Michael Furtado

COP 2362

Professor Hamilton

December 1, 2019

Data Collector

1.

A screenshot of a cell phone

Description automatically generated

My Code : IMeasureDevice.cs

namespace Data\_Collector

{

interface IMeasuringDevice

{

// method declarations

decimal MetricValue(); // This method will return a decimal that represents the metric value of the most recent measurement that was captured.

decimal ImperialValue(); // This method will return a decimal that represents the imperial value of the most recent measurement that was captured

void StartCollecting(); // This method will start the device running.

void StopCollecting(); // This method will stop the device.

FixedSizeQueue<int> GetRawData(); // This method will retrieve a copy of all of the recent data

}

}

My Code : Device.cs

using System;

using System.Threading;

namespace Data\_Collector

{

class Device

{

private Timer timer; // declare timer object

private int data = 0; // declare variable to hold randomly generated value for GetMeasurement() to return

Random rand = new Random(); // create new instance of Random to geneate measurements

// constructor

public Device()

{

//myData.Limit = 10;

timer = new Timer(timer\_Tick, null, (int)TimeSpan.FromSeconds(1).TotalMilliseconds, (int)TimeSpan.FromSeconds(2).TotalMilliseconds);

}

// method declarations

public int GetMeasurement()

{

return data;

}

//public string History => PrintValues(myData);

//public string PrintValues(FixedSizeQueue<int> myQueue)

//{

// StringBuilder myString = new StringBuilder();

// foreach (var i in myQueue.q)

// {

// myString.AppendLine(i.ToString());

// }

// return myString.ToString();

//}

private async void timer\_Tick(object state)

{

// randomly generate new random number and update variable data

//data = rand.Next(1, 11);

//myData.Enqueue(data);

await Windows.ApplicationModel.Core.CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal,

() =>

{

data = rand.Next(1, 11);

//myData.Enqueue(data);

});

}

/\* implement in MeasureLengthDevice.cs \*/

//private void OnTimedEvent(Object source, ElapsedEventArgs e)

//{

// // example from docs.microsoft.com

// data = rand.Next(1, 11);

// myData.Enqueue(data);

//}

// method to stop timer

//public void TimerStop()

//{

// timer.Stop();

//}

//// method to start timer if the timer is stopped

//public void TimerStart()

//{

// timer.Start();

//}

}

}

My Code : MeasureLengthDevice.cs

using System;

using System.Text;

using System.Threading;

namespace Data\_Collector

{

class MeasureLengthDevice : IMeasuringDevice

{

// fields

const double CENTIMETERS\_IN\_AN\_INCH = 2.54;

private enum unitsOfMeasure { imperial, metric };

private unitsOfMeasure unitsToUse;

private unitsOfMeasure UnitsToUse

{

get { return this.unitsToUse; }

set { this.unitsToUse = value; }

}

// changing dataCaptured to Fixed size queue instead of int array - to make updating easier as well as getting raw data

// private int[] dataCaptured;

FixedSizeQueue<int> dataCaptured;

private int mostRecentMeasure = 0;

public int MostRecentMeasure

{

get { return this.mostRecentMeasure; }

}

private Device dev;

private Timer timer;

// construcrtors

public MeasureLengthDevice()

{

// virtual device to provide measurements

dev = new Device();

timer = new Timer(timer\_Tick, null, (int)TimeSpan.FromSeconds(1).TotalMilliseconds, (int)TimeSpan.FromSeconds(4).TotalMilliseconds);

unitsToUse = unitsOfMeasure.imperial;

mostRecentMeasure = dev.GetMeasurement();

dataCaptured = new FixedSizeQueue<int>();

dataCaptured.Limit = 10;

//dataCaptured.Enqueue(mostRecentMeasure);

}

// methods

public string History => PrintValues(dataCaptured);

private async void timer\_Tick(object state)

{

// timer to get new measurement from Devide via GetMeasurement and Enqueue in dataCaptured FixedSizeQueue

await Windows.ApplicationModel.Core.CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal,

() => {

mostRecentMeasure = dev.GetMeasurement();

dataCaptured.Enqueue(mostRecentMeasure);

});

}

public decimal MetricValue()

{

//throw new NotImplementedException();

if(unitsToUse == unitsOfMeasure.imperial)

{

return (decimal)(mostRecentMeasure \* CENTIMETERS\_IN\_AN\_INCH);

}

else

{

return (decimal)mostRecentMeasure;

