Задание №1

Условие:

Доработать программу так, чтоб в ней выдавались ошибки при различных исключительных ситуациях, а также проверить результат работы.

Исходный код программы:

Switch.cs:

using System;

namespace SwitchDevices

{

public enum CoolantSystemStatus { OK, Check, Fail }

public enum SuccessFailureResult { Success, Fail }

public class Switch

{

private Random rand = new Random();

public SuccessFailureResult DisconnectPowerGenerator()

{

SuccessFailureResult r = SuccessFailureResult.Fail;

if (rand.Next(1, 10) > 2) r = SuccessFailureResult.Success;

if (rand.Next(1, 20) > 18) throw new PowerGeneratorCommsException("Network failure accessing Power Generator monitoring system");

return r;

}

public CoolantSystemStatus VerifyPrimaryCoolantSystem()

{

CoolantSystemStatus c = CoolantSystemStatus.Fail;

int r = rand.Next(1, 10);

if (r > 5)

{

c = CoolantSystemStatus.OK;

}

else if (r > 2)

{

c = CoolantSystemStatus.Check;

}

if (rand.Next(1, 20) > 18) throw new CoolantTemperatureReadException("Failed to read primary coolant system temperature");

if (rand.Next(1, 20) > 18) throw new CoolantPressureReadException("Failed to read primary coolant system pressure");

return c;

}

public CoolantSystemStatus VerifyBackupCoolantSystem()

{

CoolantSystemStatus c = CoolantSystemStatus.Fail;

int r = rand.Next(1, 10);

if (r > 5)

{

c = CoolantSystemStatus.OK;

}

else if (r > 2)

{

c = CoolantSystemStatus.Check;

}

if (rand.Next(1, 20) > 19) throw new CoolantTemperatureReadException("Failed to read backup coolant system temperature");

if (rand.Next(1, 20) > 19) throw new CoolantPressureReadException("Failed to read backup coolant system pressure");

return c;

}

public double GetCoreTemperature()

{

if (rand.Next(1, 20) > 18) throw new CoreTemperatureReadException("Failed to read core reactor system temperature");

return rand.NextDouble() \* 1000;

}

public SuccessFailureResult InsertRodCluster()

{

SuccessFailureResult r = SuccessFailureResult.Fail;

if (rand.Next(1, 100) > 5) r = SuccessFailureResult.Success;

if (rand.Next(1, 10) > 8) throw new RodClusterReleaseException("Sensor failure, cannot verify rod release");

return r;

}

public double GetRadiationLevel()

{

if (rand.Next(1, 20) > 18) throw new CoreRadiationLevelReadException("Failed to read core reactor system radiation levels");

return rand.NextDouble() \* 500;

}

public void SignalShutdownComplete()

{

if (rand.Next(1, 20) > 18) throw new SignallingException("Network failure connecting to broadcast systems");

}

}

public class PowerGeneratorCommsException : Exception

{

public PowerGeneratorCommsException(string message) : base(message) { }

}

public class CoolantSystemException : Exception

{

public CoolantSystemException(string message) : base(message) { }

}

public class CoolantTemperatureReadException : CoolantSystemException

{

public CoolantTemperatureReadException(string message) : base(message) { }

}

public class CoolantPressureReadException : CoolantSystemException

{

public CoolantPressureReadException(string message) : base(message) { }

}

public class CoreTemperatureReadException : Exception

{

public CoreTemperatureReadException(string message) : base(message) { }

}

public class CoreRadiationLevelReadException : Exception

{

public CoreRadiationLevelReadException(string message) : base(message) { }

}

public class RodClusterReleaseException : Exception

{

public RodClusterReleaseException(string message) : base(message) { }

}

public class SignallingException : Exception

{

public SignallingException(string message) : base(message) { }

}

}

MainWindow.xaml.cs:

using System;

using System.Windows;

namespace TestClient

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Button1\_Click(object sender, RoutedEventArgs e)

{

this.textBlock1.Text = "Initiating test sequence: " + DateTime.Now.ToLongTimeString();

SwitchDevices.Switch sd = new SwitchDevices.Switch();

try

{

this.textBlock1.Text += (sd.DisconnectPowerGenerator() == SwitchDevices.SuccessFailureResult.Fail) ? "\nStep 1: Failed to disconnect power generation system" : "\nStep 1: Successfully disconnected power generation system";

}

catch (SwitchDevices.PowerGeneratorCommsException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 1: " + ex.Message;

}

try

{

switch (sd.VerifyPrimaryCoolantSystem())

{

case SwitchDevices.CoolantSystemStatus.OK:

this.textBlock1.Text += "\nStep 2: Primary coolant system OK";

break;

case SwitchDevices.CoolantSystemStatus.Check:

this.textBlock1.Text += "\nStep 2: Primary coolant system requires manual check";

break;

case SwitchDevices.CoolantSystemStatus.Fail:

this.textBlock1.Text += "\nStep 2: Problem reported with primary coolant system";

break;

