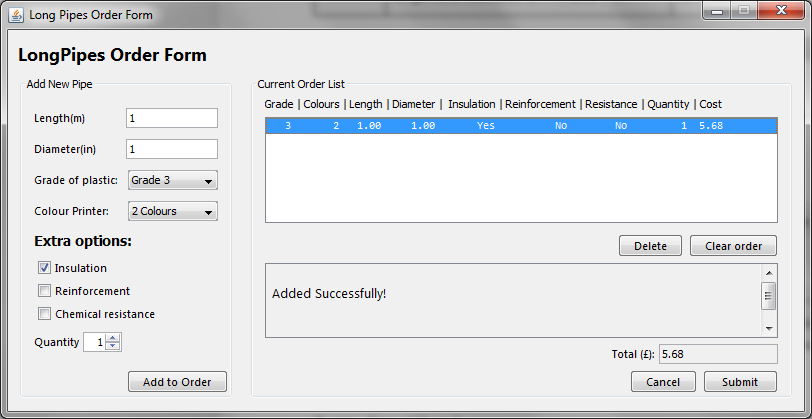
III4 correct output

Test IV - Pipe of Type 4:

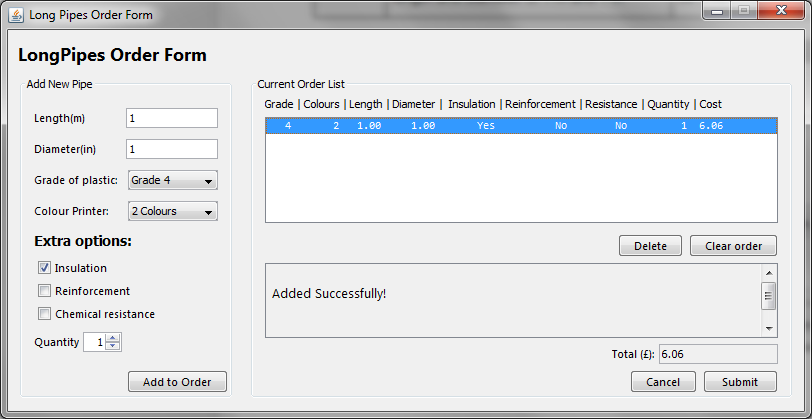
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| IV1 | Type 4 pipe created using grade 2 plastic, length and diameter of 1 meter and 1 inches. | Value will be 4.09 | Pass | Not Applicable |
| IV2 | Type 4 pipe created using grade 3 plastic, length and diameter of meter and 1 inches. | Value will be 5.11 | Pass | Not Applicable |
| IV3 | Type 4 pipe created using grade 4 plastic, length and diameter of meter and 1 inches. | Value will be  5.45 | Pass | Not Applicable |
| IV4 | Type 4 pipe created using grade 5 plastic, length and diameter of meter and 1 inches. | Value will be 6.47 | Pass | Not Applicable |



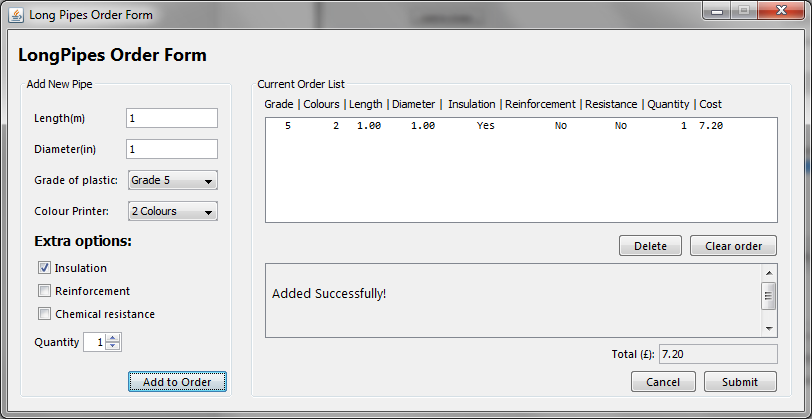
IV1 correct output



IV2 correct output



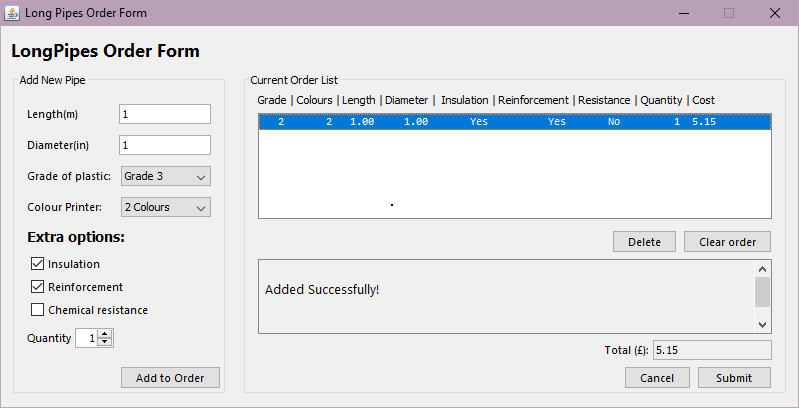
IV3 correct output



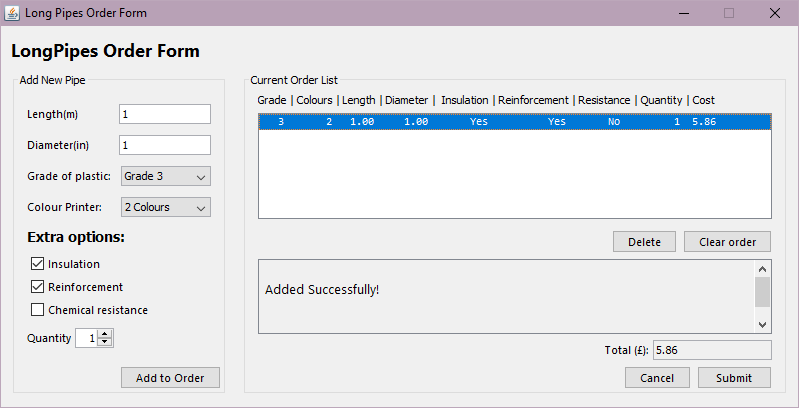
IV4 correct output

Test V - Pipe of Type 5:

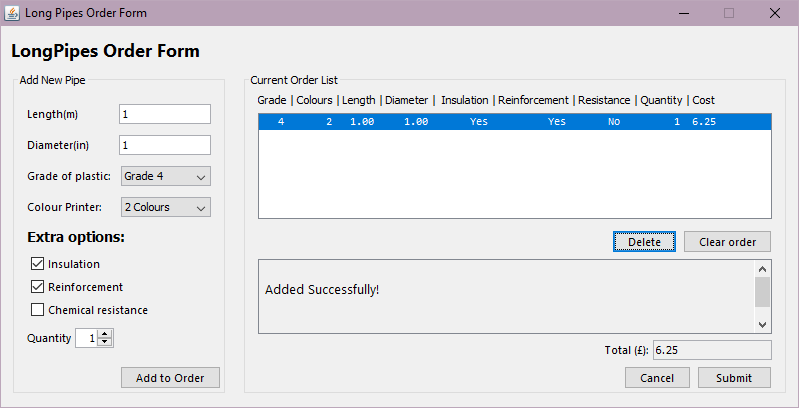
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| V1 | Type 5 pipe created using grade 3 plastic, length and diameter of 1 meter and 1 inches. | Value will be 5.86 | Fail | Edit the T5 class to implement an override to T3 class |
| V1.1 | Type 5 pipe created using grade 3 plastic, length and diameter of meter and 1 inches. | Value will be 5.86 | Pass | Not Applicable |
| V2 | Type 5 pipe created using grade 4 plastic, length and diameter of meter and 1 inches. | Value will be  6.25 | Pass | Not Applicable |
| V3 | Type 5 pipe created using grade 3 plastic, length and diameter of meter and 1 inches. | Value will be 7.42 | Pass | Not Applicable |



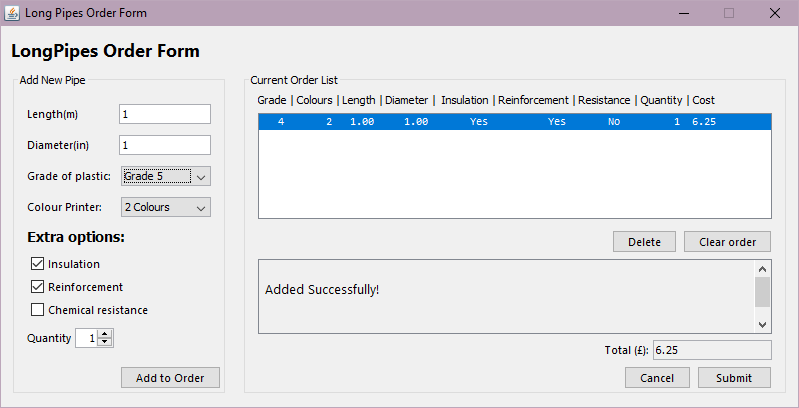
V1 incorrect output.



