ADPROC Coursework

Order system for a pipe-selling company

Group C-7
UP822876
UP823183
UP819561
UP820471

Date:
8th December 2017

Table of Contents

1.0 Brief

‘LongPipes’ application allows customers to enter information about desired pipe requirements including: ·

- 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.

Based on the customers requirements, the application will determine if the type of pipe can be supplied by LongPipes. If there are any validation errors then an error message will be displayed to the customer, informing them to change their current order. The application will calculate a final price of the customers order based on the requirements and quantity of pipes needed, which is displayed on a form to the customer. The application is designed so that customers enter inputs on a form through text boxes and check boxes. The customer can then calculate the cost of each pipe which is also added on to their final order.

1.1 Assumptions:

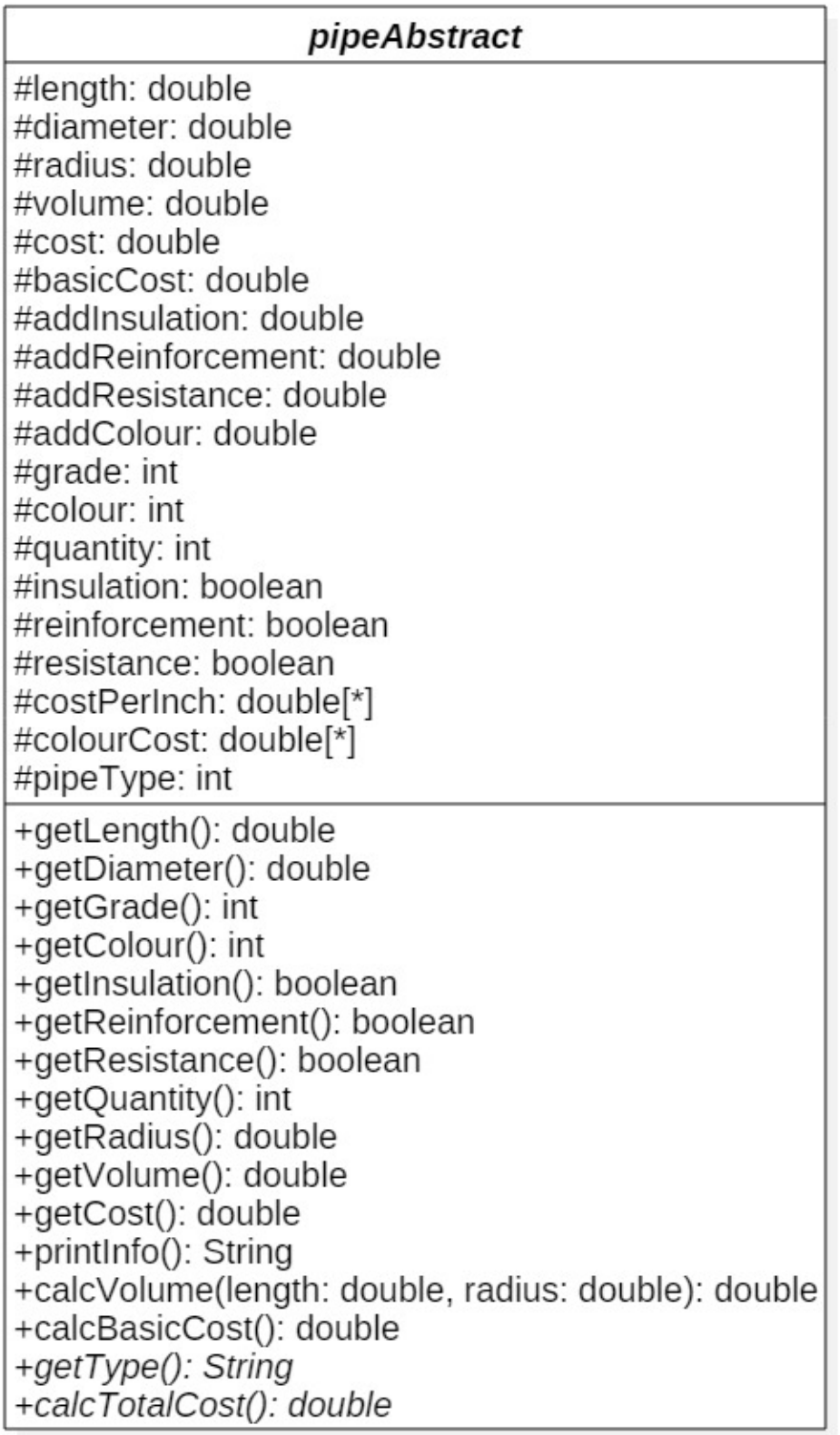
- LongPipes company does not produce pipes with length less than 1 meter and larger than 6 meter.
- They don't produce pipes with diameter less than 1 inch and larger than 50 inches.
- They produce minimum one pipe per order and maximum of 100 per order.
- Another assumption we made is that once the client is done making their order or orders, the company will close the application and reopen the application when another client comes in to make an order.
- Therefore, the application doesn’t require a clear all button on the current order.
- We assume the company needs to display all the details of the pipe after the order is been made.

1.2 Limitations:

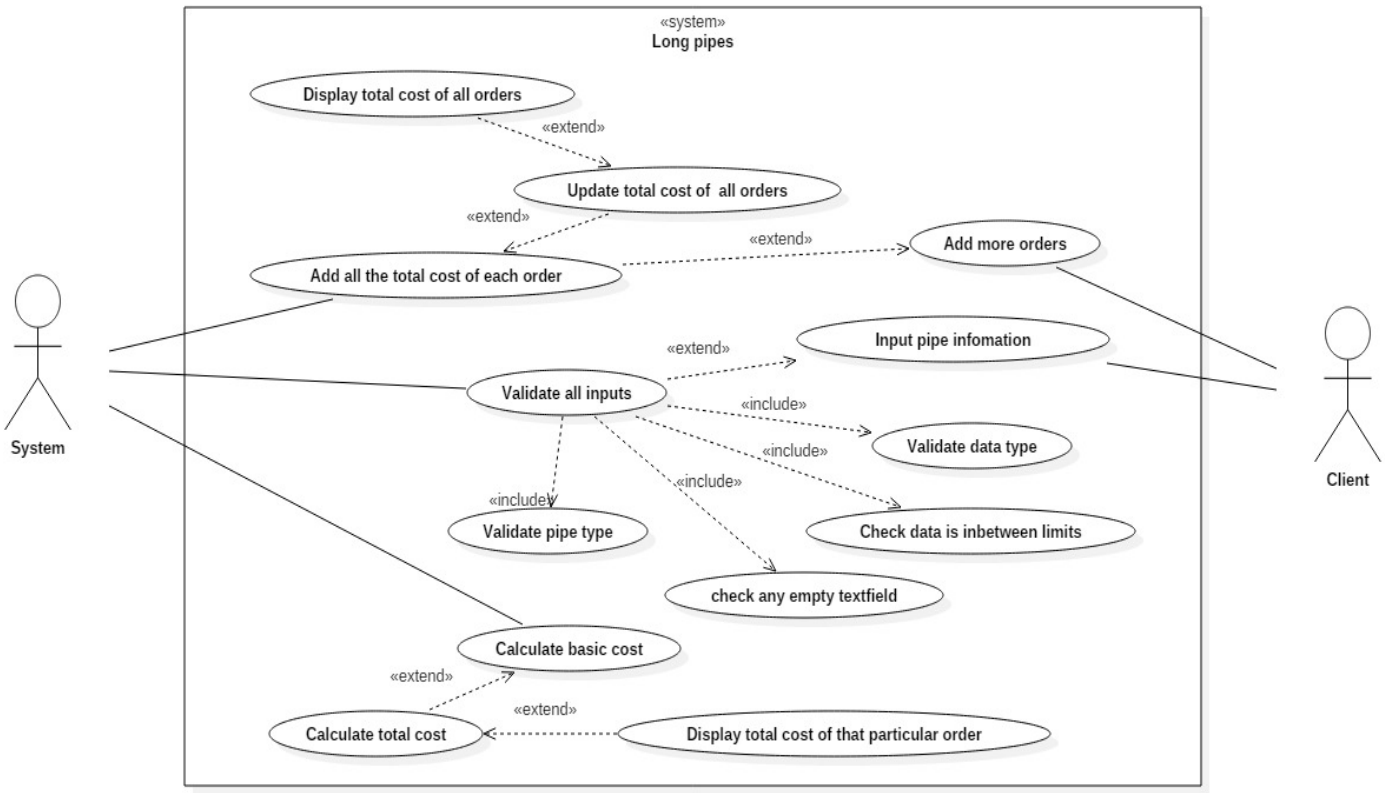
- Once the application has been closed, users are unable to view previous orders as the orders are not stored in a database.
- The arraylist (orderBasket) is declared as final, therefore any items added to the orderBasket can not be removed.
- Only one user can make an order at a time.
- The user is unable to edit orders once they have been submitted into the arraylist.
- The components on the GUI aren’t resizable so the resolution is fixed

