using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Project3VariousGeometryFormulaCalculator

{

public partial class FormGeometryCalculator : Form

{

/\*Author: Gustavo Marin Borges

Title: GeometryFormulaCalculator\*/

public FormGeometryCalculator()

{

InitializeComponent();

}

private void buttonCalculate\_Click(object sender, EventArgs e)

{

//Not used

}

//Clear button, restarts the app to manage the changes in shows and hides

private void buttonClear\_Click(object sender, EventArgs e)

{

Application.Restart();

Environment.Exit(0);

}

//Calculate button

private void buttonCalculate\_Click\_1(object sender, EventArgs e)

{

//Data validation for Errors and Warnings in case the user chooses a single group starts

//Data validation reminder starts

if (radioButtonRectangle.Checked == false && radioButtonPolygon.Checked == false && radioButtonKite.Checked == false && radioButtonParallelogram.Checked == false && radioButtonCircle.Checked == false && radioButtonTrapezoid.Checked == false && radioButtonTriangle.Checked == false && radioButtonRhombus.Checked == false && radioButtonSquare.Checked == false && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("There is no selection!", "No Selection Reminder", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

//Data validation reminder ends

//Data validation Warnings starts

else if (radioButtonRectangle.Checked == true && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("No selection for area and perimeter!", "No Selection Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if (radioButtonSquare.Checked == true && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("No selection for area and perimeter!", "No Selection Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if (radioButtonRhombus.Checked == true && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("No selection for area and perimeter!", "No Selection Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if (radioButtonTriangle.Checked == true && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("No selection for area and perimeter!", "No Selection Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if (radioButtonTrapezoid.Checked == true && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("No selection for area and perimeter!", "No Selection Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if (radioButtonCircle.Checked == true && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("No selection for area and perimeter!", "No Selection Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if (radioButtonParallelogram.Checked == true && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("No selection for area and perimeter!", "No Selection Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if (radioButtonKite.Checked == true && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("No selection for area and perimeter!", "No Selection Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if (radioButtonPolygon.Checked == true && radioButtonArea.Checked == false && radioButtonPerimeter.Checked == false && radioButtonAreaPerimeter.Checked == false)

{

MessageBox.Show("No selection for area and perimeter!", "No Selection Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

//Data validation Warnings ends

//Data validation Errors starts

else if (radioButtonArea.Checked == true && radioButtonRectangle.Checked == false && radioButtonSquare.Checked == false && radioButtonRhombus.Checked == false && radioButtonTriangle.Checked == false && radioButtonTrapezoid.Checked == false && radioButtonCircle.Checked == false && radioButtonParallelogram.Checked == false && radioButtonKite.Checked == false && radioButtonPolygon.Checked == false)

{

MessageBox.Show("Shape is not selected!", "No Selection Warning", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (radioButtonPerimeter.Checked == true && radioButtonRectangle.Checked == false && radioButtonSquare.Checked == false && radioButtonRhombus.Checked == false && radioButtonTriangle.Checked == false && radioButtonTrapezoid.Checked == false && radioButtonCircle.Checked == false && radioButtonParallelogram.Checked == false && radioButtonKite.Checked == false && radioButtonPolygon.Checked == false)

{

MessageBox.Show("Shape is not selected!", "No Selection Warning", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (radioButtonAreaPerimeter.Checked == true && radioButtonRectangle.Checked == false && radioButtonSquare.Checked == false && radioButtonRhombus.Checked == false && radioButtonTriangle.Checked == false && radioButtonTrapezoid.Checked == false && radioButtonCircle.Checked == false && radioButtonParallelogram.Checked == false && radioButtonKite.Checked == false && radioButtonPolygon.Checked == false)

{

MessageBox.Show("Shape is not selected!", "No Selection Warning", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

//Data validation Errors ends

//Data validation for Errors and Warnings in case the user chooses a single group ends