V1 correct output



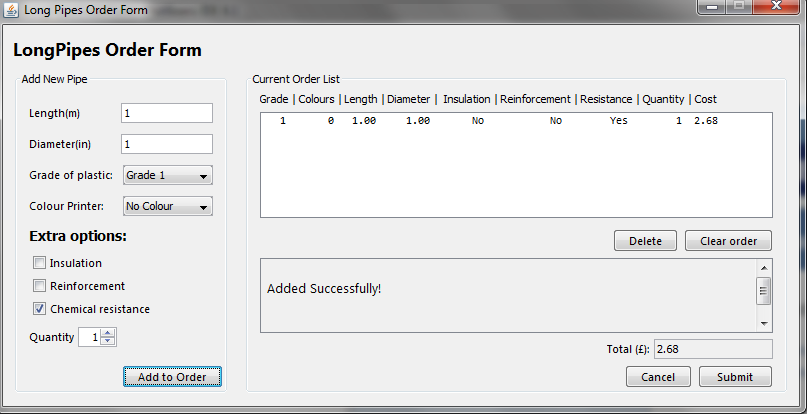
V2 correct output



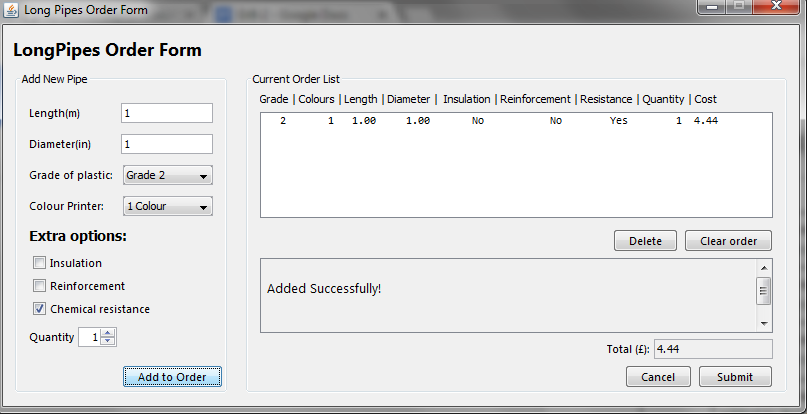
V3 correct output

Test VI - Pipe with Chemical Resistance:

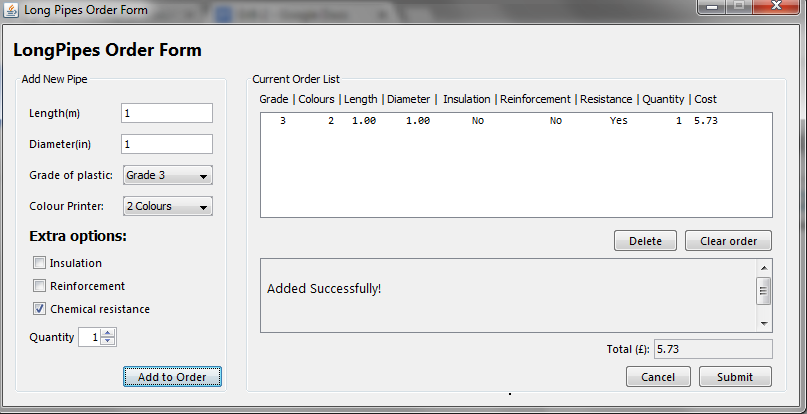
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| VI1 | Type 1 pipe with chemical resistance added to the order | Value will be 2.68 | Fail | Not Applicable |
| VI2 | Type 2 pipe with chemical resistance added to the order | Value will be 4.44 | Pass | Not Applicable |
| VI3 | Type 3 pipe with chemical resistance added to the order | Value will be  5.73 | Pass | Not Applicable |
| VI4 | Type 4 pipe with chemical resistance added to the order | Value will be 6.30 | Pass | Not Applicable |
| VI5 | Type 5 pipe with chemical resistance added to the order | Value will be 6.48 | Pass | Not Applicable |



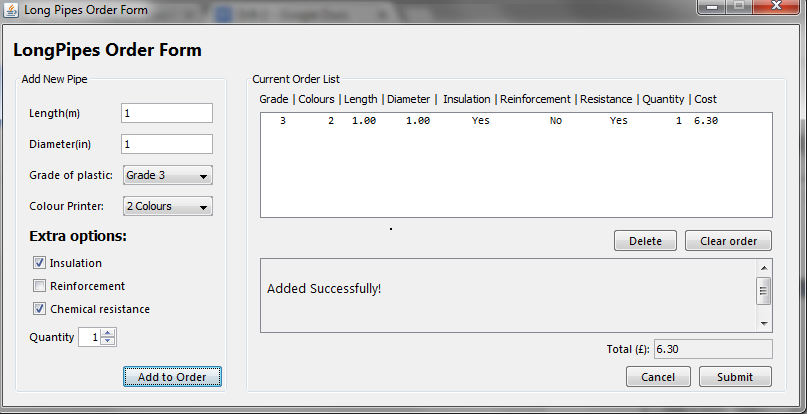
VI1 correct output



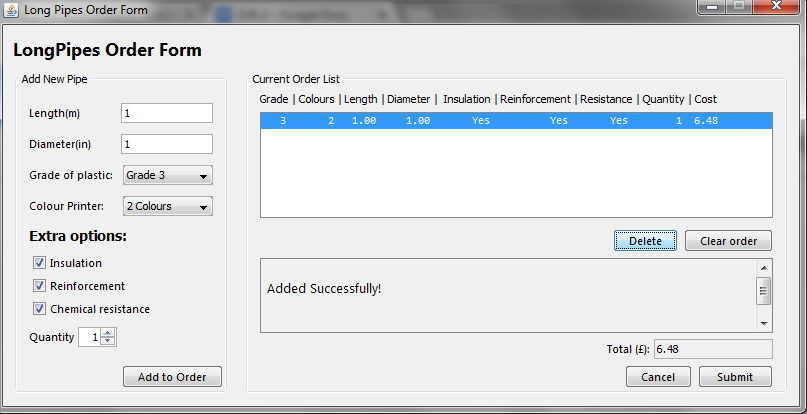
VI2 correct output



VI3 correct output



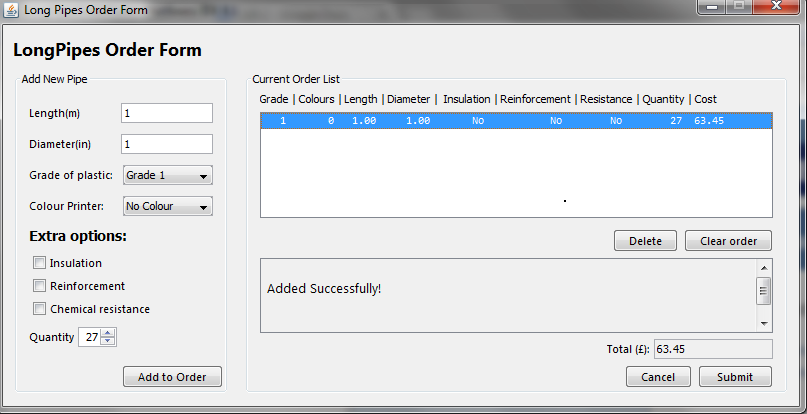
VI4 correct output



VI5 correct output

Test VII - More than 1 quantity of pipe:

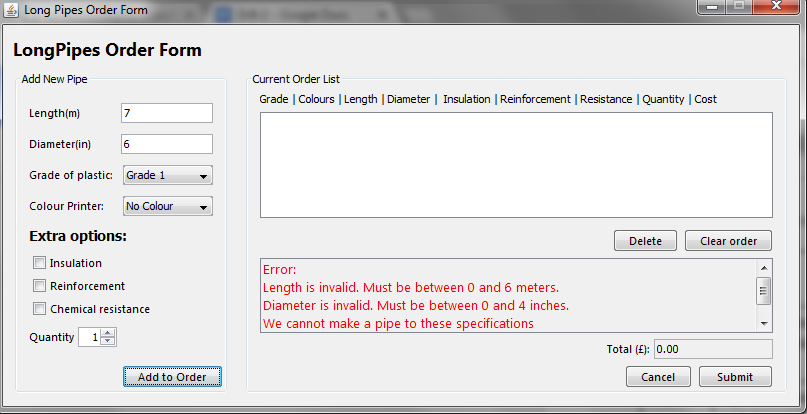
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| VII1 | Add multiple of a pipe at one time | Multiple pipes can be added in a single click of ‘add to order’ | Fail | Not Applicable |