2.0 UMLs

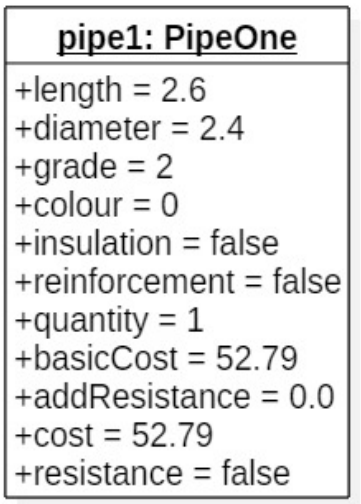
2.1 Class Diagram



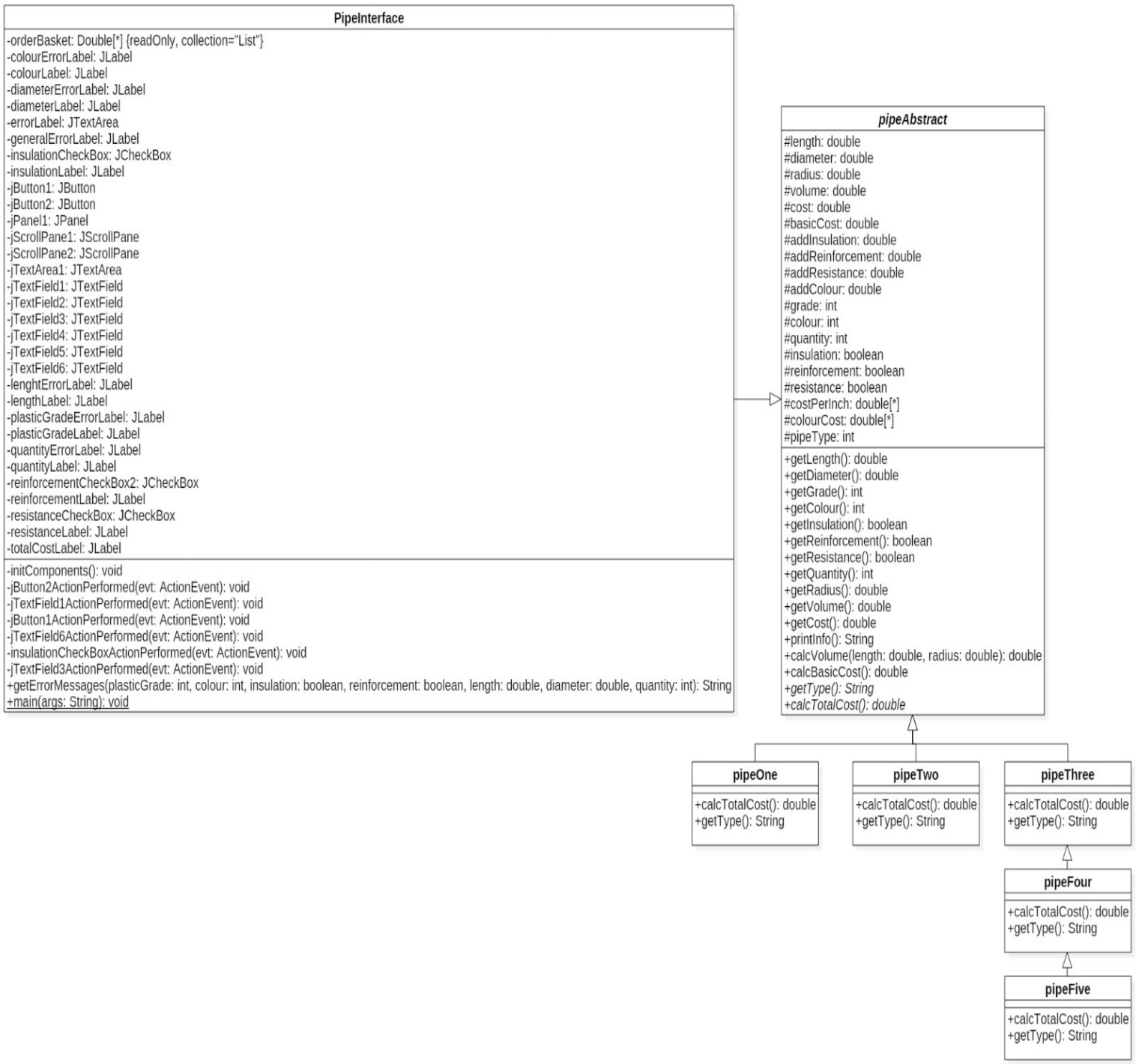
2.2 Use Case



2.3 Instance Diagram



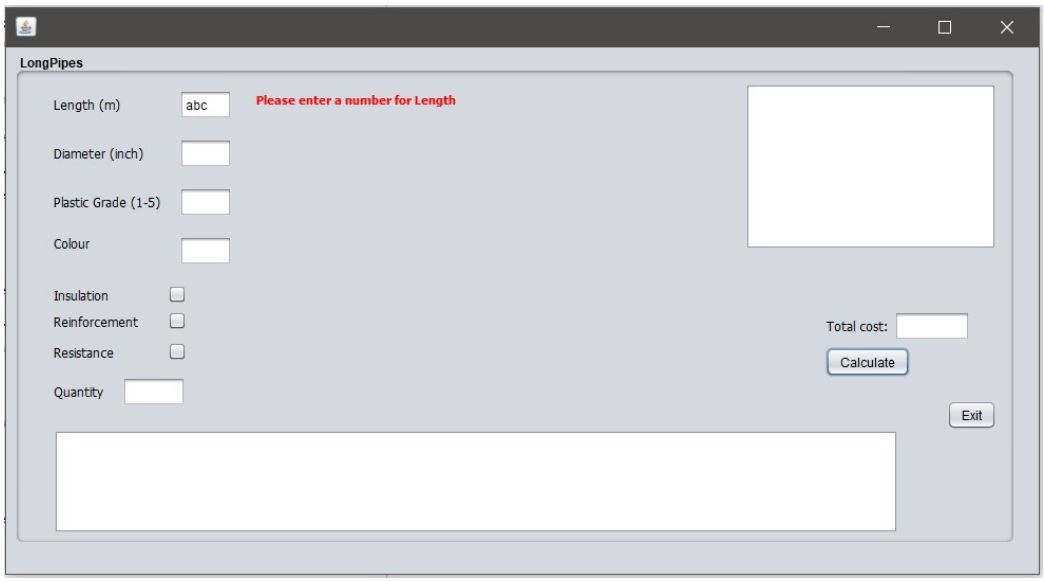
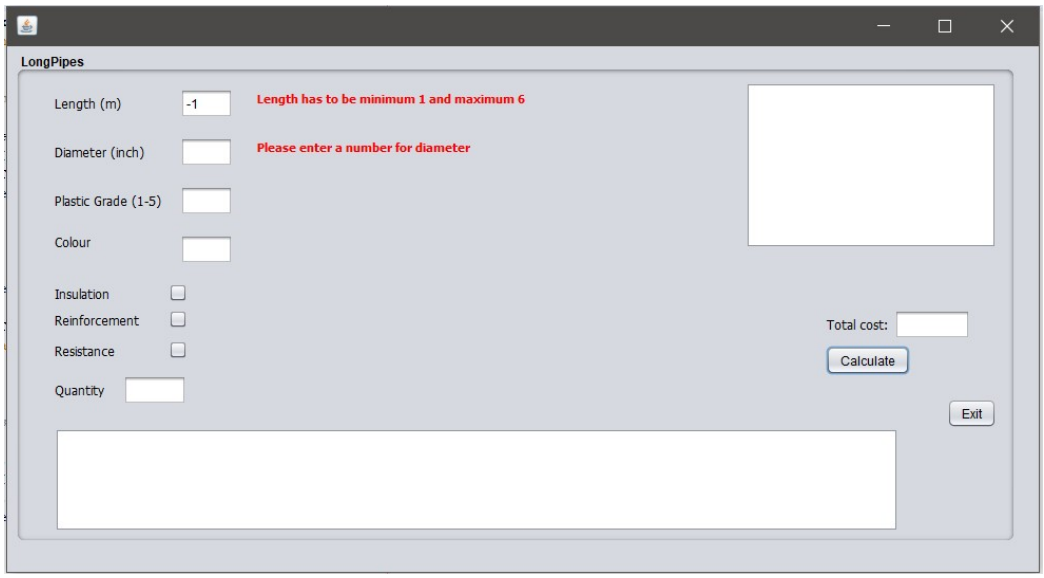
2.4 Class hierarchy Diagram

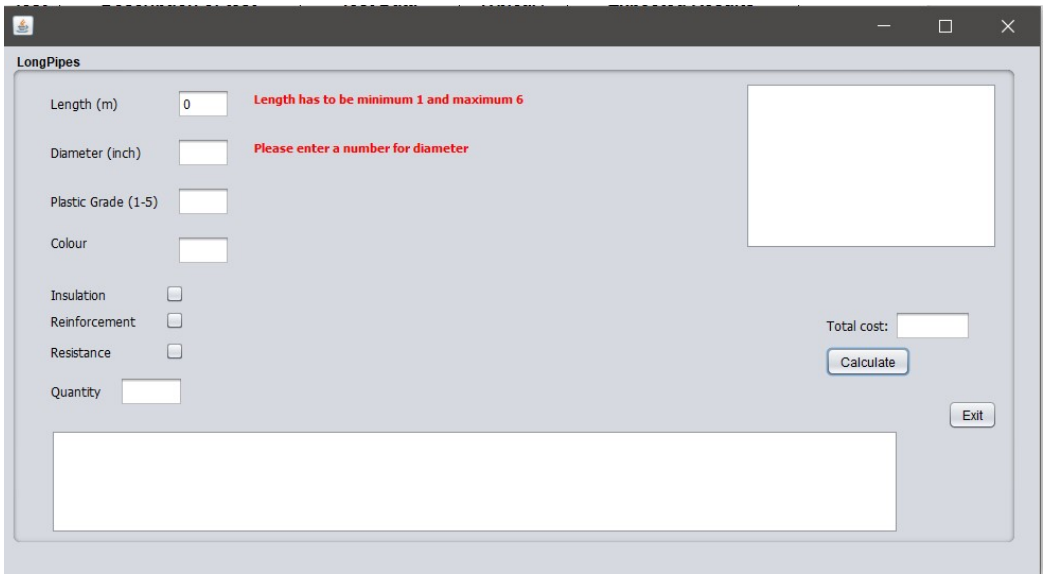
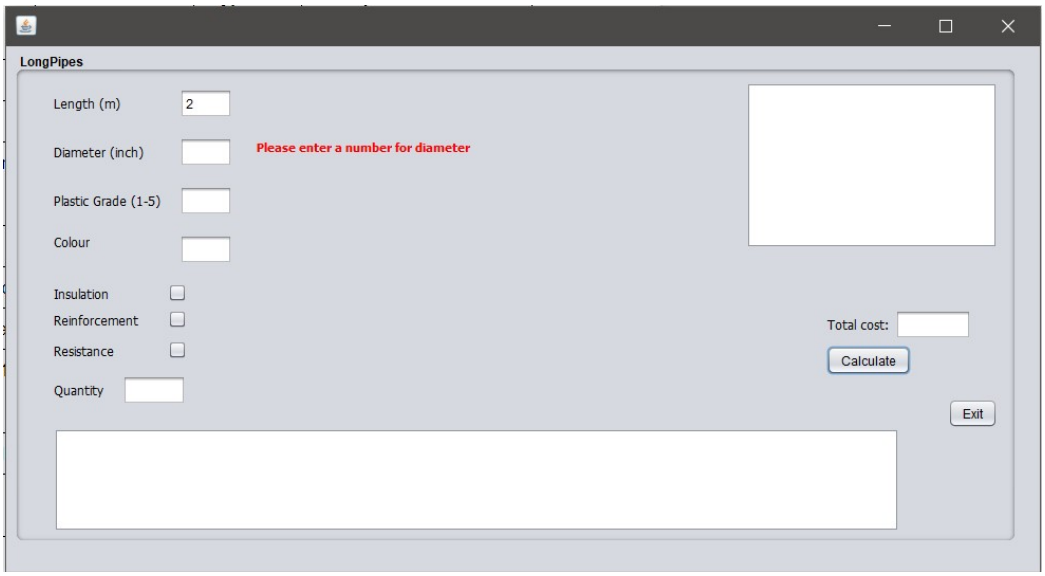


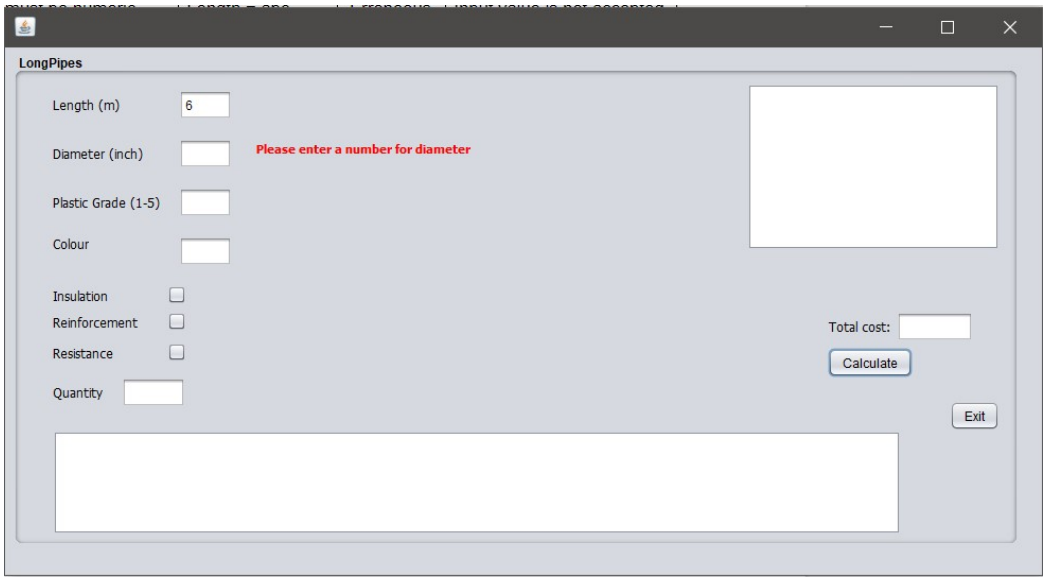
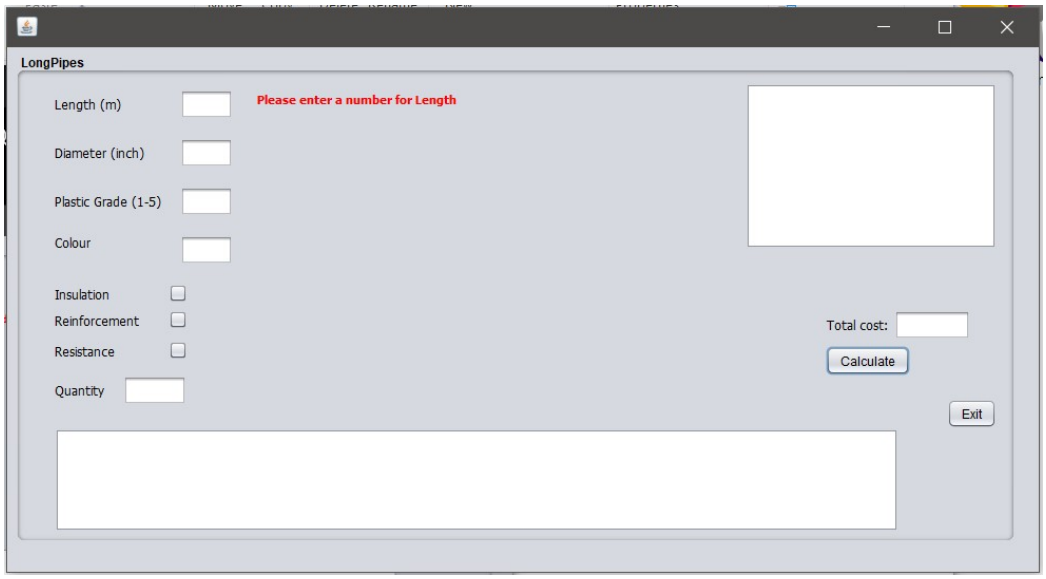
Test Schedule