//Shows rectangle area

else if (radioButtonRectangle.Checked == true && radioButtonArea.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxRectangle.Show();

pictureBoxFormulaRectangle.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows rectangle perimeter

else if (radioButtonRectangle.Checked == true && radioButtonPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxRectangle.Show();

pictureBoxFormulaRectangle.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows rectangle area&perimater

else if (radioButtonRectangle.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxRectangle.Show();

pictureBoxFormulaRectangle.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows square area

else if (radioButtonSquare.Checked == true && radioButtonArea.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxSquare.Show();

pictureBoxFormulaSquare.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows square perimeter

else if (radioButtonSquare.Checked == true && radioButtonPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxSquare.Show();

pictureBoxFormulaSquare.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows square area&perimeter

else if (radioButtonSquare.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxSquare.Show();

pictureBoxFormulaSquare.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows rhombus area

else if (radioButtonRhombus.Checked == true && radioButtonArea.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxRhombus.Show();

pictureBoxFormulaRhombus.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows rhombus perimeter

else if (radioButtonRhombus.Checked == true && radioButtonPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxRhombus.Show();

pictureBoxFormulaRhombus.Show();

groupBoxInputData.Show();

labelInputC.Show();

textBoxDisplayInputC.Show();

}

//Shows rhombus area&perimeter

else if (radioButtonRhombus.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxRhombus.Show();

pictureBoxFormulaRhombus.Show();

groupBoxInputData.Show();

labelInputC.Show();

textBoxDisplayInputC.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows triangle area

else if (radioButtonTriangle.Checked == true && radioButtonArea.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxTriangle.Show();

pictureBoxFormulaTriangle.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows triangle perimeter

else if (radioButtonTriangle.Checked == true && radioButtonPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxTriangle.Show();

pictureBoxFormulaTriangle.Show();

groupBoxInputData.Show();

labelInputC.Show();

labelInputB.Show();

labelInputD.Show();

textBoxDisplayInputC.Show();

textBoxDisplayInputD.Show();

textBoxDisplayInputB.Show();

}

//Shows triangle area&perimeter

else if (radioButtonTriangle.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxTriangle.Show();

pictureBoxFormulaTriangle.Show();

groupBoxInputData.Show();

labelInputC.Show();

textBoxDisplayInputC.Show();

labelInputA.Show();

labelInputC.Show();

labelInputB.Show();

labelInputD.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputC.Show();

textBoxDisplayInputD.Show();

}

//Shows trapezoid area

else if (radioButtonTrapezoid.Checked == true && radioButtonArea.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxTrapezoid.Show();

pictureBoxFormulaTrapezoid2.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

labelInputC.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputC.Show();

}

//Shows trapezoid perimeter

else if (radioButtonTrapezoid.Checked == true && radioButtonPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxTrapezoid.Show();

pictureBoxFormulaTrapezoid2.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

labelInputD.Show();

labelInputE.Show();

textBoxDisplayInputD.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputE.Show();

}

//Shows trapezoid area&perimeter

else if (radioButtonTrapezoid.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxTrapezoid.Show();

pictureBoxFormulaTrapezoid2.Show();

groupBoxInputData.Show();

labelInputC.Show();

textBoxDisplayInputC.Show();

labelInputA.Show();

labelInputC.Show();

labelInputB.Show();

labelInputD.Show();

labelInputE.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputC.Show();

textBoxDisplayInputD.Show();

textBoxDisplayInputE.Show();

}

//Shows circle area

else if (radioButtonCircle.Checked == true && radioButtonArea.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxCircle.Show();

pictureBoxFormulaCircle.Show();

groupBoxInputData.Show();

labelInputA.Show();

textBoxDisplayInputA.Show();

}

//Shows circle perimeter

else if (radioButtonCircle.Checked == true && radioButtonPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxCircle.Show();

pictureBoxFormulaCircle.Show();

groupBoxInputData.Show();

labelInputA.Show();

textBoxDisplayInputA.Show();

}

//Shows circle area&perimeter

else if (radioButtonCircle.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxCircle.Show();

pictureBoxFormulaCircle.Show();

groupBoxInputData.Show();

labelInputA.Show();

textBoxDisplayInputA.Show();

}

//Shows parallelogram area

else if (radioButtonParallelogram.Checked == true && radioButtonArea.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxParallelogram.Show();

pictureBoxFormulaParallelogram.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows parallelogram perimeter

else if (radioButtonParallelogram.Checked == true && radioButtonPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxParallelogram.Show();

pictureBoxFormulaParallelogram.Show();

groupBoxInputData.Show();

labelInputB.Show();

labelInputC.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputC.Show();