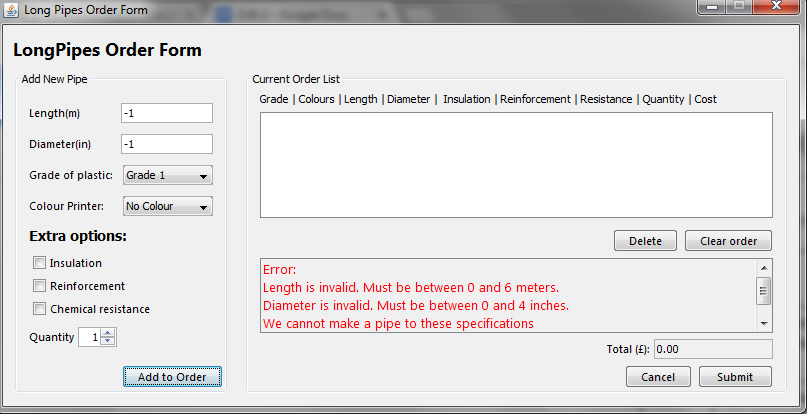
VII1 correct output

Test VII - Pipe with an incorrect diameter or length:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| VII1 | Pipe exceeding maximum length and diameter | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VII2 | Pipe exceeding minimum length and diameter | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |



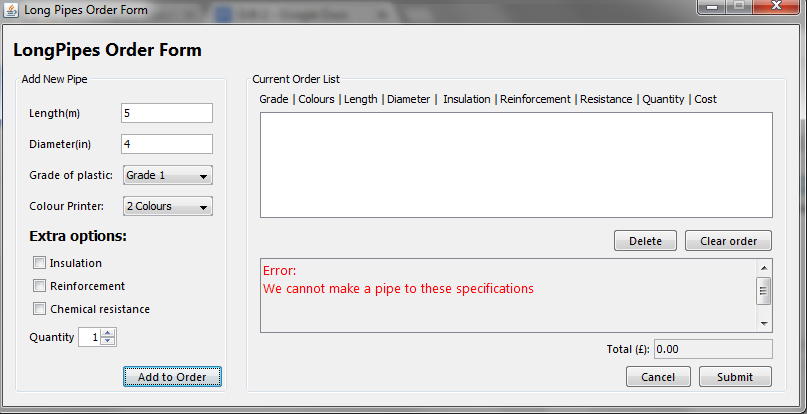
VII1 correct output



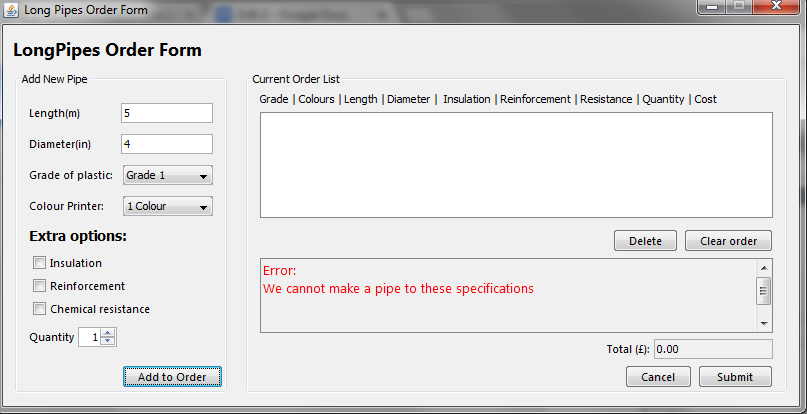
VII correct output

Test VIII - Pipe type cannot use that set of colours:

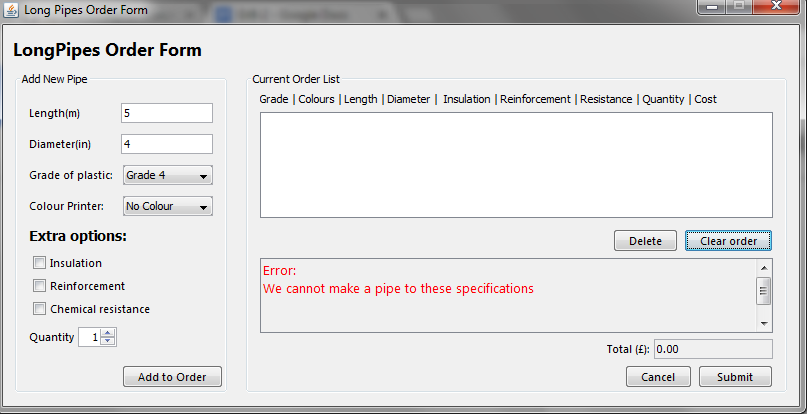
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| VIII1 | Grade 1 pipe using 2 colours | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VIII2 | Grade 1 pipe using 1 colour | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VIII3 | Grade 2 pipe using 0 colours | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VIII4 | Grade 2 pipe using 2 colours | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VIII5 | Grade 3 pipe using 1 colours | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VIII6 | Grade 3 pipe using 0 colour | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VIII7 | Grade 4 pipe using 0 colour | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VIII8 | Grade 4 pipe using 1 colours | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VIII9 | Grade 5 pipe using 1 colour | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |
| VIII10 | Grade 5 pipe using 0 colours | Red error text will appear notifying user of invalid pipe. | Pass | Not Applicable |



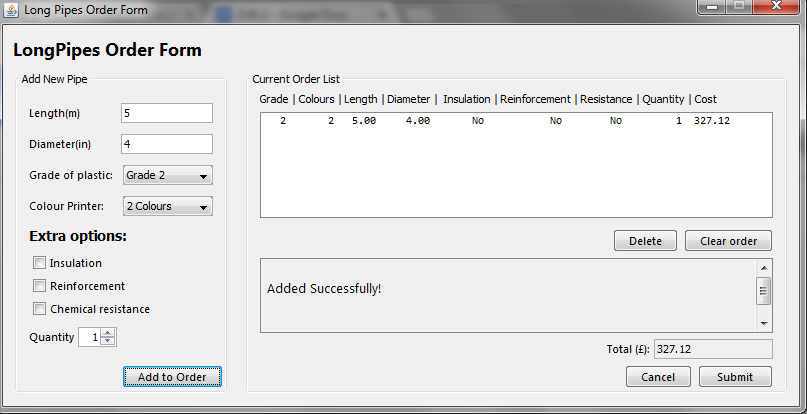
VIII1 correct output



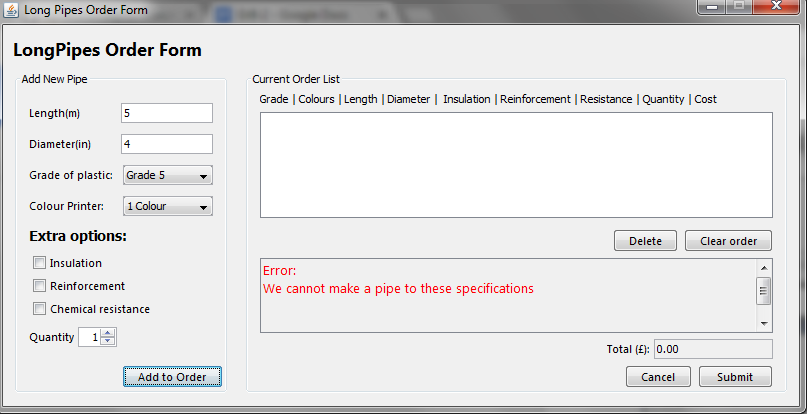
VIII2 correct output



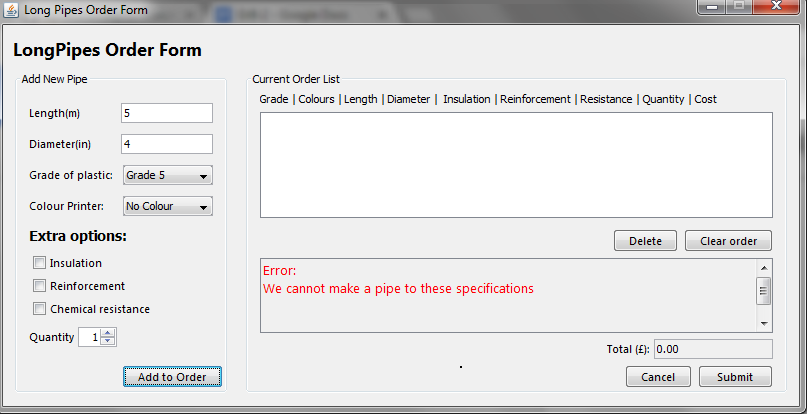
VIII3 correct output



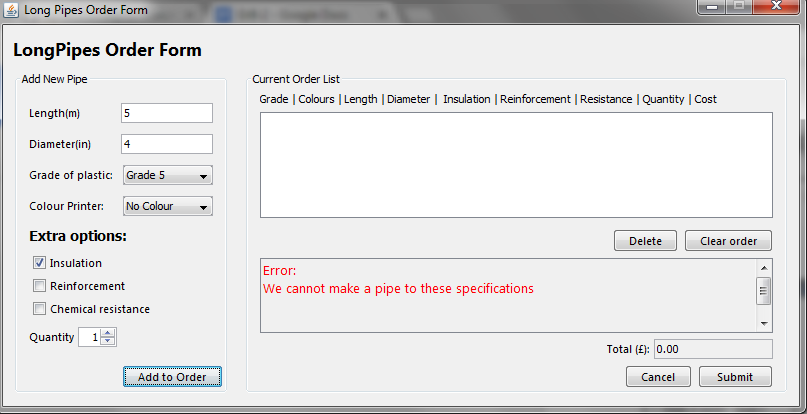
VIII4 correctly adds as a type 3 and not type 2 pipe