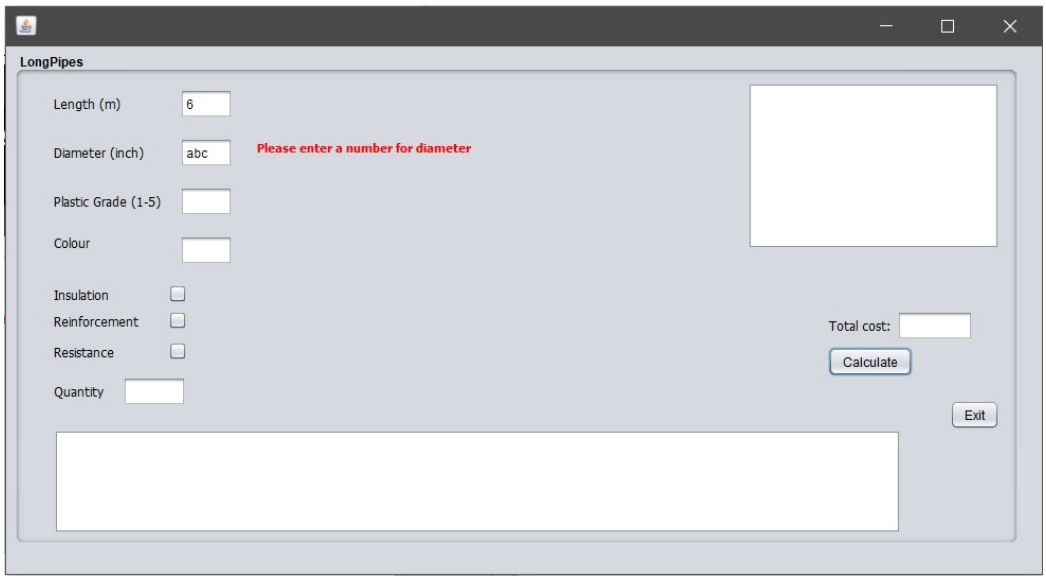
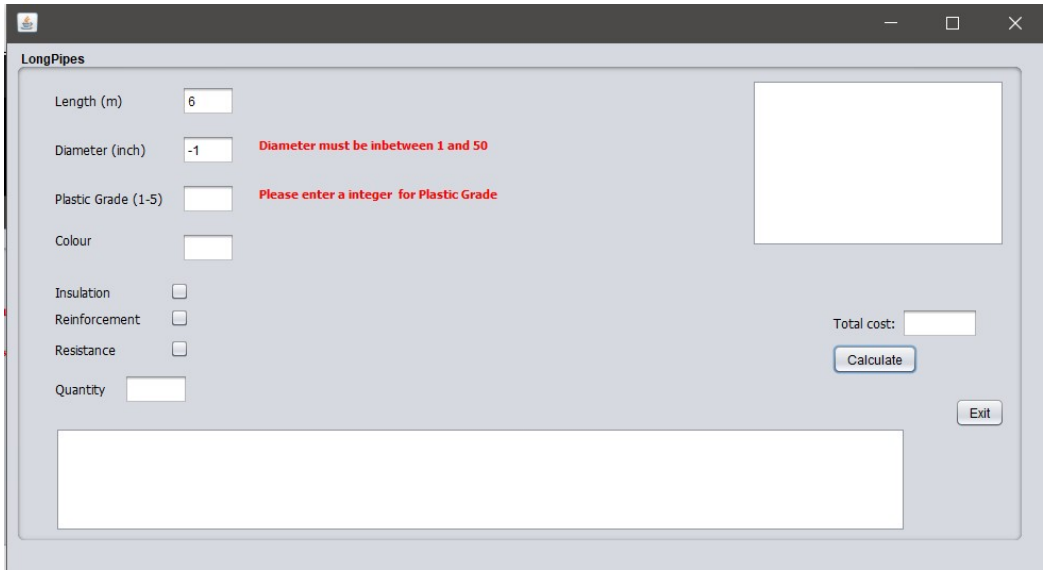
Test No.	Description of test	Test Data Entered	Typical / Extreme/ Erroneous	Expected Results
1	Length must be numeric	Length = abc	Erroneous	Input value is not accepted
2	Length must be positive	Length = -1	Erroneous	Input value is not accepted
3	Length must between 1 and 6	Length = 0 Length = 2 Length = 6	Erroneous Typical Extreme	Input value is not accepted Input value is accepted Input value is accepted
4	Length must not be null	Length = ""	Erroneous	Input value is not accepted
5	Diameter must be numeric	Diameter = abc	Typical	Input value is not accepted
6	Diameter must be positive	Diameter = -1	Erroneous	Input value is not accepted
7	Diameter must between 1 and 50	Diameter = 0 Diameter = 1 Diameter = 50	Erroneous Typical Extreme	Input value is not accepted Input value is accepted Input value is accepted
8	Diameter must not be null	Diameter= ""	Erroneous	Input value is not accepted
9	Plastic grade must be numeric	Plastic grade = abc	Erroneous	Input value is not accepted
10	Plastic grade must be positive	Plastic grade = -1	Erroneous	Input value is not accepted
11	Plastic grade must be either 1 ,2 ,3 ,4 or 5	Plastic grade = 0 Plastic grade = 2	Erroneous Typical	Input value is not accepted Input value is accepted
12	Plastic grade must not be null	Plastic grade = ""	Erroneous	Input value is not accepted
13	Colour must be numeric	Colour = abc	Erroneous	Input value is not accepted
14	Colour must be positive	Colour = -1	Erroneous	Input value is not accepted
15	Colour must be either 0 ,1 or 2	Colour = 3 Colour = 2	Erroneous Typical	Input value is not accepted Input value is accepted
16	Colour must not be null	Colour = ""	Erroneous	Input value is not accepted
17	Quantity must be numeric	Quantity = abc	Erroneous	Input value is not accepted
18	Quantity must be positive	Quantity = -1	Erroneous	Input value is not accepted
19	Quantity must be no bigger than 100 and no smaller than 1	Quantity = 0 Quantity = 2 Quantity = 100	Erroneous Typical Extreme	Input value is not accepted Input value is accepted Input value is accepted
20	Quantity must not be null	Quantity = ""	Erroneous	Input value is not accepted

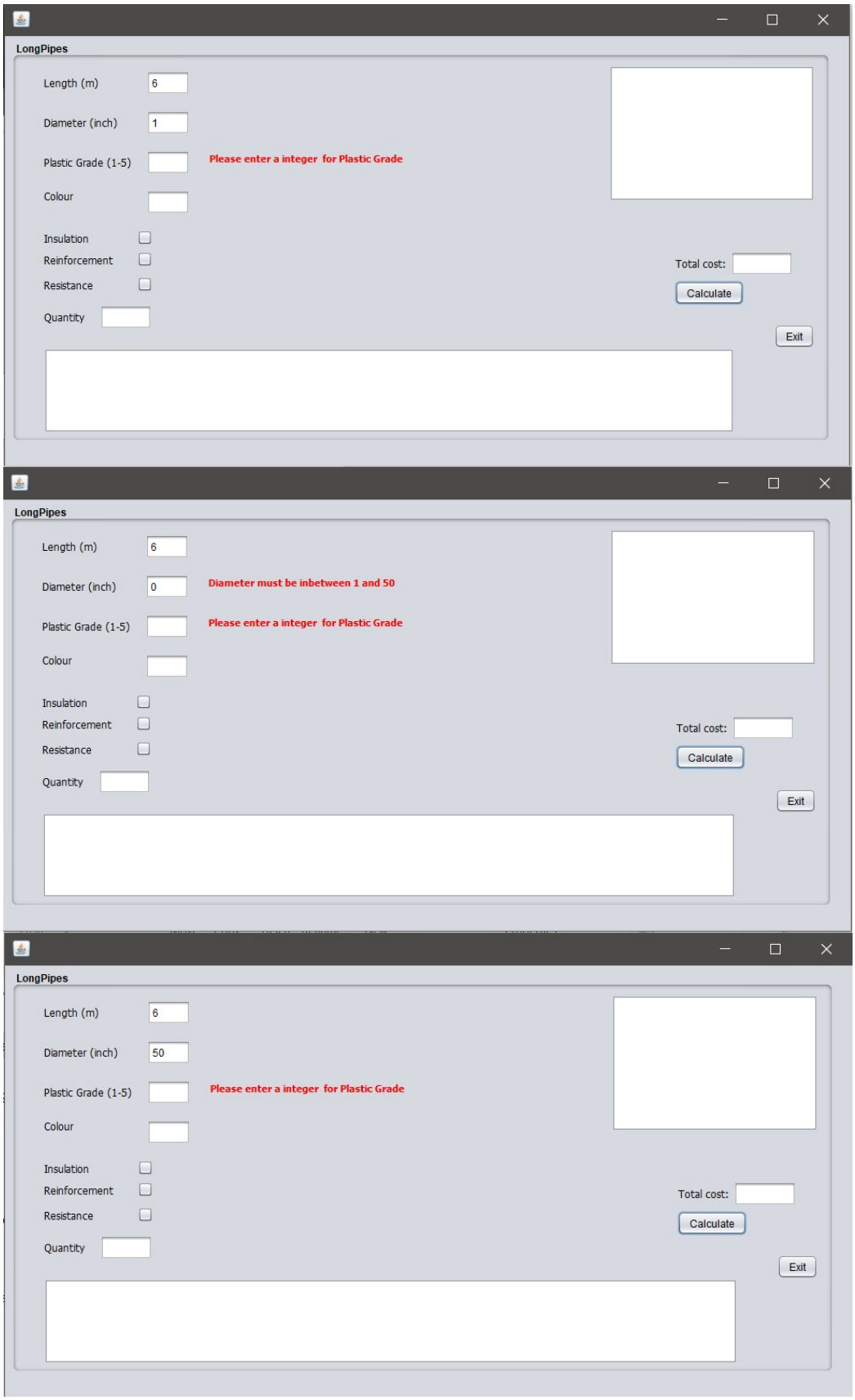
If the input value is not accepted, the application will display the corresponding error message depends on which attributes to notify users (Client) that they have inputted invalid data..

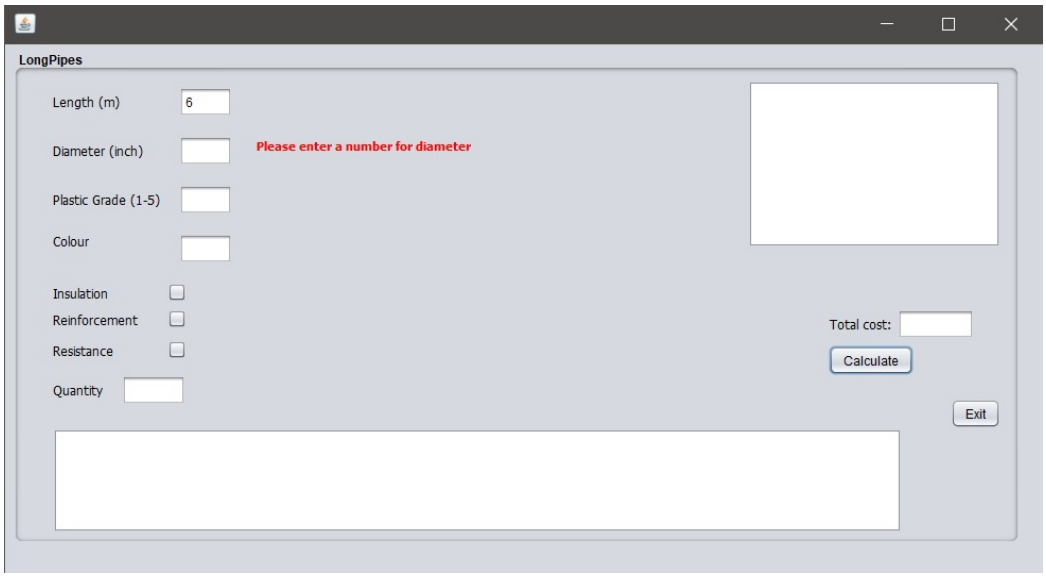
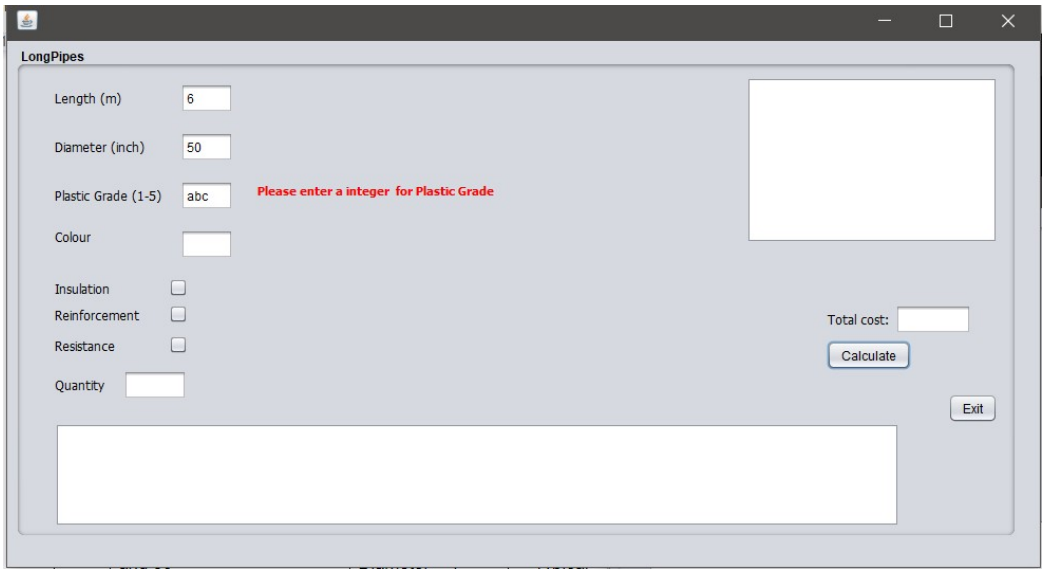
Test No.	Screenshot
1	
2	

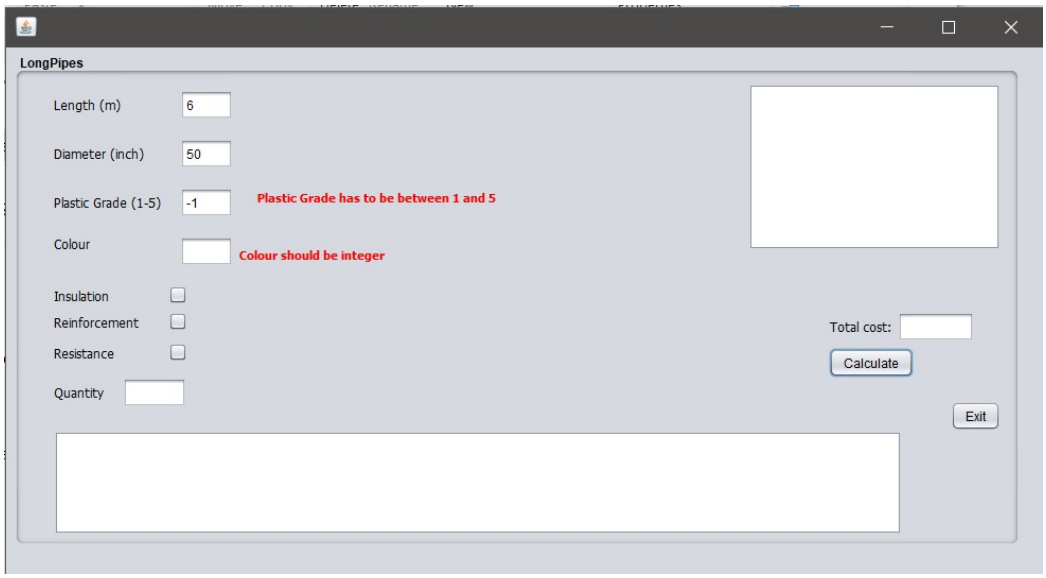
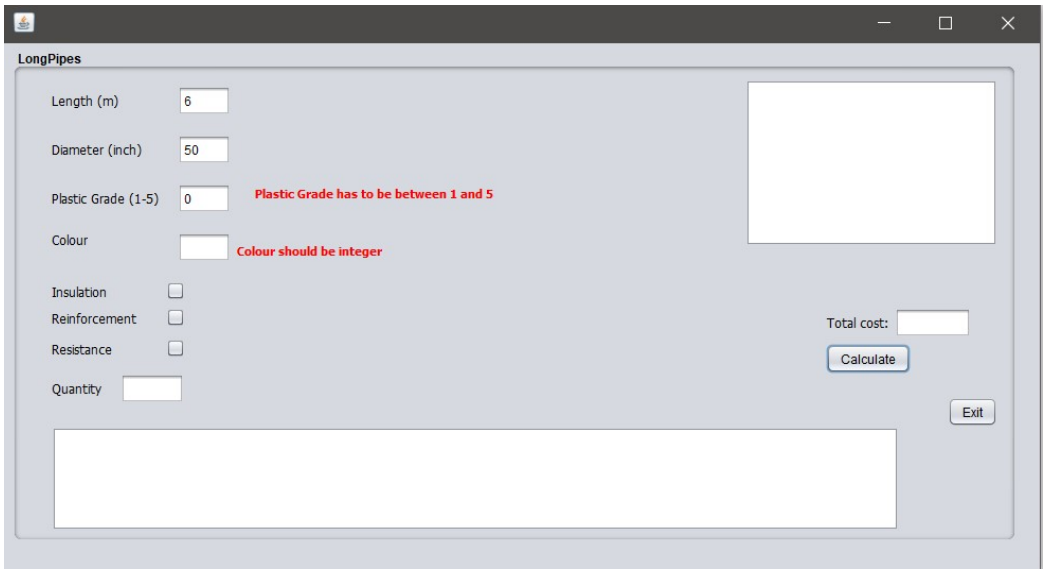
3	 LongPipes Length (m) 0 Length has to be minimum 1 and maximum 6 Diameter (inch) Please enter a number for diameter Plastic Grade (1-5) Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit  LongPipes Length (m) 2 Diameter (inch) Please enter a number for diameter Plastic Grade (1-5) Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit

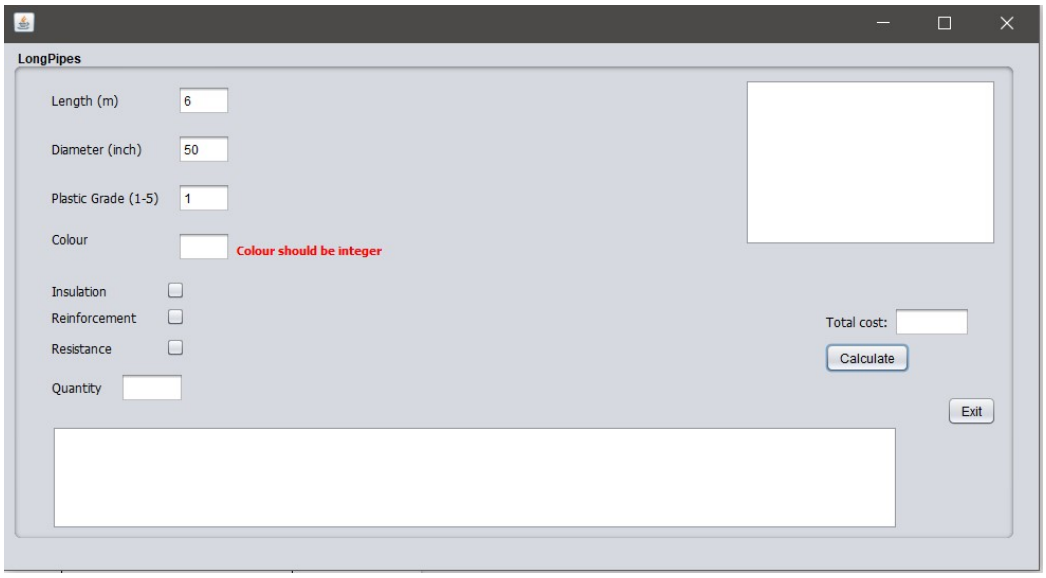
	 LongPipes Length (m) 6 Diameter (inch) Please enter a number for diameter Plastic Grade (1-5) Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit
4	 LongPipes Length (m) Please enter a number for Length Diameter (inch) Plastic Grade (1-5) Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit

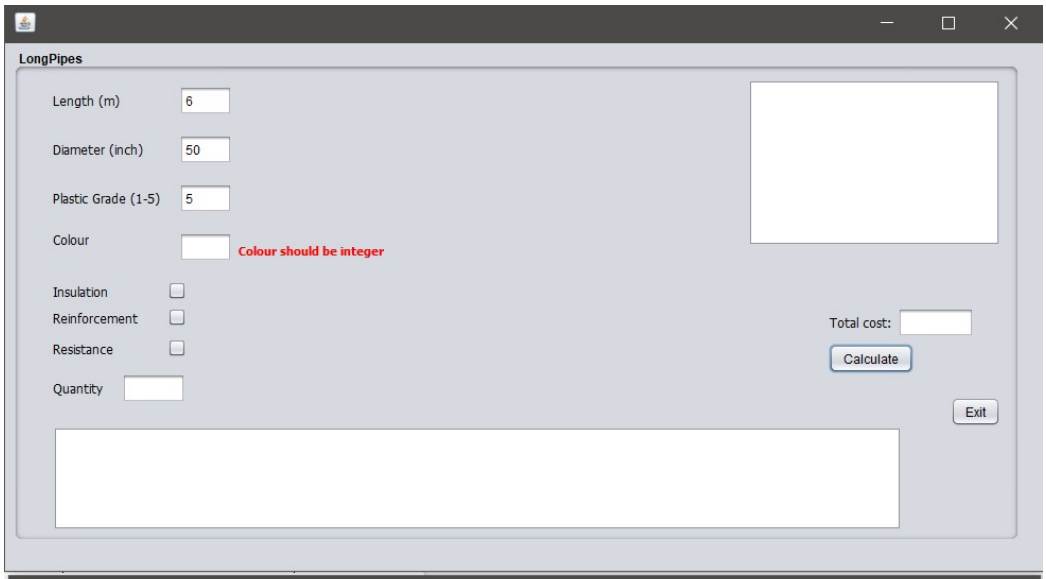
5	 LongPipes Length (m) 6 Diameter (inch) abc Please enter a number for diameter Plastic Grade (1-5) Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit
6	 LongPipes Length (m) 6 Diameter (inch) -1 Diameter must be inbetween 1 and 50 Plastic Grade (1-5) Please enter a integer for Plastic Grade Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit

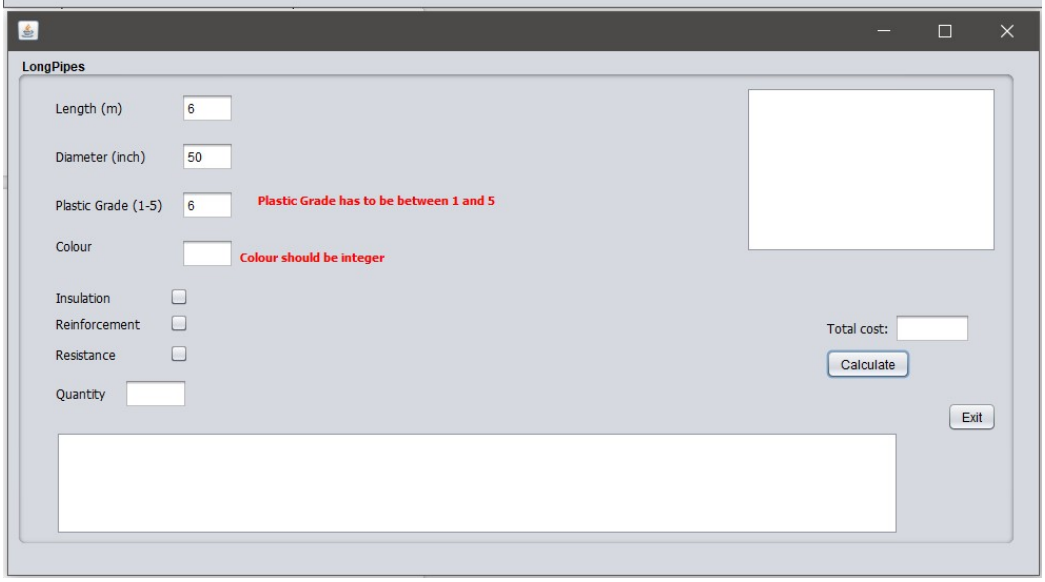
7	 LongPipes Length (m) 6 Diameter (inch) 1 Plastic Grade (1-5) Please enter a integer for Plastic Grade Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit LongPipes Length (m) 6 Diameter (inch) 0 Diameter must be inbetween 1 and 50 Plastic Grade (1-5) Please enter a integer for Plastic Grade Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) Please enter a integer for Plastic Grade Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit

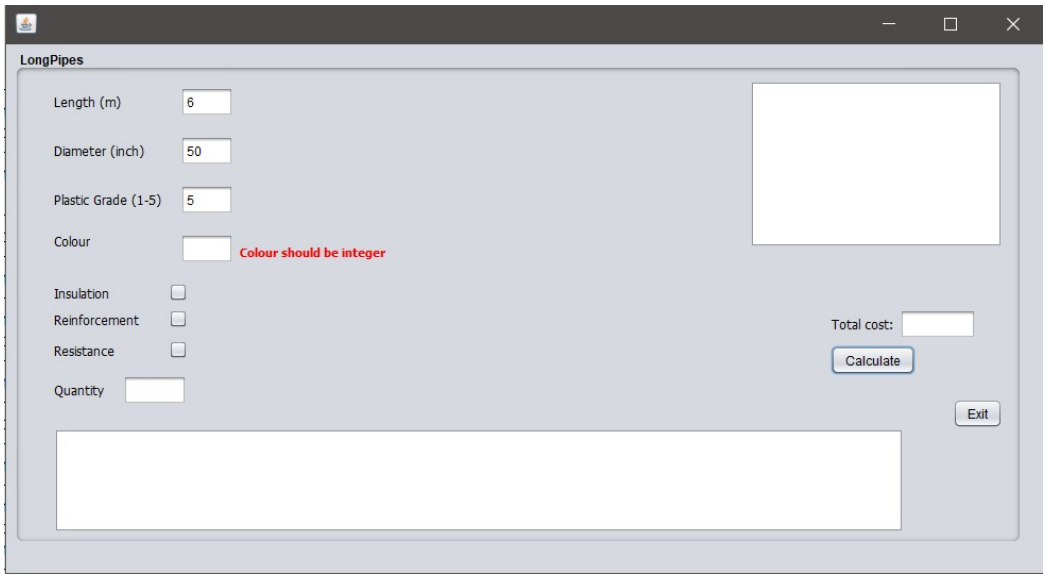
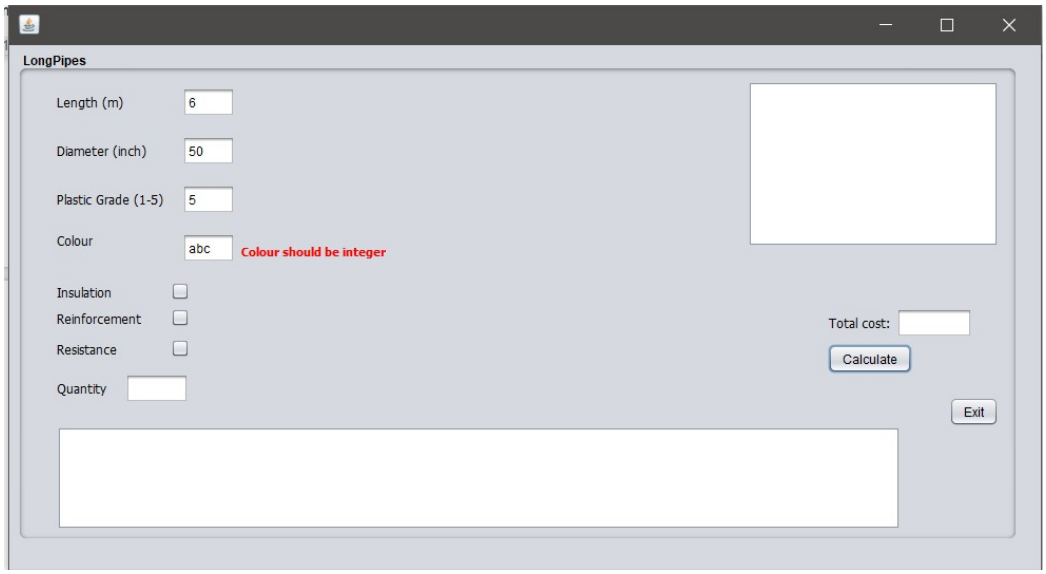
8	 LongPipes Length (m) 6 Diameter (inch) Please enter a number for diameter Plastic Grade (1-5) Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit
9	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) abc Please enter a integer for Plastic Grade Colour Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit

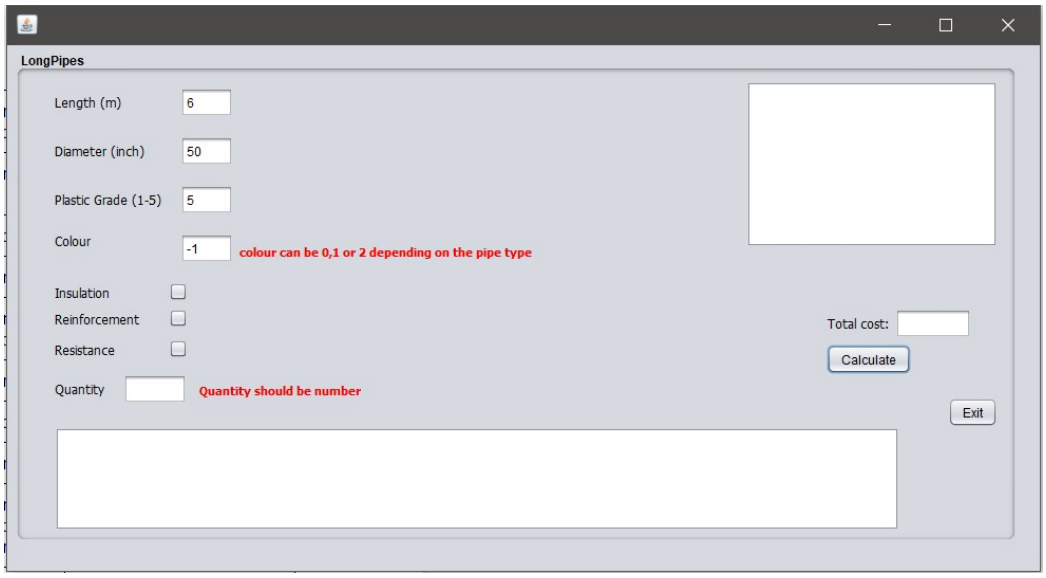
10	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) -1 Plastic Grade has to be between 1 and 5 Colour Colour should be integer Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit
11	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 0 Plastic Grade has to be between 1 and 5 Colour Colour should be integer Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit

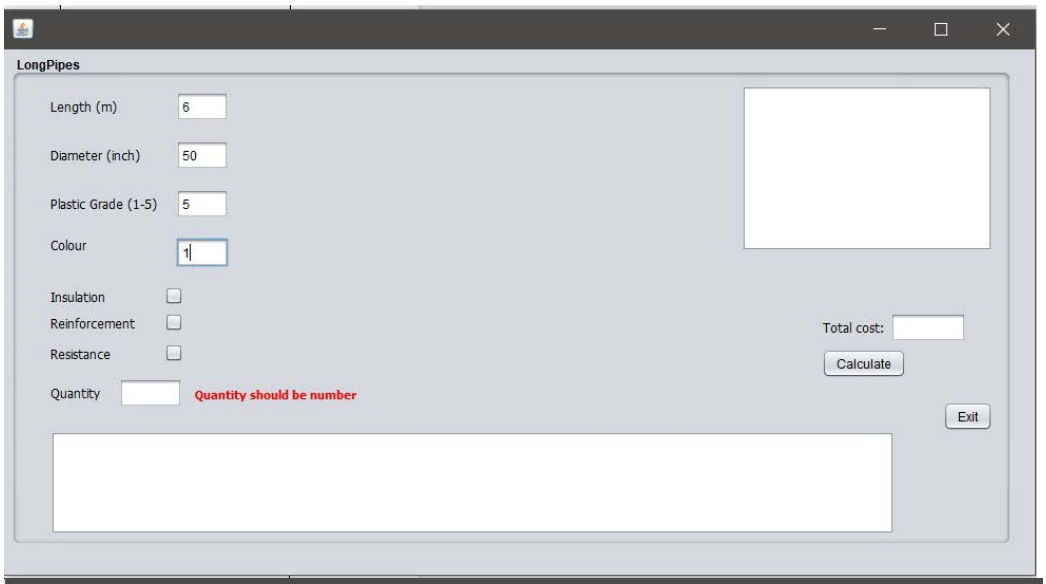

LongPipes
Length (m)
6
Diameter (inch)
50
Plastic Grade (1-5)
1
Colour
Colour should be integer
Insulation
Reinforcement
Resistance
Quantity
Total cost:
Calculate
Exit

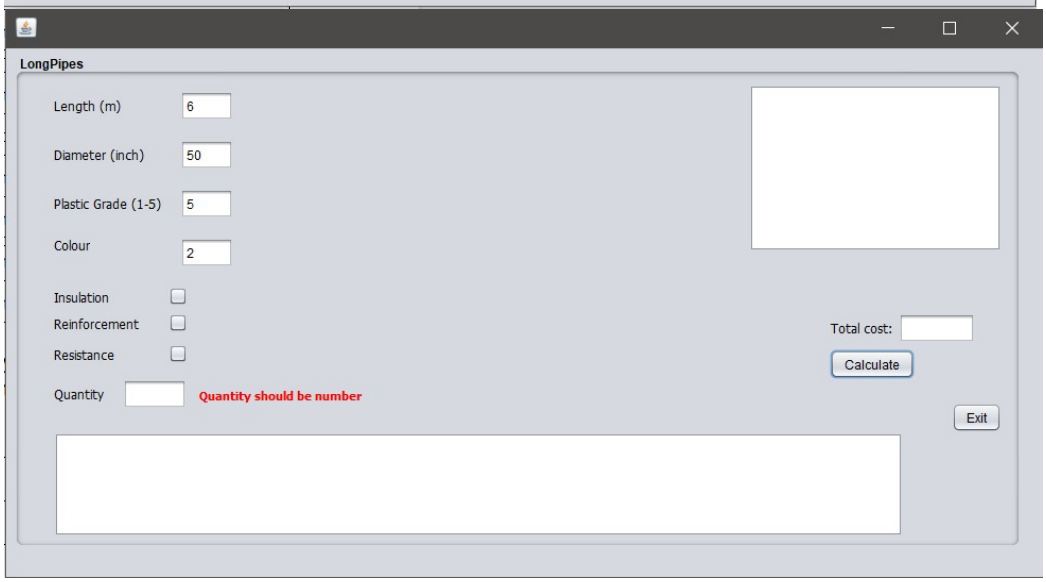

LongPipes
Length (m)
6
Diameter (inch)
50
Plastic Grade (1-5)
5
Colour
Colour should be integer
Insulation
Reinforcement
Resistance
Quantity
Total cost:
Calculate
Exit

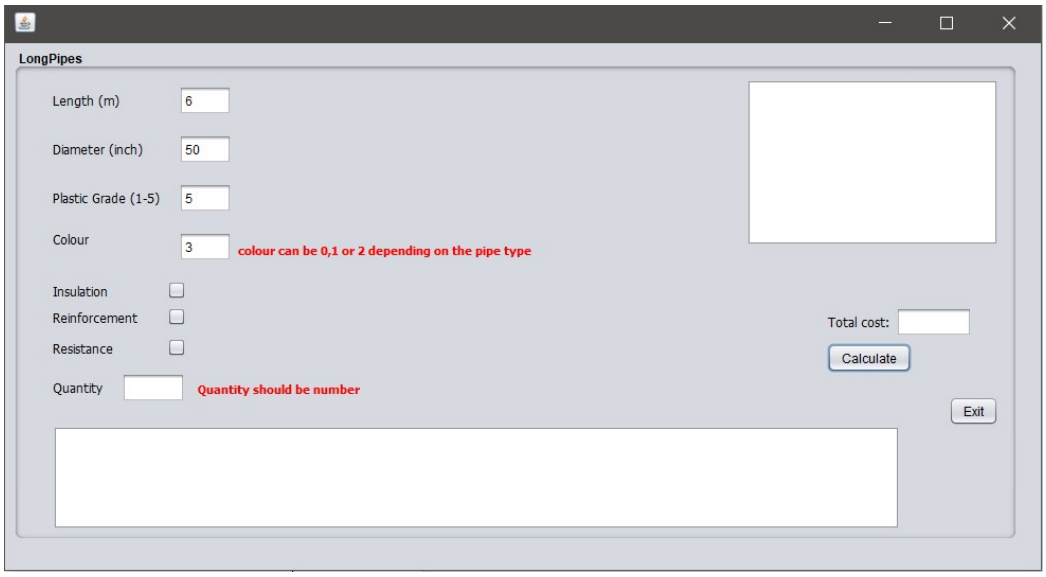

LongPipes
Length (m)
6
Diameter (inch)
50
Plastic Grade (1-5)
6
Plastic Grade has to be between 1 and 5
Colour
Colour should be integer
Insulation
Reinforcement
Resistance
Quantity
Total cost:
Calculate
Exit

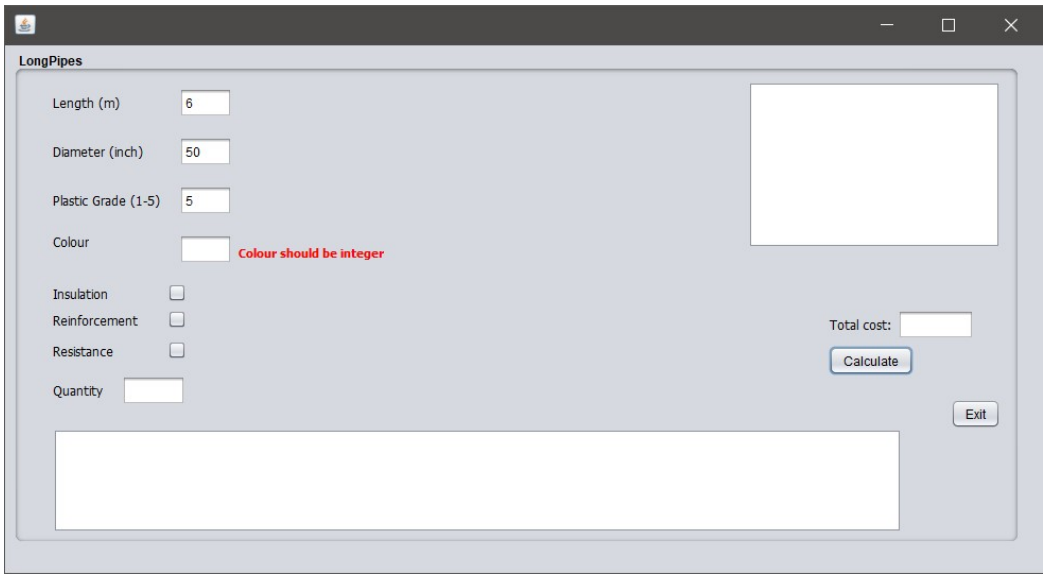
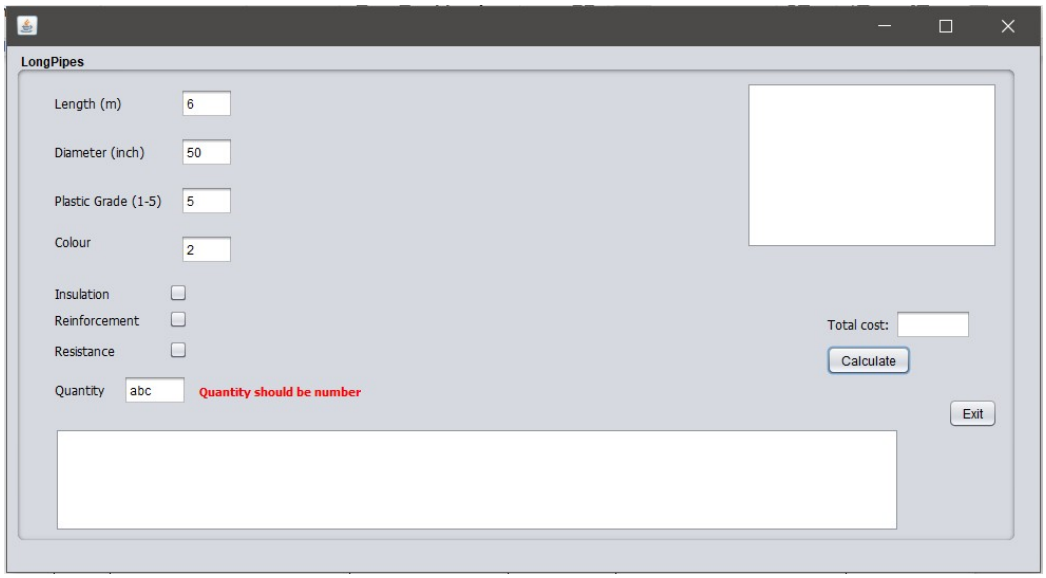
12	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 5 Colour Colour should be integer Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit
13	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 5 Colour abc Colour should be integer Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit

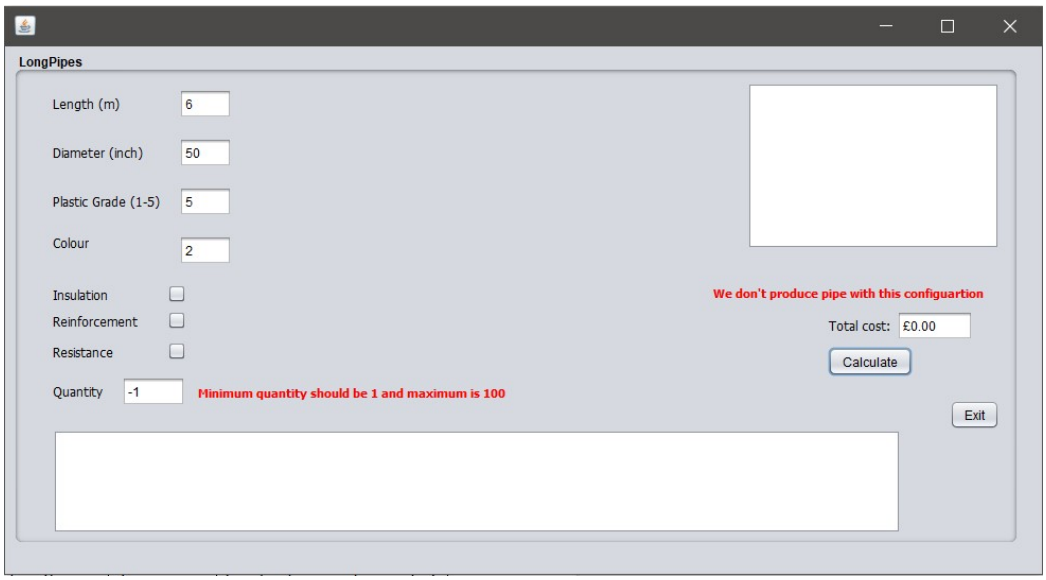
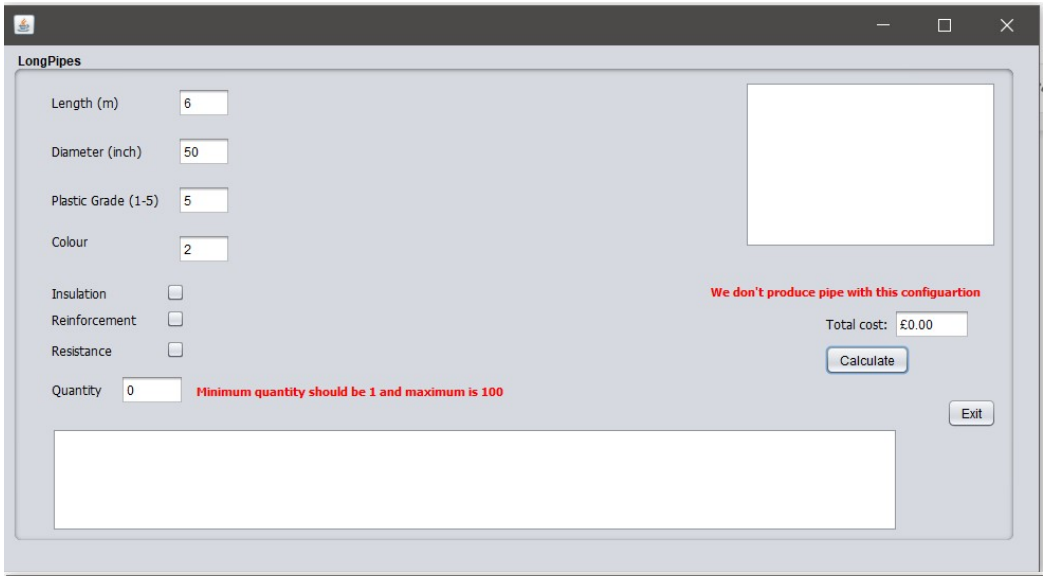
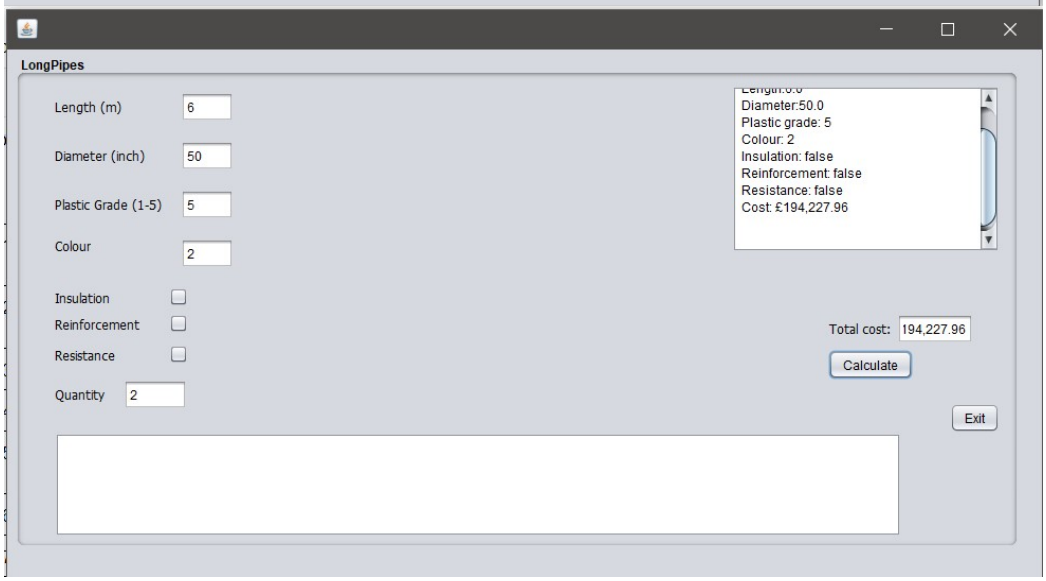
14	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 5 Colour -1 colour can be 0,1 or 2 depending on the pipe type Insulation Reinforcement Resistance Quantity Quantity should be number Total cost: Calculate Exit
15	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 5 Colour 0 Insulation Reinforcement Resistance Quantity Quantity should be number Total cost: Calculate Exit

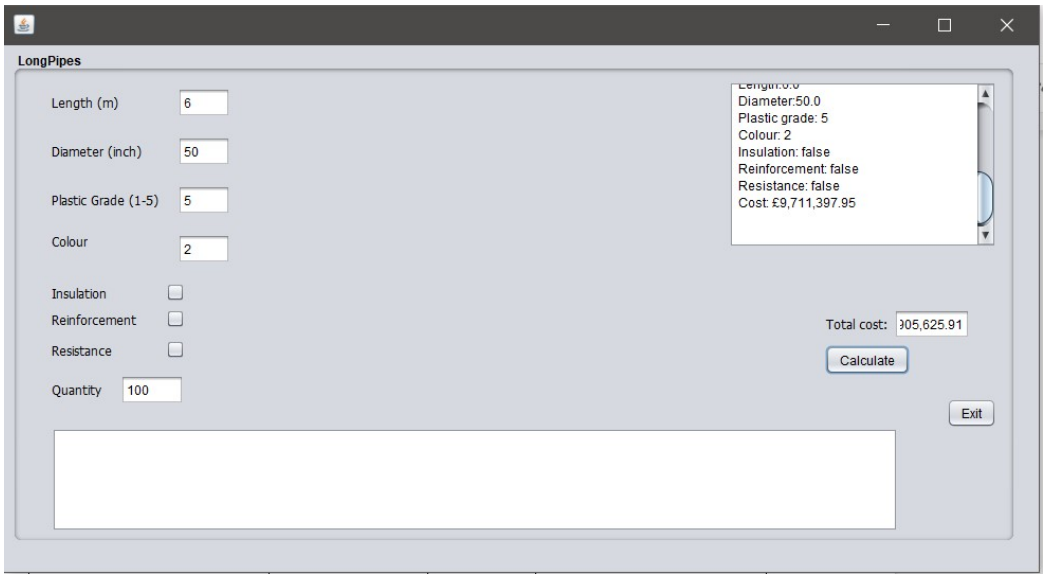

LongPipes
Length (m)
6
Diameter (inch)
50
Plastic Grade (1-5)
5
Colour
1
Insulation
Reinforcement
Resistance
Quantity
Quantity should be number
Total cost:
Calculate
Exit

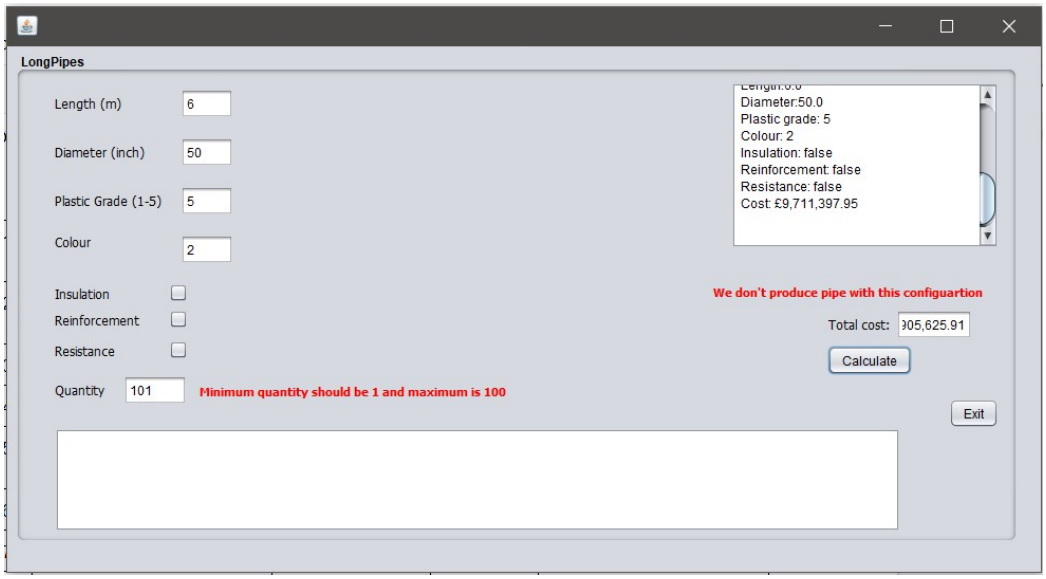

LongPipes
Length (m)
6
Diameter (inch)
50
Plastic Grade (1-5)
5
Colour
2
Insulation
Reinforcement
Resistance
Quantity
Quantity should be number
Total cost:
Calculate
Exit

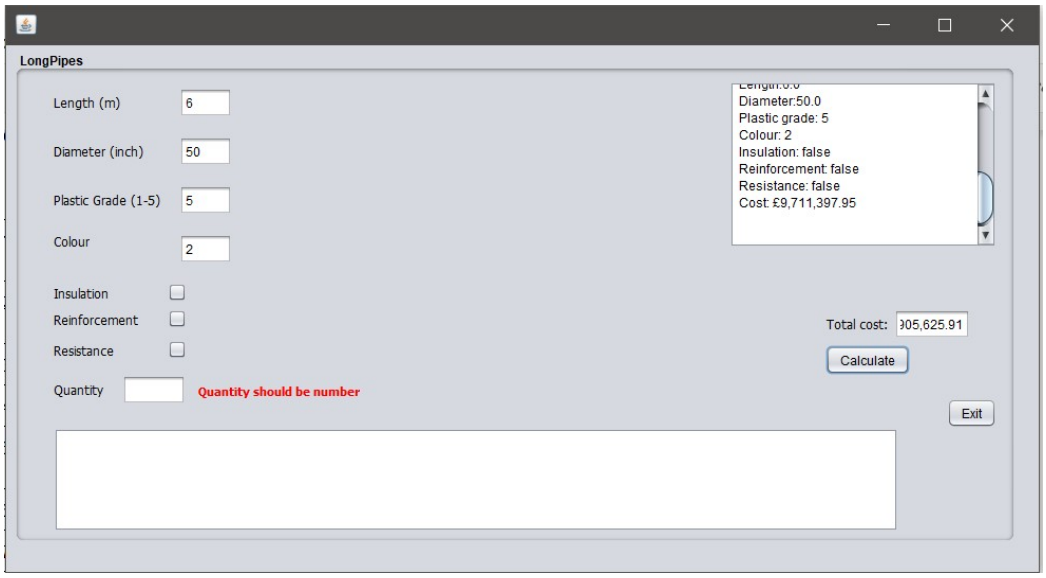

LongPipes
Length (m)
6
Diameter (inch)
50
Plastic Grade (1-5)
5
Colour
3
colour can be 0,1 or 2 depending on the pipe type
Insulation
Reinforcement
Resistance
Quantity
Quantity should be number
Total cost:
Calculate
Exit

16	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 5 Colour Colour should be integer Insulation Reinforcement Resistance Quantity Total cost: Calculate Exit
17	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 5 Colour 2 Insulation Reinforcement Resistance Quantity abc Quantity should be number Total cost: Calculate Exit

18	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 5 Colour 2 Insulation Reinforcement Resistance Quantity -1 Minimum quantity should be 1 and maximum is 100 We don't produce pipe with this configuartion Total cost: £0.00 Calculate Exit
19	 LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 5 Colour 2 Insulation Reinforcement Resistance Quantity 0 Minimum quantity should be 1 and maximum is 100 We don't produce pipe with this configuartion Total cost: £0.00 Calculate Exit  LongPipes Length (m) 6 Diameter (inch) 50 Plastic Grade (1-5) 5 Colour 2 Insulation Reinforcement Resistance Quantity 2 Diameter:50.0 Plastic grade: 5 Colour: 2 Insulation: false Reinforcement: false Resistance: false Cost: £194,227.96 Total cost: 194,227.96 Calculate Exit


LongPipes
Length (m)
6
Diameter (inch)
50
Plastic Grade (1-5)
5
Colour
2
Insulation
Reinforcement
Resistance
Quantity
100
Diameter:50.0
Plastic grade: 5
Colour: 2
Insulation: false
Reinforcement: false
Resistance: false
Cost: £9,711,397.95
Total cost:
Calculate
Exit


LongPipes
Length (m)
6
Diameter (inch)
50
Plastic Grade (1-5)
5
Colour
2
Insulation
Reinforcement
Resistance
Quantity
101
Minimum quantity should be 1 and maximum is 100
Diameter:50.0
Plastic grade: 5
Colour: 2
Insulation: false
Reinforcement: false
Resistance: false
Cost: £9,711,397.95
We don't produce pipe with this configuartion
Total cost:
Calculate
Exit