}

//Shows parallelogram area&perimeter

else if (radioButtonParallelogram.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxParallelogram.Show();

pictureBoxFormulaParallelogram.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

labelInputC.Show();

textBoxDisplayInputC.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputA.Show();

}

//Shows kite area

else if (radioButtonKite.Checked == true && radioButtonArea.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxKite.Show();

pictureBoxFormulaKite.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

}

//Shows kite perimeter

else if (radioButtonKite.Checked == true && radioButtonPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxKite.Show();

pictureBoxFormulaKite.Show();

groupBoxInputData.Show();

labelInputD.Show();

labelInputC.Show();

textBoxDisplayInputD.Show();

textBoxDisplayInputC.Show();

}

//Shows kite area&perimeter

else if (radioButtonKite.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxKite.Show();

pictureBoxFormulaKite.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

labelInputC.Show();

labelInputD.Show();

textBoxDisplayInputC.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputD.Show();

}

//Shows polygon area

else if (radioButtonPolygon.Checked == true && radioButtonArea.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxPolygon.Show();

pictureBoxFormulaPolygon.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

labelInputC.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputC.Show();

}

//Shows polygon perimeter

else if (radioButtonPolygon.Checked == true && radioButtonPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxPolygon.Show();

pictureBoxFormulaPolygon.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

labelInputC.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputC.Show();

}

//Shows polygon area&perimeter

else if (radioButtonPolygon.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

buttonCalculate.Hide();

buttonAnswer.Show();

pictureBoxPolygon.Show();

pictureBoxFormulaPolygon.Show();

groupBoxInputData.Show();

labelInputA.Show();

labelInputB.Show();

labelInputC.Show();

textBoxDisplayInputA.Show();

textBoxDisplayInputB.Show();

textBoxDisplayInputC.Show();

}

}

//Answer button

private void buttonAnswer\_Click(object sender, EventArgs e)

{

//Input validation for rectangle - area

if (radioButtonRectangle.Checked == true && radioButtonArea.Checked == true)

{

double VarAreaA = 0, VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0)

{

double VarArea = VarAreaA \* VarAreaB;

labelDisplayResult.Text = "The result of the calculation for the area of a rectangle is " + VarArea;

}

}

//Input validation for rectangle - perimeter

else if (radioButtonRectangle.Checked == true && radioButtonPerimeter.Checked == true)

{

double VarPerimeterA = 0;

double VarPerimeterB = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarPerimeterA != 0 && VarPerimeterB != 0)

{

double VarPerimeter = (VarPerimeterA \* 2) + (VarPerimeterB \* 2);

labelDisplayResult.Text = "The result of the calculation for the perimeter of a rectangle is " + VarPerimeter;

}

}

//Input validation for rectangle - area&perimeter

else if (radioButtonRectangle.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

double VarAreaA = 0, VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeterA = 0;

double VarPerimeterB = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarPerimeterA != 0 && VarPerimeterB != 0)

{

double VarArea = VarAreaA \* VarAreaB;

double VarPerimeter = (VarPerimeterA \* 2) + (VarPerimeterB \* 2);

labelDisplayResult.Text = "The result of the calculation for the area of a rectangle is " + VarArea + " and its perimeter is " + VarPerimeter;

}

}

//Input validation for square - area

else if (radioButtonSquare.Checked == true && radioButtonArea.Checked == true)

{

double VarAreaA = 0, VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0)

{

double VarArea = VarAreaA \* VarAreaB;

labelDisplayResult.Text = "The result of the calculation for the area of a Square is " + VarArea;

}

}

//Input validation for square - perimeter

else if (radioButtonSquare.Checked == true && radioButtonPerimeter.Checked == true)

{

double VarPerimeterA = 0;

double VarPerimeterB = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarPerimeterA != 0 && VarPerimeterB != 0)

{

double VarPerimeter = (VarPerimeterA \* 2) + (VarPerimeterB \* 2);

labelDisplayResult.Text = "The result of the calculation for the perimeter of a Square is " + VarPerimeter;

}

}

//Input validation for square - area&perimeter

else if (radioButtonSquare.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

double VarAreaA = 0, VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeterA = 0;

double VarPerimeterB = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarPerimeterA != 0 && VarPerimeterB != 0)