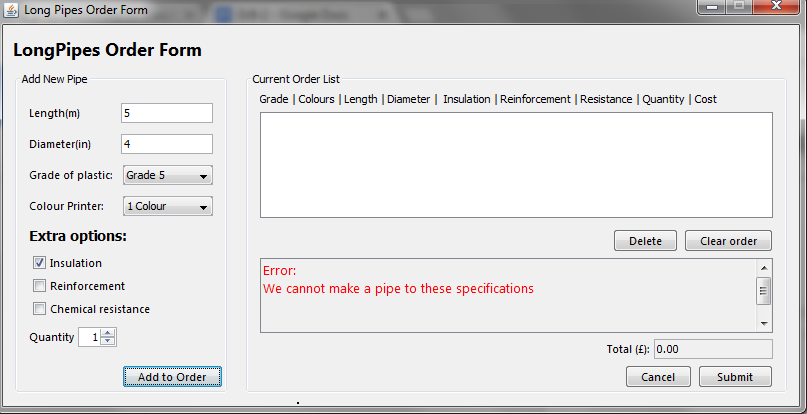
VIII5 correct output



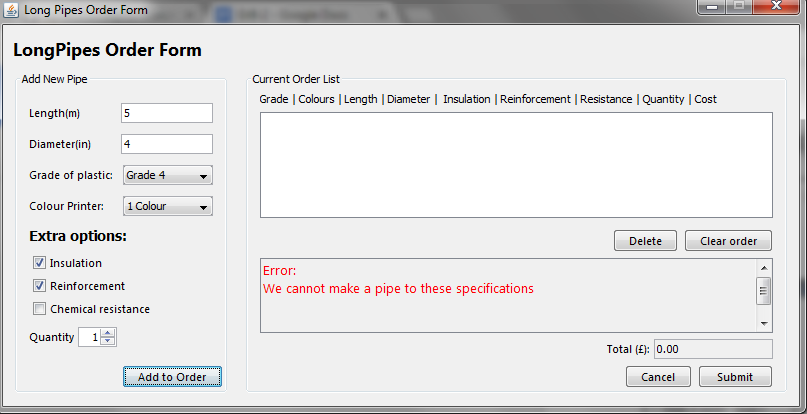
VIII6 correct output



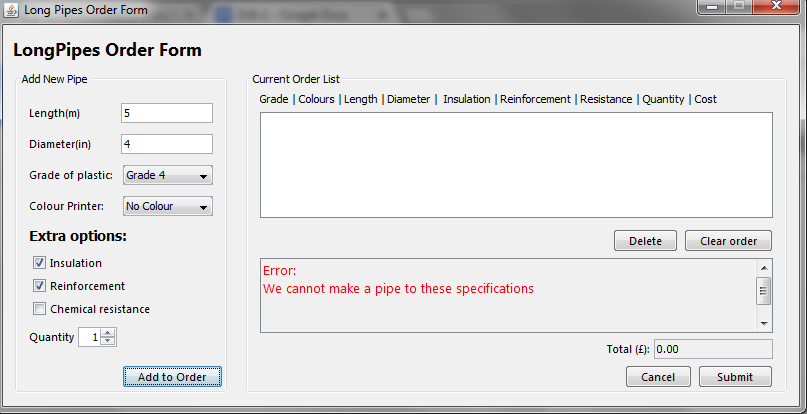
VIII7 correct output



VIII8 correct output



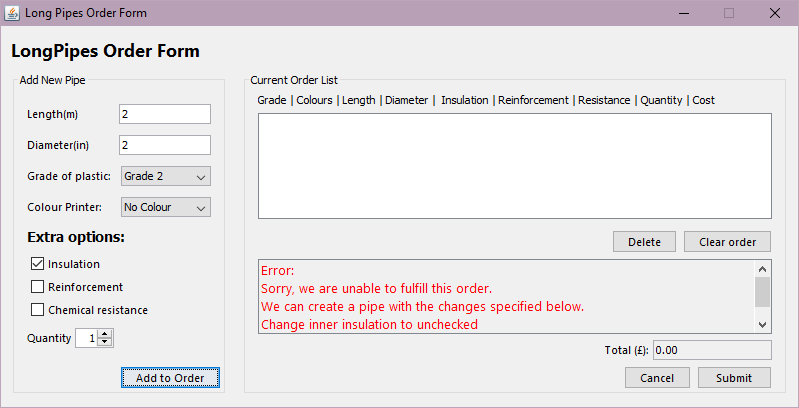
VIII9 correct output



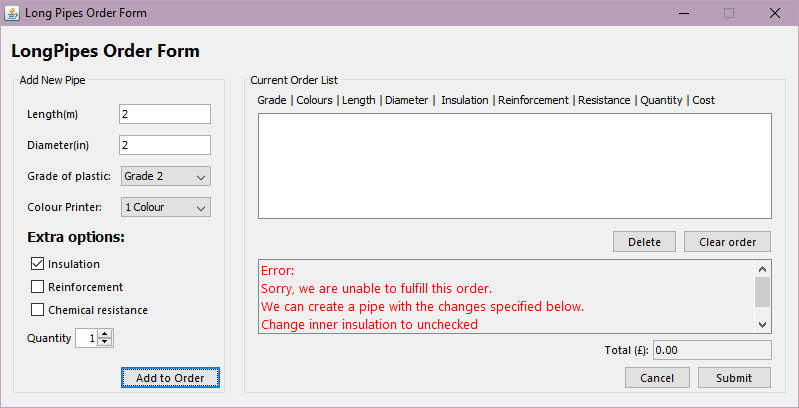
VIII10 correct output

Test IX - Pipe Type cannot use insulation

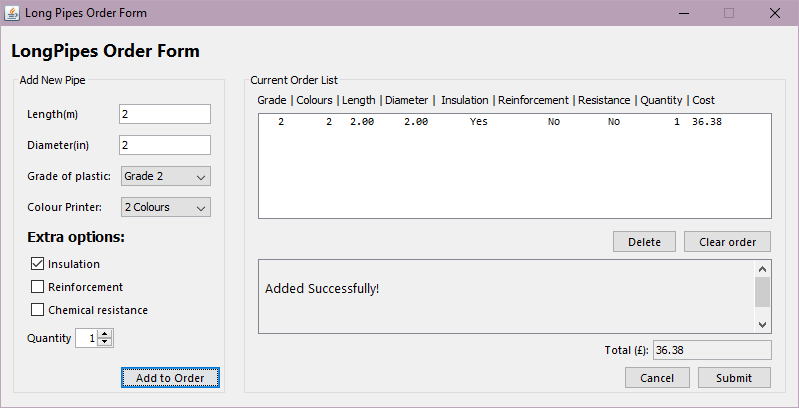
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| IX1 | Type 1 pipe using reinforcement | Error will be returned | Pass | Not Applicable |
| IX2 | Type 2 pipe using reinforcement | Error will be returned | Pass | Not Applicable |
| IX3 | Type 3 pipe using reinforcement | Pipe will become type 4 | Pass | Not Applicable |



Test X1 result



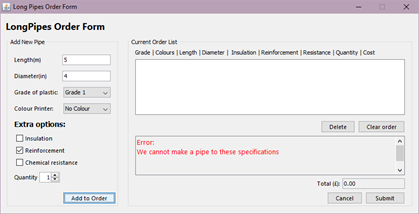
Test X2 result



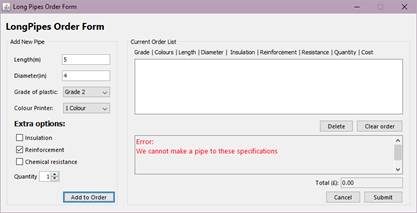
Test X3 result

Test X - Pipe Type cannot use reinforcement:

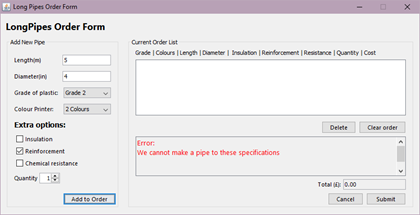
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| X1 | Grade 1 using all criteria for grade plus reinforcement and ‘Add to Order’ pressed | Error will be returned | Pass | Not Applicable |
| X2 | Grade 2 using all criteria for grade plus reinforcement and ‘Add to Order’ pressed | Error will be returned | Pass | Not Applicable |
| X3 | Grade 3 using all criteria for grade plus reinforcement and ‘Add to Order’ pressed | Error will be returned | Pass | Not Applicable |
| X4 | Grade 3 using all criteria for grade plus reinforcement and ‘Add to Order’ pressed | Error will be returned | Pass | Not applicable |



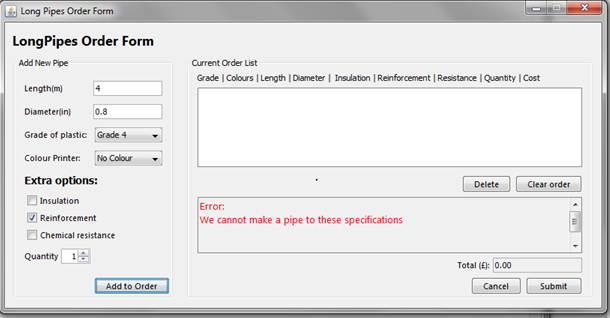
Test X1 result



Test X2 Result



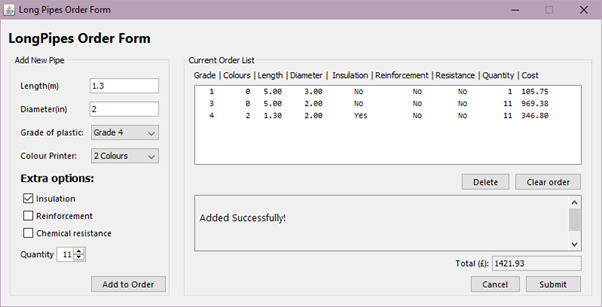
Test X3 Result



Test X4 Result

Test XI - Multiple Purchases can be made in one session:

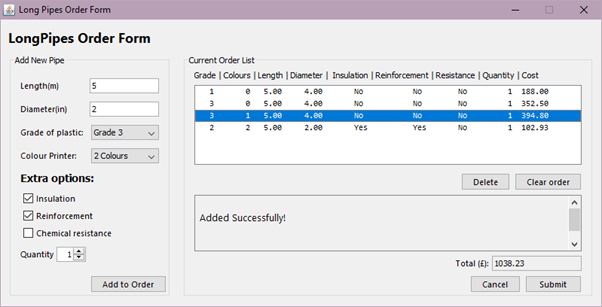
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| XI | Multiple pipes will be added to the order. | Total will sum the values of all items added multiplied by the quantity. | PASS | Not Applicable |



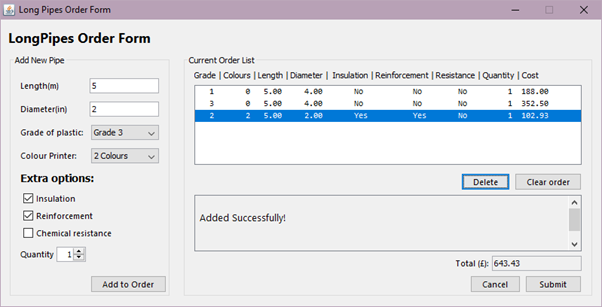
Text XI1 result.

Test XII - items can be removed:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| XII | Item can be deleted from current order by selecting the item and pressing ‘Delete’ | Item will be removed and the total updated | PASS | Not Applicable |



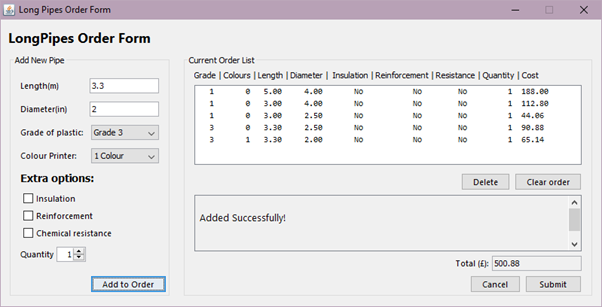
XII before removing the item highlighted



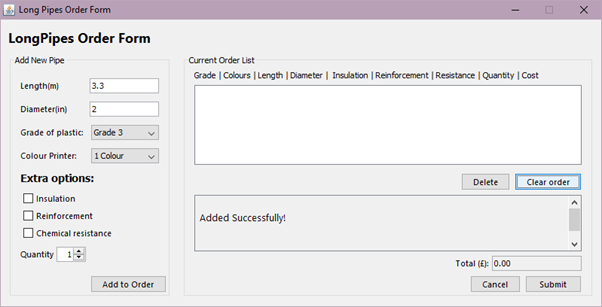
XII after removing the item previously highlighted

Test XIII – Clear order button functions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Description | Expected Result | Pass / Fail | Resolution |
| XIII | Press ‘Clear order’ button | The list will be cleared and the total set to 0 | PASS | Not Applicable |



XIII before pressing clear order



XIII after pressing clear order

**Testing schedule**

Within this document we will be going through several tests that we have applied to our application that we built for LongPipe in order to ensure that not only does the application run but also to ensure that it meets the functional requirements that the company has out prior to us starting designing it.

In order to ensure that the software works as required we are going to enter a wide range of data of which will belong to one of the types specified below.

* **Valid data:** This is what is usually expected to be entered within the application in order to achieve a desired outcome.
* **Valid extreme:** This will be data which is unusual, meaning that it will test the boundaries of the system, however the data ensured should still be valid.
* **Invalid data:** This is data which should fail.
* **Invalid extreme:** This is data which should be on the verge of being almost valid, so once again testing the boundaries of the system.
* **Erroneous data:** This is data which would not be expected to be entered, an example of such would be entering a characters when expecting a digit.

The types of data mentioned above are going to be used within several testing methods which we have devised to do:

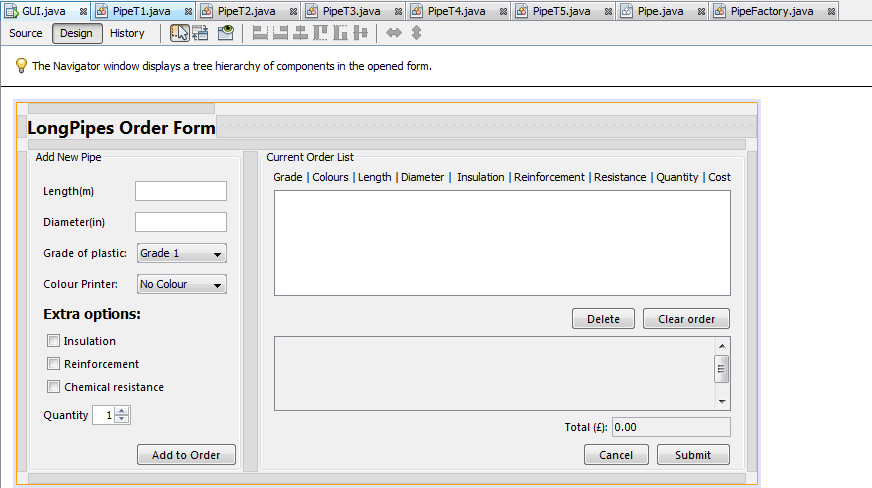
* **System testing:** This will be done against the requirements set out by the customer in in order to verify if the application does what they need it to do.

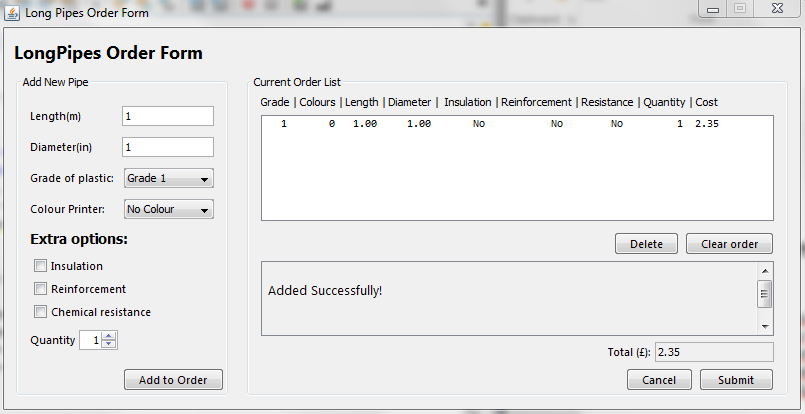
The main points we are going to test:

* Inputs into the system **(Validation)**
* Outputs of the system **(Error messages and customer orders)**
* Price calculation of pipes ordered

Appendix

**Main class:**





**Source code from HUI**

package main;

import java.util.ArrayList;

import java.awt.Color;

import java.text.DecimalFormat;

import javax.swing.DefaultListModel;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author 781212, 788777, 777611, 790487

\*/

public class GUI extends javax.swing.JFrame {

//Store all pipes made by user.

private ArrayList<Pipe> pipes = new ArrayList();

//model used to store displayed list of ordered items.