20	

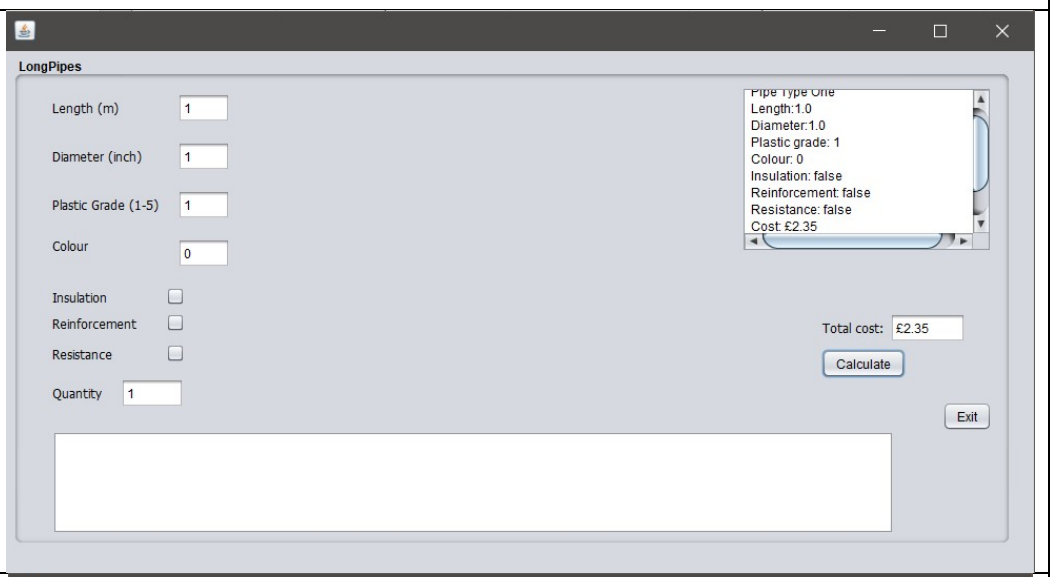
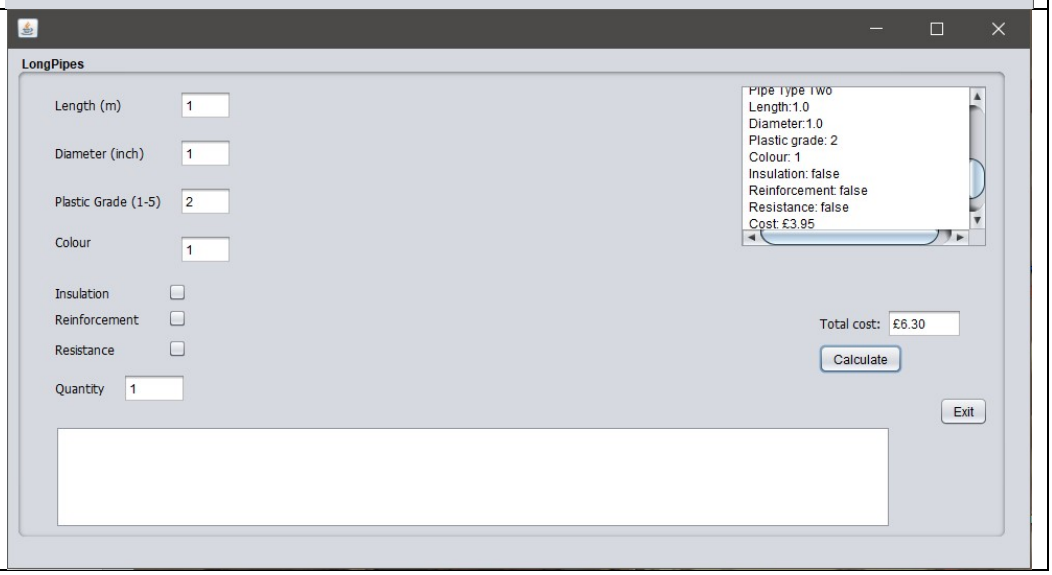
My team also performed extra test to test out different combinations such as 1f on int variables like plastic grade and symbols like “@ to find out whether the system accept it as an input.

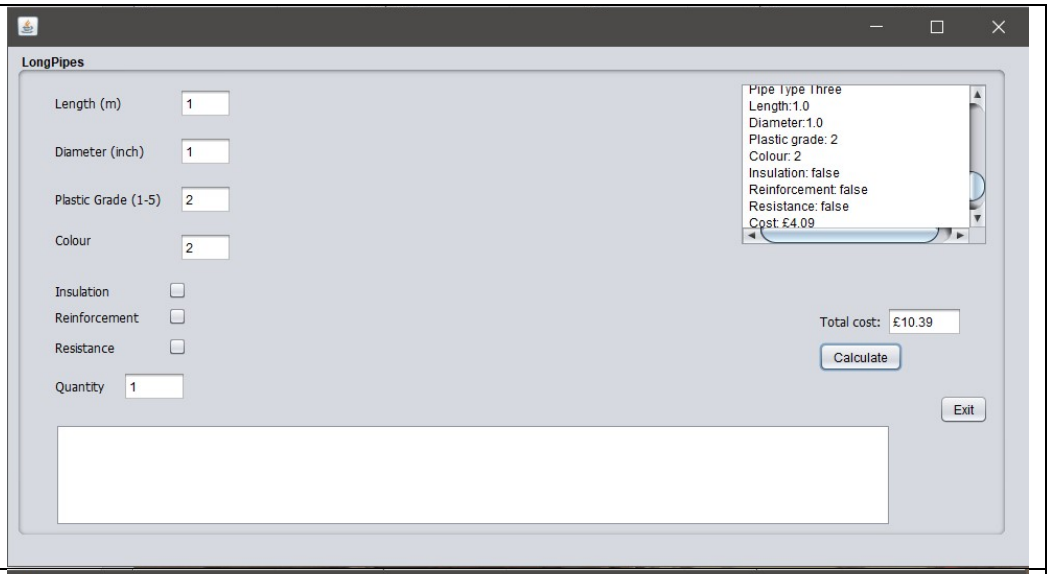
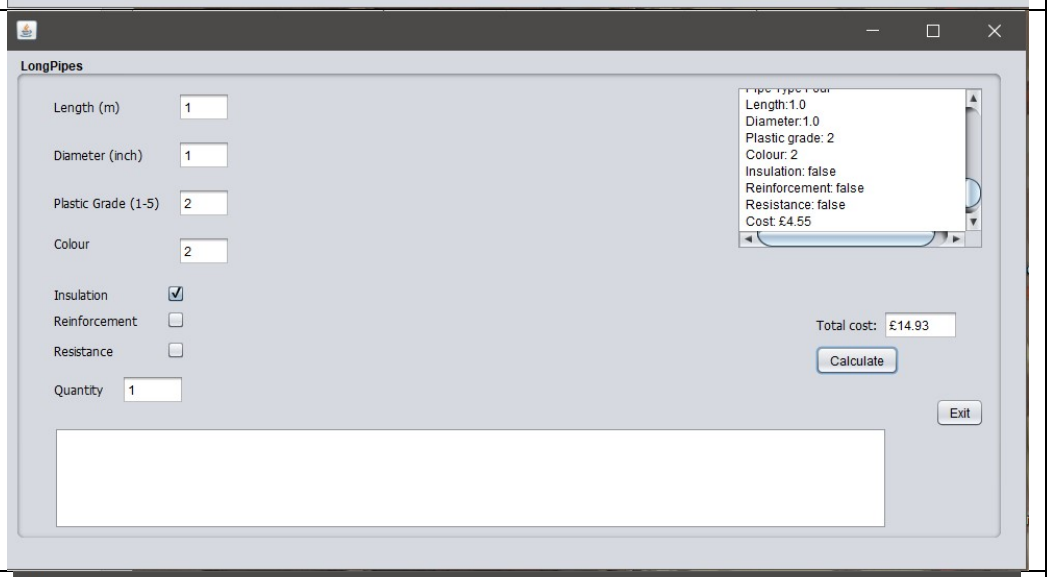
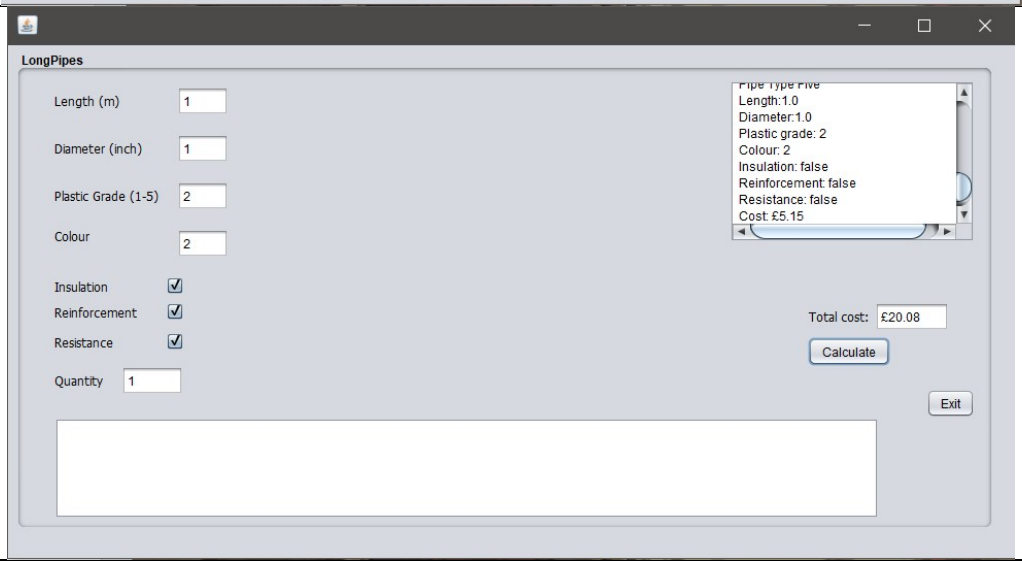
Evaluation

The test schedule was a crucial to the overall development of this application as it allows us, developers, to spot what invalid data is accepted by the system. Validation and verification are the two main thing my team used to limit both the range of each input and the data type of each input.

From all the testing my group have done, it shows that most invalid inputs are rejected and only those that both data type and data value are valid will be accepted. This will ensure the application has a high level of reliability.

Inputs and Outputs

Pipe Type	Screenshot
Pipe one	
Pipe two	

Pipe three	
Pipe four	
Pipe five	

Some sample of inputs are entered to the application to show that the user can order multiple different pipe types and with or without the chemical resistance.

Source Code

PipeInterface

```
package javacoursework;

import java.util.ArrayList;
import java.util.List;
import java.util.Iterator;
import java.text.DecimalFormat;

/**
 *
 * @author up819561, up823183, up822876, up820471
 */
public class PipeInterface extends javax.swing.JFrame {

    /**
     * Creates new form PipeInterface
     */
    public PipeInterface() {
        initComponents();
    }

   //Create a empry arraylist called orderBasket to hold all the total cost
    private final List<Double> orderBasket = new ArrayList<>();

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton2ActionPerformed
        //set the error labels to empty every cal button click
        lenghtErrorLabel.setText("");
        diameterErrorLabel.setText("");
        plasticGradeErrorLabel.setText("");
        quantityErrorLabel.setText("");
        colourErrorLabel.setText("");
        generalErrorLabel.setText("");
        errorLabel.setText("");
        //Initalising the variables to hold the inputs
        double length, diameter, total = 0;
```

```
int plasticGrade, colour, quantity;
boolean insulation = insulationCheckBox.isSelected();
boolean reinforcement = reinforcementCheckBox2.isSelected();
boolean resistance = resistanceCheckBox.isSelected();   // TODO add your handling code here:
boolean isValid;

//Check if any of the input text field is empty
if (jTextField1.getText() == null) {
    lenghtErrorLabel.setText("Length field can't be empty");
    isValid = false;
} else if (jTextField2.getText() == null) {
    diameterErrorLabel.setText("Diameter field can't be empty");
    isValid = false;
} else if (jTextField3.getText() == null) {
    plasticGradeErrorLabel.setText("Plastic grade field can't be empty");
    isValid = false;
} else if (jTextField4.getText() == null) {
    colourErrorLabel.setText("Colour field can't be empty");
    isValid = false;
} else if (jTextField5.getText() == null) {
    quantityErrorLabel.setText("Quntity field can't be");
    isValid = false;
} else {
    isValid = true;
}

//Validating the input(length) data type
try {
    length = Double.parseDouble(jTextField1.getText());

} catch (Exception exRef) {
    lenghtErrorLabel.setText("Please enter a number for Length");
    System.err.println(exRef);
    isValid = false;
    return;
}

//Checking if the input is inbetween the set limit
if (length < 1 || length >= 6) {
    isValid = false;
    lenghtErrorLabel.setText("Length has to be minimum 1 and maximum 6");
    System.err.println("Length smaller than 0");
}

//Validating the input(diameter) data type
try {
    diameter = Double.parseDouble(jTextField2.getText());
} catch (Exception exRef) {
    diameterErrorLabel.setText("Please enter a number for diameter");
    System.err.println(exRef);
    isValid = false;
    return;
}
//Checking if the input is inbetween the set limit
if (diameter < 1 || diameter > 50) {
    isValid = false;
```

```
            diameterErrorLabel.setText("Diameter must be inbetween 1 and 50");
            System.err.println("diameter smaller than 0");

        }

        //Validating the input(plasticGrade) data type
        try {
            plasticGrade = Integer.parseInt(jTextField3.getText());
        } catch (Exception exRef) {
            plasticGradeErrorLabel.setText("Please enter a integer  for Plastic Grade");
            System.err.println(exRef);
            isValid = false;
            return;
        }
        //Checking if the input is inbetween the set limit
        if (plasticGrade <= 0 || plasticGrade > 5) {
            isValid = false;
            plasticGradeErrorLabel.setText("Plastic Grade has to be between 1 and 5");
        }

        //Validating the input(colour) data type
        try {
            colour = Integer.parseInt(jTextField4.getText());
        } catch (Exception exRef) {
            colourErrorLabel.setText("Colour should be integer");
            System.err.println(exRef);
            isValid = false;
            return;
        }
        //Checking if the input is inbetween the set limit
        if (colour < 0 || colour > 2) {
            isValid = false;
            colourErrorLabel.setText("colour can be 0,1 or 2 depending on the pipe type");
            System.err.println("colour can be 0,1 or 2 depending on the pipe type");
        }
        if (plasticGrade == 1 && colour != 0) {
            colourErrorLabel.setText("Plastic grade must be greater than 1 to have multiple colours");
        }
        //Validating the input(quantity) data type
        try {
            quantity = Integer.parseInt(jTextField5.getText());
        } catch (Exception exRef) {
            quantityErrorLabel.setText("Quantity should be number");
            System.err.println(exRef);
            isValid = false;
            return;
        }
        //Checking if the input is inbetween the set limit
        if (quantity <= 0 || quantity > 100) {
            isValid = false;
            quantityErrorLabel.setText("Minimum quantity should be 1 and maximum is 100");
            System.err.println("Minimum quantity should be 1");
        }
        String message = getErrorMessages(plasticGrade, colour, insulation, reinforcement, length, diameter,
quantity);
        //Perform calculate pipe type method if all the inputs are valid and add to orderBasket plus print out the info
```

```
        if (isValid) {
            if (plasticGrade >= 1 && plasticGrade <= 3 && colour == 0 && insulation == false && reinforcement ==
false) {
                pipeOne pipe1 = new pipeOne(length, diameter, plasticGrade, colour, insulation, reinforcement,
resistance, quantity);
                jTextArea1.append(pipe1.printInfo());
                orderBasket.add(pipe1.calcTotalCost());
            } else if (plasticGrade >= 2 && plasticGrade <= 4 && colour == 1 && insulation == false && reinforcement
== false) {
                pipeTwo pipe2 = new pipeTwo(length, diameter, plasticGrade, colour, insulation, reinforcement,
resistance, quantity);
                jTextArea1.append(pipe2.printInfo());
                orderBasket.add(pipe2.calcTotalCost());
            } else if (plasticGrade >= 2 && plasticGrade <= 5 && colour == 2 && insulation == false && reinforcement
== false) {
                pipeThree pipe3 = new pipeThree(length, diameter, plasticGrade, colour, insulation, reinforcement,
resistance, quantity);
                jTextArea1.append(pipe3.printInfo());
                orderBasket.add(pipe3.calcTotalCost());
            } else if (plasticGrade >= 2 && plasticGrade <= 5 && colour == 2 && insulation == true && reinforcement
== false) {
                pipeFour pipe4 = new pipeFour(length, diameter, plasticGrade, colour, insulation, reinforcement,
resistance, quantity);
                jTextArea1.append(pipe4.printInfo());
                orderBasket.add(pipe4.calcTotalCost());
            } else if (plasticGrade >= 2 && plasticGrade <= 5 && colour == 2 && insulation == true && reinforcement
== true) {
                pipeFive pipe5 = new pipeFive(length, diameter, plasticGrade, colour, insulation, reinforcement,
resistance, quantity);
                jTextArea1.append(pipe5.printInfo());
                orderBasket.add(pipe5.calcTotalCost());
            } else {
                errorLabel.setText(message);
                generalErrorLabel.setText("We don't produce pipe with this configuartion ");
            }
        } else {
            generalErrorLabel.setText("We don't produce pipe with this configuartion ");
        }
        //loop through the list to find to total cost of all the orders
        Iterator<Double> iterator = orderBasket.iterator();
        while (iterator.hasNext()) {
            total += iterator.next();
        }

        DecimalFormat df = new DecimalFormat("£#,##0.00");
        jTextField6.setText(df.format(total));


    }//GEN-LAST:event_jButton2ActionPerformed

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jTextField1ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jTextField1ActionPerformed

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
```

```
FIRST:event_jButton1ActionPerformed
        System.exit(0);// TODO add your handling code here:
    }//GEN-LAST:event_jButton1ActionPerformed

    private void jTextField6ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jTextField6ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jTextField6ActionPerformed

    private void insulationCheckBoxActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_insulationCheckBoxActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_insulationCheckBoxActionPerformed

    private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jTextField3ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jTextField3ActionPerformed

    //Method to get the correct error message depends on the invalid input
    public String getErrorMessages(int plasticGrade, int colour, boolean insulation, boolean reinforcement, double
length, double diameter, int quantity) {
        String message = "";
        if (plasticGrade == 5 && colour != 2) {
            message += "Colour must be 2 if the plastic grade is 5\n";
        }
        if (plasticGrade >= 4 && colour == 0) {
            message += "Must have 2 colours if the plastic grade is higher than 3\n";
        }
        if (colour != 2 && (insulation == true || reinforcement == true)) {
            message += "Must have 2 colours to have inner insulation or outer reinforcement extras\n";
        }
        if (plasticGrade < 3 && reinforcement == true) {
            message += "Plastic grade must be greater than 2 to have outer reinforcement\n";
        }
        if (insulation == false && reinforcement == true) {
            message += "Must have inner insluation to have outer reinforcement\n";
        }
        return message;
    }
```