{

double VarArea = VarAreaA \* VarAreaB;

double VarPerimeter = (VarPerimeterA \* 2) + (VarPerimeterB \* 2);

labelDisplayResult.Text = "The result of the calculation for the area of a Square is " + VarArea + " and its perimeter is " + VarPerimeter;

}

}

//Input validation for rhombus - area

else if (radioButtonRhombus.Checked == true && radioButtonArea.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0)

{

double VarArea = (VarAreaA \* VarAreaB) / 2;

labelDisplayResult.Text = "The result of the calculation for the area of a Rhombus is " + VarArea;

}

}

//Input validation for rhombus - perimeter

else if (radioButtonRhombus.Checked == true && radioButtonPerimeter.Checked == true)

{

double VarPerimeterC = 0;

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarPerimeterC != 0)

{

double VarPerimeter = VarPerimeterC \* 2;

labelDisplayResult.Text = "The result of the calculation for the perimeter of a Rhombus is " + VarPerimeter;

}

}

//Input validation for rhombus - area&perimeter

else if (radioButtonRhombus.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeterC = 0;

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarPerimeterC != 0)

{

double VarArea = (VarAreaA \* VarAreaB) / 2;

double VarPerimeter = VarPerimeterC \* 2;

labelDisplayResult.Text = "The result of the calculation for the area of a Rhombus is " + VarArea + " and its perimeter is " + VarPerimeter;

}

}

//Input validation for triangle - area

else if (radioButtonTriangle.Checked == true && radioButtonArea.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0)

{

double VarArea = (VarAreaA \* VarAreaB) / 2;

labelDisplayResult.Text = "The result of the calculation for the area of a Triangle is " + VarArea;

}

}

//Input validation for triangle - perimeter

else if (radioButtonTriangle.Checked == true && radioButtonPerimeter.Checked == true)

{

double VarPerimeterA = 0;

double VarPerimeterB = 0;

double VarPerimeterC = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarPerimeterA != 0 && VarPerimeterB != 0 && VarPerimeterC != 0)

{

double VarPerimeter = VarPerimeterC + VarPerimeterA + VarPerimeterB;

labelDisplayResult.Text = "The result of the calculation for the perimeter of a Triangle is " + VarPerimeter;

}

}

//Input validation for triangle - area&perimeter

else if (radioButtonTriangle.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeterA = 0;

double VarPerimeterB = 0;

double VarPerimeterC = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarPerimeterA != 0 && VarPerimeterB != 0 && VarPerimeterC != 0)

{

double VarArea = (VarAreaA \* VarAreaB) / 2;

double VarPerimeter = VarPerimeterC + VarPerimeterA + VarPerimeterB;

labelDisplayResult.Text = "The result of the calculation for the area of a Triangle is " + VarArea + " and its perimeter is " + VarPerimeter;

}

}

//Input validation for trapezoid - area

else if (radioButtonTrapezoid.Checked == true && radioButtonArea.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

double VarAreaC = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarAreaC != 0)

{

double VarArea = ((VarAreaA + VarAreaB) \* VarAreaC) / 2;

labelDisplayResult.Text = "The result of the calculation for the area of a Trapezoid is " + VarArea;

}

}

//Input validation for trapezoid - perimeter

else if (radioButtonTrapezoid.Checked == true && radioButtonPerimeter.Checked == true)

{

double VarPerimeterA = 0;

double VarPerimeterB = 0;

double VarPerimeterE = 0;

double VarPerimeterD = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterE = Convert.ToDouble(textBoxDisplayInputE.Text);

}

catch

{

MessageBox.Show("The [input E] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input E", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterD = Convert.ToDouble(textBoxDisplayInputD.Text);

}

catch

{

MessageBox.Show("The [input D] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input D", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarPerimeterA != 0 && VarPerimeterB != 0 && VarPerimeterD != 0 && VarPerimeterE != 0)

{

double VarPerimeter = VarPerimeterA + VarPerimeterB + VarPerimeterE + VarPerimeterD;

labelDisplayResult.Text = "The result of the calculation for the perimeter of a Trapezoid is " + VarPerimeter;