DefaultListModel<String> model = new DefaultListModel<String>();

//Used for creating and validating pipes.

private PipeFactory pipeFactory = new PipeFactory();

//To make sure everything looks tidy!(adds formatting)

private DecimalFormat dec = new DecimalFormat("#.00");

//Set maximum limits on quantity, width and height. (100, 6m, 4in respectively)

private final int maxLength = 6, maxDiameter = 4, maxQty = 100;

/\*\*

\* Creates new form GUI

\*/

public GUI() {

initComponents();

}

/\*\*

\* 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">

private void initComponents() {

jScrollPane1 = new javax.swing.JScrollPane();

jLabel1 = new javax.swing.JLabel();

jPanel1 = new javax.swing.JPanel();

jLabel3 = new javax.swing.JLabel();

tbxDiameter = new javax.swing.JTextField();

tbxLength = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();

cbxGrade = new javax.swing.JComboBox<>();

jLabel7 = new javax.swing.JLabel();

cbxColour = new javax.swing.JComboBox<>();

cbxInsulation = new javax.swing.JCheckBox();

cbxReinforcement = new javax.swing.JCheckBox();

cbxChemicalResistance = new javax.swing.JCheckBox();

jLabel11 = new javax.swing.JLabel();

jLabel12 = new javax.swing.JLabel();

btnAdd = new javax.swing.JButton();

spnQty = new javax.swing.JSpinner();

jPanel2 = new javax.swing.JPanel();

btnClear = new javax.swing.JButton();

jScrollPane4 = new javax.swing.JScrollPane();

lbxOrderList = new javax.swing.JList<>();

btnDelete = new javax.swing.JButton();

btnSubmit = new javax.swing.JButton();

btnCancel = new javax.swing.JButton();

tbxTotal = new javax.swing.JTextField();

jLabel14 = new javax.swing.JLabel();

jScrollPane2 = new javax.swing.JScrollPane();

tbxError = new javax.swing.JTextArea();

jLabel4 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle("Long Pipes Order Form");

setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));

setResizable(false);

jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

jLabel1.setText("LongPipes Order Form");

jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Add New Pipe"));

jLabel3.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

jLabel3.setText("Length(m)");

jLabel2.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

jLabel2.setText("Diameter(in)");

jLabel5.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

jLabel5.setText("Grade of plastic:");

cbxGrade.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Grade 1", "Grade 2", "Grade 3", "Grade 4", "Grade 5" }));

jLabel7.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

jLabel7.setText("Colour Printer:");

cbxColour.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "No Colour", "1 Colour", "2 Colours" }));

cbxInsulation.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

cbxInsulation.setText("Insulation");

cbxReinforcement.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

cbxReinforcement.setText("Reinforcement");

cbxChemicalResistance.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

cbxChemicalResistance.setText("Chemical resistance");

cbxChemicalResistance.setToolTipText("");

cbxChemicalResistance.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

cbxChemicalResistanceActionPerformed(evt);

}

});

jLabel11.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

jLabel11.setText("Extra options:");

jLabel12.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

jLabel12.setText("Quantity");

btnAdd.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

btnAdd.setText("Add to Order");

btnAdd.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnAddActionPerformed(evt);

}

});

spnQty.setModel(new javax.swing.SpinnerNumberModel(1, 1, 100, 1));

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(btnAdd, javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(cbxReinforcement)

.addComponent(cbxInsulation)

.addComponent(cbxChemicalResistance)

.addComponent(jLabel11)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jLabel12)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(spnQty, javax.swing.GroupLayout.PREFERRED\_SIZE, 39, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel2)

.addComponent(jLabel3))

.addGap(30, 30, 30)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(tbxDiameter, javax.swing.GroupLayout.DEFAULT\_SIZE, 92, Short.MAX\_VALUE)

.addComponent(tbxLength)))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel7)

.addComponent(jLabel5))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(cbxGrade, 0, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(cbxColour, 0, 90, Short.MAX\_VALUE))))

.addContainerGap())))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(15, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel3)

.addComponent(tbxLength, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(tbxDiameter, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(cbxGrade, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel5))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(cbxColour, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel7))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel11)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(cbxInsulation)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cbxReinforcement)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cbxChemicalResistance)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel12)

.addComponent(spnQty, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addComponent(btnAdd))

);

jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Current Order List"));

jPanel2.setToolTipText("");

btnClear.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

btnClear.setText("Clear order");

btnClear.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnClearActionPerformed(evt);

}

});

lbxOrderList.setFont(new java.awt.Font("Consolas", 0, 11)); // NOI18N

lbxOrderList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE\_SELECTION);

lbxOrderList.setToolTipText("");

jScrollPane4.setViewportView(lbxOrderList);

btnDelete.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

btnDelete.setText("Delete");

btnDelete.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnDeleteActionPerformed(evt);

}

});

btnSubmit.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

btnSubmit.setText("Submit");

btnSubmit.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnSubmitActionPerformed(evt);

}

});

btnCancel.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

btnCancel.setText("Cancel");

btnCancel.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnCancelActionPerformed(evt);

}

});

tbxTotal.setEditable(false);

tbxTotal.setText("0.00");

jLabel14.setFont(new java.awt.Font("Segoe UI", 0, 11)); // NOI18N

jLabel14.setText("Total (£):");

tbxError.setEditable(false);

tbxError.setBackground(getBackground());

tbxError.setColumns(20);

tbxError.setFont(new java.awt.Font("Segoe UI", 0, 13)); // NOI18N

tbxError.setRows(5);

jScrollPane2.setViewportView(tbxError);

jLabel4.setText("Grade | Colours | Length | Diameter | Insulation | Reinforcement | Resistance | Quantity | Cost");

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()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jScrollPane2)

.addComponent(jScrollPane4)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addGap(0, 347, Short.MAX\_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addComponent(btnDelete)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(btnClear))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addComponent(btnCancel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(btnSubmit, javax.swing.GroupLayout.PREFERRED\_SIZE, 75, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addComponent(jLabel14)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(tbxTotal, javax.swing.GroupLayout.PREFERRED\_SIZE, 119, javax.swing.GroupLayout.PREFERRED\_SIZE))))

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jLabel4)

.addGap(0, 0, Short.MAX\_VALUE)))

.addContainerGap())

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(4, 4, 4)

.addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jScrollPane4, javax.swing.GroupLayout.PREFERRED\_SIZE, 106, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(btnClear)

.addComponent(btnDelete))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT\_SIZE, 75, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(tbxTotal, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel14))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(btnCancel)

.addComponent(btnSubmit)))

);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(10, 10, 10)

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(15, 15, 15)

.addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addContainerGap())

);

pack();

}// </editor-fold>

private void cbxChemicalResistanceActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

//pretend the order is submitted (beyond the scope of requirements)

private void btnSubmitActionPerformed(java.awt.event.ActionEvent evt) {

//show dialog saying order has been submitted (actual code for this outside the scope of coursework)

if (pipes.size() != 0) {

JOptionPane.showMessageDialog(null, "Thank you, your order has been submitted!", "Submitted", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(null, "Your order is empty, please add items to order!", "Error: Empty Order", JOptionPane.INFORMATION\_MESSAGE);

}

}

//when the cancel button is pressed, close the application on prompt

private void btnCancelActionPerformed(java.awt.event.ActionEvent evt) {

//confirm dialog asking the user if they want to quit.

int close = JOptionPane.showConfirmDialog(null, "Do you want to exit?", "Confirm", JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE);

if (close == JOptionPane.YES\_OPTION) {

this.dispose();

}

}

//When Adding a new pipe to the order:

private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {

//Set error variable to clear

tbxError.setText("");

String err = "";

byte grade = 0;

double length = 0;

double diameter = 0;

//check textboxes are valid input (non-empty and parse to double)

try {

length = Double.parseDouble(tbxLength.getText());

//check length

if (length > maxLength || length <= 0) {

err += "Length is invalid. Must be between 0 and 6 meters.\n";

} else {

}

} catch (NumberFormatException e) {

//Not a number

err += "Length is not a number. \n";

}

try {

//check diameter

diameter = Double.parseDouble(tbxDiameter.getText());

if (diameter > maxDiameter || diameter <= 0) {

err += "Diameter is invalid. Must be between 0 and 4 inches.\n";

}

} catch (NumberFormatException e) {

//Not a Number

err += "Diameter is not a number. \n";

}

//store current grade

grade = (byte) cbxGrade.getSelectedIndex();

grade++;

//test the pipe is valid.