pipeAbstract

```
package javacoursework;

import java.text.DecimalFormat;

/**
 *
 * @author up819561, up823183, up822876, up820471
 */
public abstract class pipeAbstract {

    //Create varibles for holding the pipe info and calculation
    protected double length, diameter, radius, volume, cost, basicCost, addInsulation, addReinforcement,
addResistance, addColour;
```

```
    protected int grade, colour, quantity;
    protected boolean insulation, reinforcement, resistance;
    protected double[] costPerInch = {0.4, 0.6, 0.75, 0.8, 0.95};
    protected double[] colourCost = {0, 0.12, 0.16};
    protected int pipeType;

    //Empty constructor
    public pipeAbstract() {
    }

    //Constructor
    public pipeAbstract(double pipeLength, double pipeDiameter, int pipeGrade, int pipeColour, boolean
pipeInsulation, boolean pipeReinforcement, boolean pipeResistence, int pipeQuantity) {
        length = pipeLength * 39.37;    //meters to inches
        diameter = pipeDiameter;
        radius = diameter / 2;
        grade = pipeGrade;
        colour = pipeColour;
        insulation = pipeInsulation;
        reinforcement = pipeReinforcement;
        resistance = pipeResistence;
        quantity = pipeQuantity;
    }

    //Accessors - return the corresponding varilbe
    public double getLength() {
        return length / 39.37;
    }

    public double getDiameter() {
        return diameter;
    }

    public int getGrade() {
        return grade;
    }

    public int getColour() {
        return colour;
    }

    public boolean getInsulation() {
        return insulation;
    }

    public boolean getReinforcement() {
        return reinforcement;
    }

    public boolean getResistance() {
        return resistance;
    }

    public int getQuantity() {
        return quantity;
    }
```

```
    public double getRadius() {
        return radius;
    }

    public double getVolume() {
        return volume;
    }

    public double getCost() {
        return cost;
    }

    //OTHER METHODS
    //Print info to the textArea box
    public String printInfo() {
        double price = calcTotalCost();
        DecimalFormat df = new DecimalFormat("£#,##0.00");
        return getType() + "\n"+"Length:"+getLength()+ "\n"+"Diameter:"+getDiameter()+ "\n"+"Plastic grade:
"+getGrade()+ "\n"+"Colour: "+getColour()+ "\n"+"Insulation: "+getInsulation()+"\n"+"Reinforcement:
"+getReinforcement()+ "\n"+"Resistance: "+getResistance()+ "\n"+"Cost: "+df.format(price) + "\n\n";
    }

    //Calculate and return the pipe volume
    public double calcVolume(double length, double radius) {
        //Whole volume  - inner volume
        volume = ((Math.PI * Math.pow(radius, 2) * length) - (Math.PI * Math.pow(radius * 0.9, 2) * length));
        return volume;
    }

    //Calculate and return the basic pipe cost (base cost * quantity)
    public double calcBasicCost() {
        this.volume = calcVolume(length, radius);
        basicCost = volume * costPerInch[grade - 1];
        return basicCost * quantity;
    }

    //Return the pipeType
    public abstract String getType();

    //Return the Total cost
    public abstract double calcTotalCost();
}
```

pipeOne

```
package javacoursework;

/**
 *
 * @author up819561, up823183, up822876, up820471
 */
public class pipeOne extends pipeAbstract {

    //Creates a new instance of Pipe Type 1
    public pipeOne() {
    }
```

```
    //Constructor
    public pipeOne(double pipeLength, double pipeDiameter, int pipeGrade, int pipeColour, boolean
pipeInsulation, boolean pipeReinforcement, boolean pipeResistence, int pipeQuantity) {
        super(pipeLength, pipeDiameter, pipeGrade, 0, false, false, pipeResistence, pipeQuantity);
    }

    // overide the method to return the total cost of pipeOne with or without the Resistence plus any additonal cost
    @Override
    public double calcTotalCost() {
        basicCost = calcBasicCost();
        addColour = basicCost * colourCost[colour];
        if (getResistance() == true) {
            addResistance = basicCost * 0.14;
            cost = basicCost + addResistance;
        } else {
            cost = basicCost;
        }
        return cost;
    }

    // overide the metheod to return the name, pipeOne
    @Override
    public String getType() {
        return "Pipe Type One";
    }
}
```

pipeTwo

```
package javacoursework;

/**
 *
 * @author up819561, up823183, up822876, up820471
 */
public class pipeTwo extends pipeAbstract {

    //Creates a new instance of Pipe Type 1
    public pipeTwo() {
    }

    //Constructor
    public pipeTwo(double pipeLength, double pipeDiameter, int pipeGrade, int pipeColour, boolean
pipeInsulation, boolean pipeReinforcement, boolean pipeResistence, int pipeQuantity) {
        super(pipeLength, pipeDiameter, pipeGrade, 1, false, false, pipeReinforcement, pipeQuantity);
    }

    // overide the method to return the total cost of pipeTwo with or without the Resistence plus any additonal cost
    @Override
    public double calcTotalCost() {
        basicCost = calcBasicCost();
        addColour = basicCost * colourCost[colour];
        if (getResistance() == true) {
            addResistance = basicCost * 0.14;
            cost = basicCost + addResistance + addColour;
        } else {
```

```
            cost = basicCost + addColour;
        }
        return cost;
    }

    // overide the metheod to return the name, pipeTwo
    @Override
    public String getType() {
        return "Pipe Type Two";
    }
}
```

pipeThree

```
package javacoursework;

/**
 *
 * @author up819561, up823183, up822876, up820471
 */
public class pipeThree extends pipeAbstract {

    //Creates a new instance of Pipe Type 1
    public pipeThree() {
    }

    //Constructor
    public pipeThree(double pipeLength, double pipeDiameter, int pipeGrade, int pipeColour, boolean
pipeInsulation, boolean pipeReinforcement, boolean pipeResistence, int pipeQuantity) {
        super(pipeLength, pipeDiameter, pipeGrade, 2, false, false, pipeReinforcement, pipeQuantity);
    }

    // overide the method to return the total cost of pipeThree with or without the Resistence plus any additonal
cost
    @Override
    public double calcTotalCost() {
        basicCost = calcBasicCost();
        addColour = basicCost * colourCost[colour];
        if (getResistance() == true) {
            addResistance = basicCost * 0.14;
            cost = basicCost + addResistance + addColour;
        } else {
            cost = basicCost + addColour;
        }
        return cost;
    }

    // overide the metheod to return the name, pipeThree
    @Override
    public String getType() {
        return "Pipe Type Three";
    }
```

```
}
```

pipeFour

```
package javacoursework;

/**
 *
 * @author up819561, up823183, up822876, up820471
 */
public class pipeFour extends pipeThree {

    //Creates a new instance of Pipe Type 1
    public pipeFour() {
    }

    //Constructor
    public pipeFour(double pipeLength, double pipeDiameter, int pipeGrade, int pipeColour, boolean
pipeInsulation, boolean pipeReinforcement, boolean pipeResistence, int pipeQuantity) {
        super(pipeLength, pipeDiameter, pipeGrade, 2, true, false, pipeReinforcement, pipeQuantity);
    }

    // overide the method to return the total cost of pipeFour with or without the Resistence plus any additonal cost
    @Override
    public double calcTotalCost() {
        basicCost = calcBasicCost();
        addColour = basicCost * colourCost[colour];
        addInsulation = basicCost * 0.13;
        if (getResistance() == true) {
            addResistance = basicCost * 0.14;
            cost = basicCost + addResistance + addColour + addInsulation;
        } else {
            cost = basicCost + addColour + addInsulation;
        }
        return cost;
    }

    // overide the metheod to return the name, pipeFour
    @Override
    public String getType() {
        return "Pipe Type Four";
    }
}
```

pipeFive

```
package javacoursework;

/**
 *
 * @author up819561, up823183, up822876, up820471
 */
public class pipeFive extends pipeFour {

    //Creates a new instance of Pipe Type 1
    public pipeFive() {
```

```
    }

    //Constructor
    public pipeFive(double pipeLength, double pipeDiameter, int pipeGrade, int pipeColour, boolean
pipeInsulation, boolean pipeReinforcement, boolean pipeResistence, int pipeQuantity) {
        super(pipeLength, pipeDiameter, pipeGrade, 2, true, true, pipeResistence, pipeQuantity);
    }

    // overide the method to return the total cost of pipeFive with or without the Resistence plus any additonal cost
    @Override
    public double calcTotalCost() {
        basicCost = calcBasicCost();
        addColour = basicCost * colourCost[colour];
        addInsulation = basicCost * 0.13;
        addReinforcement = basicCost * 0.17;
        if (getResistance() == true) {
            addResistance = basicCost * 0.14;
            cost = basicCost + addResistance + addColour + addInsulation + addReinforcement;
        } else {
            cost = basicCost + addColour + addInsulation + addReinforcement;
        }
        return cost;
    }

    // overide the metheod to return the name, pipeFive
    @Override
    public String getType() {
        return "Pipe Type Five";
    }
}
```

Coursework Specification

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 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:

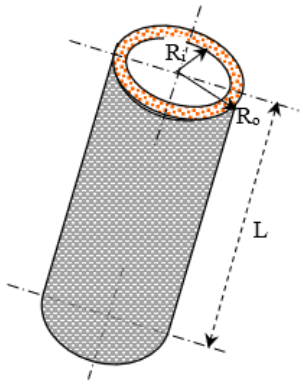


Fig.1. Pipe's cross-section.

- They are all made of plastic;
- 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.

Table 1. *Types of plastic pipes available.*

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

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 R_o) – in inches (1" = 0.0254m). Assume the inner diameter (or the inner radius R_i) 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 inch^3 of plastic is given in Table 2.

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

Grade of plastic	1	2	3	4	5
Cost per inch^3 [in £]	0.4	0.6	0.75	0.8	0.95

Table 3. *Additional costs.*

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

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 (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 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	

Group contribution form

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.

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 give her/him 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 *GrC-7*

Student number and contribution:

1. UP822876 33/100

2. UP819561 33/100

3. UP823183 33/100

4. UP820471 1/100

TOTAL 100/100