}

}

//Input validation for trapezoid - area&perimeter

else if (radioButtonTrapezoid.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

double VarAreaC = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeterA = 0;

double VarPerimeterB = 0;

double VarPerimeterE = 0;

double VarPerimeterD = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterE = Convert.ToDouble(textBoxDisplayInputE.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterD = Convert.ToDouble(textBoxDisplayInputD.Text);

}

catch

{

MessageBox.Show("The [input D] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input D", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarAreaC != 0 && VarPerimeterA != 0 && VarPerimeterB != 0 && VarPerimeterD != 0 && VarPerimeterE != 0)

{

double VarArea = ((VarAreaA + VarAreaB) \* VarAreaC) / 2;

double VarPerimeter = VarPerimeterA + VarPerimeterB + VarPerimeterE + VarPerimeterD;

labelDisplayResult.Text = "The result of the calculation for the area of a Trapezoid is " + VarArea + " and its perimeter is " + VarPerimeter;

}

}

//Input validation for circle - area

else if (radioButtonCircle.Checked == true && radioButtonArea.Checked == true)

{

double VarAreaA = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0)

{

double VarPI = 3.14;

double VarArea = VarPI \* (VarAreaA \* VarAreaA);

labelDisplayResult.Text = "The result of the calculation for the area of a Circle is " + VarArea;

}

}

//Input validation for circle - perimeter

else if (radioButtonCircle.Checked == true && radioButtonPerimeter.Checked == true)

{

double VarPerimeterA = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarPerimeterA != 0)

{

double VarPI = 3.14;

double VarPerimeter = VarPerimeterA \* 2 \* VarPI;

labelDisplayResult.Text = "The result of the calculation for the perimeter of a Circle is " + VarPerimeter;

}

}

//Input validation for circle - area

else if (radioButtonCircle.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

double VarAreaA = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeterA = 0;

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarPerimeterA != 0)

{

double VarPI = 3.14;

double VarArea = VarPI \* (VarAreaA \* VarAreaA);

double VarPerimeter = VarPerimeterA \* 2 \* VarPI;

labelDisplayResult.Text = "The result of the calculation for the area of a Circle is " + VarArea + " and its perimeter is " + VarPerimeter;

}

}

//Input validation for parallelogram - area

else if (radioButtonParallelogram.Checked == true && radioButtonArea.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0)

{

double VarArea = (VarAreaA \* VarAreaB);

labelDisplayResult.Text = "The result of the calculation for the area of a Parallelogram is " + VarArea;

}

}

//Input validation for parallelogram - perimeter

else if (radioButtonParallelogram.Checked == true && radioButtonPerimeter.Checked == true)

{

double VarPerimeterC = 0;

double VarPerimeterB = 0;

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input D] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input D", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarPerimeterB != 0 && VarPerimeterC != 0)

{

double VarPerimeter = 2 \* (VarPerimeterB + 2) \* VarPerimeterC;

labelDisplayResult.Text = "The result of the calculation for the perimeter of a Parallelogram is " + VarPerimeter;

}

}

//Input validation for parallelogram - area&perimeter

else if (radioButtonParallelogram.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeterC = 0;

double VarPerimeterB = 0;

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input D] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input D", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarPerimeterB != 0 && VarPerimeterC != 0)

{

double VarArea = (VarAreaA \* VarAreaB);

double VarPerimeter = 2 \* (VarPerimeterB + 2) \* VarPerimeterC;

labelDisplayResult.Text = "The result of the calculation for the area of a Parallelogram is " + VarArea + " and its perimeter is " + VarPerimeter;

}

}

//Input validation for kite - area

else if (radioButtonKite.Checked == true && radioButtonArea.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0)

{

double VarArea = (VarAreaA \* VarAreaB) / 2;

labelDisplayResult.Text = "The result of the calculation for the area of a Kite is " + VarArea;

}

}

//Input validation for kite - perimeter

else if (radioButtonKite.Checked == true && radioButtonPerimeter.Checked == true)

{

double VarPerimeterC = 0;

double VarPerimeterD = 0;

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterD = Convert.ToDouble(textBoxDisplayInputD.Text);

}

catch

{

MessageBox.Show("The [input D] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input D", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarPerimeterC != 0 && VarPerimeterD != 0)