}

}

catch (SwitchDevices.CoolantPressureReadException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 2: " + ex.Message;

}

catch (SwitchDevices.CoolantTemperatureReadException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 2: " + ex.Message;

}

try

{

switch (sd.VerifyBackupCoolantSystem())

{

case SwitchDevices.CoolantSystemStatus.OK:

this.textBlock1.Text += "\nStep 3: Backup coolant system OK";

break;

case SwitchDevices.CoolantSystemStatus.Check:

this.textBlock1.Text += "\nStep 3: Backup coolant system requires manual check";

break;

case SwitchDevices.CoolantSystemStatus.Fail:

this.textBlock1.Text += "\nStep 3: Backup reported with primary coolant system";

break;

}

}

catch (SwitchDevices.CoolantPressureReadException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 3: " + ex.Message;

}

catch (SwitchDevices.CoolantTemperatureReadException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 3: " + ex.Message;

}

try

{

this.textBlock1.Text += "\nStep 4: Core temperature before shutdown: " + sd.GetCoreTemperature();

}

catch (SwitchDevices.CoreTemperatureReadException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 4: " + ex.Message;

}

try

{

this.textBlock1.Text += (sd.InsertRodCluster() == SwitchDevices.SuccessFailureResult.Success) ? "\nStep 5: Control rods successfully inserted" : "\nStep 5: Control rod insertion failed";

}

catch (SwitchDevices.RodClusterReleaseException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 5: " + ex.Message;

}

try

{

this.textBlock1.Text += "\nStep 6: Core temperature after shutdown: " + sd.GetCoreTemperature();

}

catch (SwitchDevices.CoreTemperatureReadException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 6: " + ex.Message;

}

try

{

this.textBlock1.Text += "\nStep 7: Core radiation level after shutdown: " + sd.GetRadiationLevel();

}

catch (SwitchDevices.CoreRadiationLevelReadException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 7: " + ex.Message;

}

try

{

sd.SignalShutdownComplete();

this.textBlock1.Text += "\nStep 8: Broadcasting shutdown complete message";

}

catch (SwitchDevices.SignallingException ex)

{

this.textBlock1.Text += "\n\*\*\*Exception in step 8: " + ex.Message;

}

this.textBlock1.Text += "\nTest sequence complete: " + DateTime.Now.ToLongTimeString();

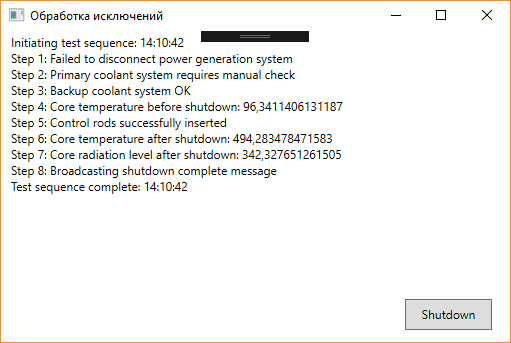
}

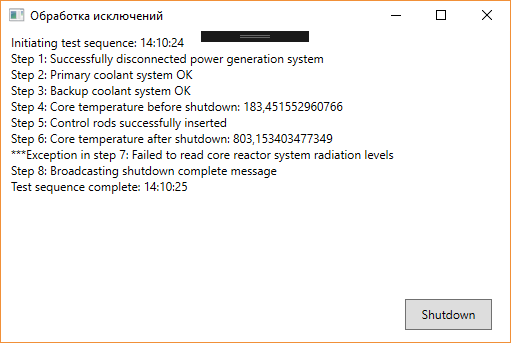
}

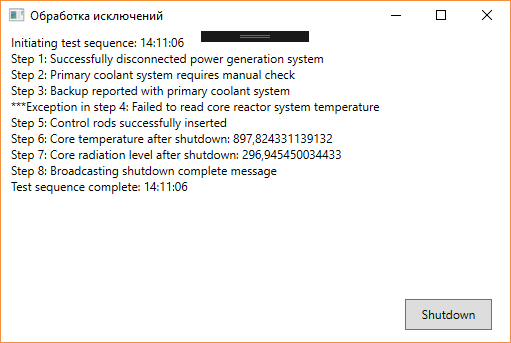
}

Скриншоты программы:

а) Пример работы без ошибок



б) Пример обработки ошибки на шаге 7

в) Пример обработки ошибки на шаге 4

Задание №2

Задание:

Доработать программу для перемножения двух матриц так, чтоб в ней выдавались ошибки при некоторых исключительных ситуациях.