String testPipe = pipeFactory.ValidatePipe(grade, cbxChemicalResistance.isSelected(), length, diameter, cbxInsulation.isSelected(),

cbxReinforcement.isSelected(), cbxColour.getSelectedIndex(), (int) spnQty.getValue());

if (testPipe.charAt(0) != 'E' && err == "") {

//Finally add the pipe!

pipes.add(pipeFactory.MakePipe(grade, cbxChemicalResistance.isSelected(), length, diameter, cbxInsulation.isSelected(),

cbxReinforcement.isSelected(), cbxColour.getSelectedIndex(), (int) spnQty.getValue()));

Pipe p = pipes.get(pipes.size() - 1); //get last pipe added.

//adjust the length of spacing to format correctly

String spaceLen = " ";

if (p.getQty() >= 10 && p.getQty() < 100) {

spaceLen = " ";

}

if (p.getQty() == 100) {

spaceLen = " ";

}

//add the new pipe to the UI so the user can view / delete pipes they do not want

model.addElement(" "

+ Integer.toString(p.getGrade())

+ " "

+ p.getColour()

+ " "

+ dec.format(pipeFactory.convertToMeters(p.getLength()))

+ " "

+ dec.format(p.getOuterDiameter())

+ " "

+ p.getInnerInsulation()

+ " "

+ p.getOuterReinforcement()

+ " "

+ p.getChemicalResistance()

+ spaceLen

+ p.getQty()

+ " "

+ dec.format(p.getCostTotal()));

//end adding to element

lbxOrderList.setModel(model);

err += "\n Added Successfully!";

tbxError.setForeground(Color.BLACK);

tbxError.setText(err);

updateTotal();

} else {

err += "We cannot make a pipe to these specifications";

tbxError.setForeground(Color.RED);

tbxError.setText("Error:\n" + err);

}

}

//delete a pipe from the order, removes both from the GUI JList and the ArrayList.

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {

int selectedItem = lbxOrderList.getSelectedIndex();

//check selected, if so remove from the model and the pipes arraylist

if (selectedItem > -1) {

model.remove(selectedItem);

pipes.remove(selectedItem);

updateTotal();

//Select the item at the same index as the deleted, if it doesn't exist pick the previous item.

if (selectedItem < model.size()) {

lbxOrderList.setSelectedIndex(selectedItem);

} else {

lbxOrderList.setSelectedIndex(selectedItem - 1);

}

}

}

//button to remove all from current order.

private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {

pipes.clear();

model.clear();

updateTotal();

}

//Update total cost in tbxTotal

private void updateTotal() {

double total = 0;

for (Pipe pipe : pipes) {

total += pipe.getCostTotal();

}

if (total != 0) {

tbxTotal.setText(dec.format(total));

} else {

tbxTotal.setText("0.00");

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the System look and feel, modified to do so.\*/

//<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 {

// Set System L&F

javax.swing.UIManager.setLookAndFeel(

javax.swing.UIManager.getSystemLookAndFeelClassName());

//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(GUI.class

.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(GUI.class

.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(GUI.class

.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(GUI.class

.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\*PROGRAM STARTS HERE\*/

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new GUI().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton btnAdd;

private javax.swing.JButton btnCancel;

private javax.swing.JButton btnClear;

private javax.swing.JButton btnDelete;

private javax.swing.JButton btnSubmit;

private javax.swing.JCheckBox cbxChemicalResistance;

private javax.swing.JComboBox<String> cbxColour;

private javax.swing.JComboBox<String> cbxGrade;

private javax.swing.JCheckBox cbxInsulation;

private javax.swing.JCheckBox cbxReinforcement;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel11;

private javax.swing.JLabel jLabel12;

private javax.swing.JLabel jLabel14;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel7;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JScrollPane jScrollPane4;

private javax.swing.JList<String> lbxOrderList;

private javax.swing.JSpinner spnQty;

private javax.swing.JTextField tbxDiameter;

private javax.swing.JTextArea tbxError;

private javax.swing.JTextField tbxLength;

private javax.swing.JTextField tbxTotal;

// End of variables declaration

}

**Main code (Pipe)**

package main;

/\*\*

\*

\* @author 781212, 788777, 777611, 790487

\*

\*/

//UPDATE SUBCLASSES TO BE MORE STRUCTURED INSTEAD OF ALL COMING OFF OF THIS DIRECTLY

/\*

Ah, a glorious Pipe!

Or, Picasso's implementation of one... Make sure not to reference this class

directly to make a pipe. PipeFactory will do the work for you.

\*/

public abstract class Pipe {

private int qty;

protected int grade; //Grade 1-5

private boolean chemicalResistance; //Available for all

protected double[] costPerInch; //Used to display available grades and their equivilence

protected double costTotal; //The total cost of the pipe

protected double baseCost; //The base cost

protected int colours;

//All dimensions stored in inches

private double length;

private double outerDiameter;

private double innerDiameter;

public Pipe(int grade, boolean chemicalResistance, double length, double outerDiameter, int qty, int colours) {

this.grade = grade;

this.chemicalResistance = chemicalResistance;

this.length = length;

this.outerDiameter = outerDiameter;

innerDiameter = outerDiameter \* 0.9;

this.qty = qty;

this.colours = colours;

}

//get / set methods

public int getGrade() {

return grade;

}

public void setChemicalResistance(boolean chemicalResistance) {

this.chemicalResistance = chemicalResistance;

}

public double getCostPerInch() {

return costPerInch[grade];

}

public double getCostTotal() {

return costTotal \* qty;

}

public double getLength() {

return length;

}

public double getOuterDiameter() {

return outerDiameter;

}

public double getInnerDiameter() {

return innerDiameter;

}

public int getQty() {

return qty;

}

public int getColour() {

return colours;

}

public String getInnerInsulation() {

return "No ";

}

public String getOuterReinforcement() {

return "No ";

}

public String getChemicalResistance() {

if (chemicalResistance) {

return "Yes";

}

return "No ";

}

/\*

METHODS

\*/

//Work out area of pipe

private double pipeVolume() {

//subtract volume inside of the pipe from the total volume to get pipe volume

return cylinderVolume(outerDiameter) - cylinderVolume(innerDiameter);

}

//Work out area of cylinder

private double cylinderVolume(double d) {

//formula for area of a cylindar.

return Math.PI \* Math.pow((d / 2), 2) \* length;

}

//Work out your cost

protected void calculateBaseCost() {

double cost = pipeVolume() \* costPerInch[grade - 1];

baseCost = cost;

if (chemicalResistance) {

cost += baseCost \* 0.14;//add chemical resistance

}

costTotal = cost;

}

/\*Other additionalCosts can be calculated here.

The base cost is universal.

Class is overridden in every Type with exception to

T1 in order to add all additional costs.

\*/

protected void calculateCost() {

calculateBaseCost();

}

}

**Main code (PipeFactory):**

package main;

/\*\*

\*

\* @author 781212, 788777, 777611, 790487

\*

\*/

/\*

Welcome to the Pipe Factory!

Here pipes are made based on the input criteria given

(check the different features of the pipe, then create the right tier in the return)

Please input the data from user input here for validation and to determine which pipe is needed,

DO NOT REFERENCE PIPE OR ITS CHILDREN DIRECTLY FROM OUTSIDE PIPEFACTORY WHEN CREATING PIPES!!!

\*/

public class PipeFactory {

//makes the pipes. Ensure pipes have been tested by TestPipe before doing so.

public Pipe MakePipe(int grade, boolean chemicalResistance, double length, double outerDiameter, boolean innerInsulation, boolean outerReinforcement, int colour, int qty) {

length = convertToInches(length);//convert length to inches, all values are the same type within pipe classes.