{

double VarPerimeter = 2 \* (VarPerimeterC + 2) \* VarPerimeterD;

labelDisplayResult.Text = "The result of the calculation for the perimeter of a Kite is " + VarPerimeter;

}

}

//Input validation for kite - area&perimeter

else if (radioButtonKite.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeterC = 0;

double VarPerimeterD = 0;

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterD = Convert.ToDouble(textBoxDisplayInputD.Text);

}

catch

{

MessageBox.Show("The [input D] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input D", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarPerimeterC != 0 && VarPerimeterD != 0)

{

double VarArea = (VarAreaA \* VarAreaB) / 2;

double VarPerimeter = 2 \* (VarPerimeterC + 2) \* VarPerimeterD;

labelDisplayResult.Text = "The result of the calculation for the area of a Kite is " + VarArea + " and its perimeter is " + VarPerimeter;

}

}

//Input validation for polygon - area

else if (radioButtonPolygon.Checked == true && radioButtonArea.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

double VarAreaC = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarAreaC != 0)

{

double VarArea = ((VarAreaA \* VarAreaC) \* VarAreaB) / 2;

labelDisplayResult.Text = "The result of the calculation for the area of a Polygon is " + VarArea;

}

}

//Input validation for polygon - perimeter

else if (radioButtonPolygon.Checked == true && radioButtonPerimeter.Checked == true)

{

double VarPerimeterC = 0;

double VarPerimeterA = 0;

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeter = VarPerimeterA \* VarPerimeterC;

labelDisplayResult.Text = "The result of the calculation for the perimeter of a Polygon is " + VarPerimeter;

}

//Input validation for polygon - area&perimeter

else if (radioButtonPolygon.Checked == true && radioButtonAreaPerimeter.Checked == true)

{

double VarAreaA = 0;

double VarAreaB = 0;

double VarAreaC = 0;

try

{

VarAreaA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaB = Convert.ToDouble(textBoxDisplayInputB.Text);

}

catch

{

MessageBox.Show("The [input B] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input B", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarAreaC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

double VarPerimeterC = 0;

double VarPerimeterA = 0;

try

{

VarPerimeterC = Convert.ToDouble(textBoxDisplayInputC.Text);

}

catch

{

MessageBox.Show("The [input c] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input c", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

try

{

VarPerimeterA = Convert.ToDouble(textBoxDisplayInputA.Text);

}

catch

{

MessageBox.Show("The [input A] field may not be empty! It must also contain a valid data type. No alphanumeric characters are allowed!", "Validation Error - Input A", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (VarAreaA != 0 && VarAreaB != 0 && VarAreaC != 0 && VarPerimeterC != 0 && VarPerimeterA != 0)

{

double VarArea = ((VarAreaA \* VarAreaC) \* VarAreaB) / 2;

double VarPerimeter = VarPerimeterA \* VarPerimeterC;

labelDisplayResult.Text = "The result of the calculation for the area of a Polygon is " + VarArea + " and its perimeter is " + VarPerimeter;

}

}

}

//Exit button

private void buttonExit\_Click(object sender, EventArgs e)

{

this.Close();

}

//Radio rectangle after calculate is clicked

private void radioButtonRectangle\_CheckedChanged(object sender, EventArgs e)

{

if(buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio square after calculate is clicked

private void radioButtonSquare\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio rhombus after calculate is clicked

private void radioButtonRhombus\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio triangle after calculate is clicked

private void radioButtonTriangle\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio trapezoid after calculate is clicked

private void radioButtonTrapezoid\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio circle after calculate is clicked

private void radioButtonCircle\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio parallelogram after calculate is clicked

private void radioButtonParallelogram\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio kite after calculate is clicked

private void radioButtonKite\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio polygon after calculate is clicked

private void radioButtonPolygon\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio area after calculate is clicked

private void radioButtonArea\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio perimeter after calculate is clicked

private void radioButtonPerimeter\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

//Radio area&perimeter after calculate is clicked

private void radioButtonAreaPerimeter\_CheckedChanged(object sender, EventArgs e)

{

if (buttonCalculate.Visible == false)

{

MessageBox.Show("The calculation is final and the answer is concluded. Same data may not be used for other formula or calculation. Please clear the data to continue...", "Data Entry Procedure Violation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

Application.Restart();

Environment.Exit(0);

}

}

}

}