Исходный код программы:

TextFileOperations.cs:

using System;

using System.Windows;

namespace MatrixMultiplication

{

static class Matrix

{

public static double[,] MatrixMultiply(double[,] matrix1, double[,] matrix2)

{

try

{

if (matrix1.GetLength(0) != matrix2.GetLength(1))

throw new ArgumentException();

}

catch (ArgumentException)

{

MessageBox.Show("Число столбцов матрицы №1 и число строк матрицы №2 должно совпадать!");

}

int m1columns\_m2rows = matrix1.GetLength(0);

int m1rows = matrix1.GetLength(1);

int m2columns = matrix2.GetLength(0);

double[,] result = new double[m2columns, m1rows];

try

{

for (int row = 0; row < m1rows; row++)

for (int column = 0; column < m2columns; column++)

{

double accumulator = 0;

for (int cell = 0; cell < m1columns\_m2rows; cell++)

{

if (matrix1[column, cell] < 0)

throw new ArgumentException(string.Format(" Matrix1 содержит некорректное значение в ячейке[{0}, {1}].", column + 1, cell + 1));

if (matrix2[column, cell] < 0)

throw new ArgumentException(string.Format(" Matrix2 содержит некорректное значение в ячейке[{0}, {1}].", column + 1, cell + 1));

accumulator += matrix1[cell, row] \* matrix2[column, cell];

}

result[column, row] = accumulator;

}

}

catch (ArgumentException ex){

MessageBox.Show(ex.Message);

}

return result;

}

}

}

MainWindow.xaml.cs:

using System;

using System.Windows;

using System.Windows.Controls;

namespace MatrixMultiplication

{

public partial class MainWindow : Window

{

double[,] matrix1;

double[,] matrix2;

double[,] result;

public MainWindow()

{

InitializeComponent();

}

private void MatrixDimensions\_Changed(object sender, SelectionChangedEventArgs e)

{

int m1rows = 1;

int m1columns\_m2rows = 1;

int m2columns = 1;

if (matrix1width != null) m1columns\_m2rows = matrix1width.SelectedIndex + 1;

if (matrix1height != null) m1rows = matrix1height.SelectedIndex + 1;

if (matrix2width != null) m2columns = matrix2width.SelectedIndex + 1;

matrix1 = new double[m1columns\_m2rows, m1rows];

matrix2 = new double[m2columns, m1columns\_m2rows];

result = new double[m2columns, m1rows];

InitializeGrid(grid1, matrix1);

InitializeGrid(grid2, matrix2);

InitializeGrid(grid3, result);

}

private void InitializeGrid(Grid grid, double[,] matrix)

{

if (grid != null)

{

grid.Children.Clear();

grid.ColumnDefinitions.Clear();

grid.RowDefinitions.Clear();

int columns = matrix.GetLength(0);

int rows = matrix.GetLength(1);

for (int x = 0; x < columns; x++)

grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = new GridLength(1, GridUnitType.Star), });

for (int y = 0; y < rows; y++)

grid.RowDefinitions.Add(new RowDefinition() { Height = new GridLength(1, GridUnitType.Star), });

for (int x = 0; x < columns; x++)

for (int y = 0; y < rows; y++)

{

double cell = (double)matrix[x, y];

TextBox t = new TextBox();

t.Text = cell.ToString();

t.VerticalAlignment = System.Windows.VerticalAlignment.Center;

t.HorizontalAlignment = System.Windows.HorizontalAlignment.Center;

t.SetValue(Grid.RowProperty, y);

t.SetValue(Grid.ColumnProperty, x);

grid.Children.Add(t);

}

}

}

private void ButtonCalculate\_Click(object sender, RoutedEventArgs e)

{

GetValuesFromGrid(grid1, matrix1);

GetValuesFromGrid(grid2, matrix2);

try

{

result = Matrix.MatrixMultiply(matrix1, matrix2);

throw new ArgumentException();

}

catch (ArgumentException ex)

{

MessageBox.Show(ex.Message);

}

InitializeGrid(grid3, result);

}

private void GetValuesFromGrid(Grid grid, double[,] matrix)

{

int columns = grid.ColumnDefinitions.Count;

int rows = grid.RowDefinitions.Count;

if (columns != matrix.GetLength(0)) throw new ArgumentException("Grid and matrix have different number of columns");

if (rows != matrix.GetLength(1)) throw new ArgumentException("Grid and matrix have different number of rows");

for (int c = 0; c < grid.Children.Count; c++)

{

TextBox t = (TextBox)grid.Children[c];

int row = Grid.GetRow(t);

int column = Grid.GetColumn(t);

matrix[column, row] = double.Parse(t.Text);

}

}

}

}

Скриншоты программы:

а) Исходный файл

б) Отображение информации в программе

в) Измененный файл