String type = ValidatePipe(grade, chemicalResistance, length, outerDiameter, innerInsulation, outerReinforcement, colour, qty);

switch (type) {

case "Type 1":

PipeT1 pip1 = new PipeT1(grade, chemicalResistance, length, outerDiameter, qty);

return pip1;

case "Type 2":

PipeT2 pip2 = new PipeT2(grade, chemicalResistance, length, outerDiameter, qty);

return pip2;

case "Type 3":

PipeT3 pip3 = new PipeT3(grade, chemicalResistance, length, outerDiameter, qty);

return pip3;

case "Type 4":

PipeT4 pip4 = new PipeT4(grade, chemicalResistance, length, outerDiameter, qty);

return pip4;

default:

PipeT5 pip5 = new PipeT5(grade, chemicalResistance, length, outerDiameter, qty);

return pip5;

}

}

//checks if a pipe is valid and returns a String of the pipe's type. Error is given in the case of a pipe being invalid.

public String ValidatePipe(int grade, boolean chemicalResistance, double length, double outerDiameter, boolean innerInsulation, boolean outerReinforcement, int colour, int qty) {

if (grade > 0 && colour >= 0 && colour <= 2) {

if (colour == 2 && grade >= 2) {

if (innerInsulation) {

if (outerReinforcement && grade >= 3) {

return "Type 5"; //type 5

} else if (!outerReinforcement){

return "Type 4"; //type 4

}

}

if (!innerInsulation && !outerReinforcement) {

return "Type 3"; //type 3

}

}

if (!innerInsulation && !outerReinforcement) {

if (colour == 1 && grade >= 2 && grade <= 4) {

return "Type 2"; //type 2

}

if (colour == 0 && grade <= 3) {

return "Type 1"; //type 1

}

}

}

return "Error, this type of pipe is not available."; //grade invalid. Either invalid input or invalid type

}

protected double convertToInches(double meter) {

return meter \* 39.37;

}

protected double convertToMeters(double inch) {

return inch \* 0.0254;

}

}

**Pipe Type 1 code:**

package main;

/\*\*

\*

\* @author 781212, 788777, 777611, 790487

\*

\*/

public class PipeT1 extends Pipe {

public PipeT1(int grade, boolean chemicalResistance, double length, double outerDiameter, int qty) {

super(grade, chemicalResistance, length, outerDiameter, qty,0);

costPerInch = new double[]{0.4, 0.6, 0.75,0,0}; //SET COST PER INCH FOR CLASS

calculateCost();

}

}

**Pipe Type 2 code:**

package main;

/\*\*

\*

\* @author 781212, 788777, 777611, 790487

\*

\*/

public class PipeT2 extends Pipe {

public PipeT2(int grade, boolean chemicalResistance, double length, double outerDiameter, int qty) {

super(grade, chemicalResistance, length, outerDiameter, qty,1);

costPerInch = new double[]{0, 0.6, 0.75, 0.8};//set cost per inch

calculateCost();

}

//override the cost calculation to add a colour

@Override

public void calculateCost() {

calculateBaseCost();

costTotal += baseCost \* 0.12; //add 1 colour

}

//override to return correct grade.

@Override

public int getGrade() {

return grade;

}

}

**Pipe Type 3 code:**

package main;

/\*\*

\*

\* @author 781212, 788777, 777611, 790487

\*

\*/

public class PipeT3 extends Pipe {

public PipeT3(int grade, boolean chemicalResistance, double length, double outerDiameter, int qty) {

super(grade, chemicalResistance, length, outerDiameter, qty, 2);

costPerInch = new double[]{0, 0.6, 0.75, 0.8, 0.95};//set cost per inch

calculateCost();

}

//Add two colours to the base cost

@Override

public void calculateCost() {

calculateBaseCost();

costTotal += baseCost \* 0.16; //add 2 colours

}

//override to return correct grade.

@Override

public int getGrade() {

return grade;

}

}

**Pipe Type 4 code:**

package main;

/\*\*

\*

\* @author 781212, 788777, 777611, 790487

\*

\*/

public class PipeT4 extends PipeT3 {

//costPerInch not needed, T3 and 4 share the same cost per inch.

public PipeT4(int grade, boolean chemicalResistance, double length, double outerDiameter, int qty) {

super(grade, chemicalResistance, length, outerDiameter, qty);

calculateCostT4();

}

public void calculateCostT4() {

costTotal += baseCost \* 0.13;//add Inner insulation

}

@Override

public String getInnerInsulation(){

return "Yes";

}

}

**Pipe Type 5 code:**

package main;

/\*\*

\*

\* @author 781212, 788777, 777611, 790487

\*

\*/

public class PipeT5 extends PipeT4 {

public PipeT5(int grade, boolean chemicalResistance, double length, double outerDiameter, int qty) {

super(grade, chemicalResistance, length, outerDiameter, qty);

costPerInch = new double[]{0, 0, 0.75, 0.8, 0.95};//set cost per inch

calculateCost();

}

//override to ensure correct output

@Override

public void calculateCost() {

calculateBaseCost();

costTotal += baseCost \* 0.16; //add 1 colour

costTotal += baseCost \* 0.17;//add Inner insulation

}

//override to return correct grade.

@Override

public int getGrade() {

return grade;

}

@Override

public String getOuterReinforcement() {

return "Yes";

}

}

**ADPROC Coursework - Group Contribution**

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

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

## ADPROC, Advanced Programming Concepts (U21266)

## 

## Coursework

Hand out: 23.X.2017 Submission (Moodle): 8.XII.2017 (Demonstration: by week11- starting 4th Dec)

This is an assessed piece of group (of two) 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 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 pipe-selling company

***“LongPipes”*** is a company producing a variety of pipes for water, drainage, fuel, gas, conduit, plumbing and heating. Due to the wide range of requirements of their customers, the variety of pipes they produce is very extensive.

The pipes are all made of plastic, but some may have metallic enforcement and other features:

* They are all made of plastic;

**Fig.1.**Pipe’s cross-section.

Ro

Ri

L

* Their plastic has a specified grade;
* They may have no colour, 1 colour, or 2 colours;
* They may have inner insulation layer;
* They may have outer metallic reinforcement;
* They may also have improved chemical resistance.

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Type** | **Plastic’s grade** | **Colour print** | | | **Inner insulation** | **Outer reinforcement** | **Chemical**  **resistance** |
| **0** | **1** | **2** |
| I | 1 – 3 | YES | NO | NO | NO | NO | YES/NO |
| II | 2 – 4 | NO | YES | NO | NO | NO | YES/NO |
| III | 2 – 5 | NO | NO | YES | NO | NO | YES/NO |
| IV | 2 – 5 | NO | NO | YES | YES | NO | YES/NO |
| V | 3 – 5 | NO | NO | YES | YES | YES | YES/NO |

***Table 1.*** *Types of plastic pipes available.*

Pipes are available in straights (up to 6 meters). When ordering, the client should specify the length (in meters, 1m = 39.37”) and the outer diameter (or outer radius Ro) – in inches (1” = 0.0254m). Assume the inner diameter (or the inner radius Ri) is always 90% of the outer one. The basic cost is calculated using the volume of the pipe’s material (in cubic inches) and the costs of 1 inch3 of plastic is given in Table 2.

***Table 3.*** *Additional costs.*

|  |  |
| --- | --- |
| 1 colour | 12% extra |
| 2 colours | 16% extra |
| Inner insulation | 13% extra |
| Outer reinforcement | 17% extra |
| Chemical resistance | 14% extra |

***Table 2.*** *Cost of 1 cubic inch of plastic*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Grade of plastic | 1 | 2 | 3 | 4 | 5 |
| Cost per inch3 [in £] | 0.4 | 0.6 | 0.75 | 0.8 | 0.95 |

There are some additional costs, depending on whether the pipe has colour and if there is any insulation and/or reinforcement. These are shown in Table 3 and the percentage increase **is calculated using the basic cost**.

All pipes may have improved chemical resistance. When customers ask ***LongPipes*** to quote a price for an order, they specify the following features:

* The size of pipe (length in meters and outer diameter in inches);
* The grade of the plastic;
* Whether they want any colour (no colour, or 1, or 2 colour plastic);
* Whether they want any insulation or/and reinforcement;
* Whether they want pipe with chemical resistance;
* The quantity of pipes for the order.

From this information, the order system should determine if the requested type of pipe can be supplied by ***LongPipes***, and if not, it should display an appropriate message and reject the order (e.g., pipe of plastic grade 1 and inner insulation is an invalid order). If the ordered pipe corresponds to one of the types given in Table 1, and can be supplied by ***LongPipes***, the cost of the pipes 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 pipe 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 type of pipes based on the ordered pipe characteristics.**

Customers should be able to receive a quote for as many pipes (of different types) as they like (within the capacity of ***LongPipes***) 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 used, you will lose the marks allocated for this part of your coursework.

## Your Task

* Write an application that allows the customer to enter the details of his/her order and subsequently prompts the cost of the order. Your application should verify that ***LongPipes*** can supply the corresponding to the order type of pipe (the customer should not be asked to specify the pipe type).
* Use OO design approach (abstraction, inheritance and polymorphism) and create appropriate class hierarchy, which reflects on the types of pipes that ***LongPipes*** 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.
* 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 week11 (starting **4.XII.2017**), during your lab session.

On **8.XII.2017** 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.) **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 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 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 implement 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 | 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?  Is the layout clear? | 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 |  |