}

}

public decimal ImperialValue()

{

//throw new NotImplementedException();

if (unitsToUse == unitsOfMeasure.metric)

{

return (decimal)mostRecentMeasure;

}

else

{

return (decimal)(mostRecentMeasure \* CENTIMETERS\_IN\_AN\_INCH);

}

}

public void StartCollecting()

{

//throw new NotImplementedException();

//dev.TimerStart();

}

public void StopCollecting()

{

//throw new NotImplementedException();

//dev.TimerStop();

}

public FixedSizeQueue<int> GetRawData()

{

throw new NotImplementedException();

}

public string PrintValues(FixedSizeQueue<int> myQueue)

{

StringBuilder myString = new StringBuilder();

foreach (var i in myQueue.q)

{

myString.Append(i.ToString() + " ");

}

return myString.ToString();

}

}

}

My Code : MainPage.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

using System.Threading;

using Windows.ApplicationModel.Core;

using Windows.UI.Core;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace Data\_Collector

{

/// <summary>

/// An empty page that can be used on its own or navigated to within a Frame.

/// </summary>

public sealed partial class MainPage : Page

{

MeasureLengthDevice dataCollector = null;

MainDataDisplay displayData = null;

public MainPage()

{

this.InitializeComponent();

dataCollector = new MeasureLengthDevice();

displayData = new MainDataDisplay

{

Measurement = dataCollector.MostRecentMeasure,

History = dataCollector.History

};

ImperialRB.IsChecked = true;

bool collectingData = true;

}

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

{

currentValueOutputTBlk.Text = dataCollector.MostRecentMeasure.ToString();

dataHistoryTBox.Text = dataCollector.History;

}

}

}

My Code : MainPage.xaml

<Page

x:Class="Data\_Collector.MainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:Data\_Collector"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid>

<RadioButton x:Name="MetricRB" Content="Metric" Margin="128,113,0,0" VerticalAlignment="Top"/>

<RadioButton x:Name="ImperialRB" Content="Imperial" Margin="128,165,0,0" VerticalAlignment="Top" IsChecked="True"/>

<Button x:Name="StartCollectBtn" Content="Start Collecting Data" Margin="128,241,0,0" VerticalAlignment="Top" Width="166"/>

<Button x:Name="StopCollectingBtn" Content="Stop Collecting Data" Margin="325,241,0,0" VerticalAlignment="Top" Width="166"/>

<TextBlock x:Name="CollectingStateTBlk" HorizontalAlignment="Left" Margin="188,307,0,0" Text="State:" TextWrapping="Wrap" VerticalAlignment="Top" Height="39" RenderTransformOrigin="0.534,0.69" Width="104" TextAlignment="Center"/>

<TextBox x:Name="CurrentCollectingStateTBox" HorizontalAlignment="Left" Margin="325,300,0,0" Text="Collecting" TextWrapping="Wrap" VerticalAlignment="Top" RenderTransformOrigin="-0.669,-0.406" Width="166" TextAlignment="Center"/>

<TextBlock x:Name="currentValueTBlk" HorizontalAlignment="Left" Margin="128,373,0,0" Text="Current Value: " TextWrapping="Wrap" VerticalAlignment="Top" Height="31" Width="171" RenderTransformOrigin="1.155,2.387" TextAlignment="Center"/>

<TextBlock x:Name="currentValueOutputTBlk" HorizontalAlignment="Left" Margin="325,373,0,0" Text="" TextWrapping="Wrap" VerticalAlignment="Top" Height="28" Width="105" TextAlignment="Center"/>

<Button x:Name="getCurrentDataBtn" Content="Get Most Current Data" Margin="128,425,0,0" VerticalAlignment="Top" Width="363" Click="getCurrentDataBtn\_Click"/>

<TextBox x:Name="dataHistoryTBox" HorizontalAlignment="Left" Margin="188,516,0,0" Text="" TextWrapping="Wrap" VerticalAlignment="Top" Width="238" Height="34" TextAlignment="Center"/>

<TextBlock x:Name="datacapturedTBlk" HorizontalAlignment="Left" Margin="188,479,0,0" Text="Data Previously Captured" TextWrapping="Wrap" VerticalAlignment="Top" Width="252" TextAlignment="Center"/>

</Grid>

</Page>

My Code : MainDataDisplay.cs

namespace Data\_Collector

{

class MainDataDisplay

{

private int measurement;

private string history;

public int Measurement

{

get { return this.measurement; }

set { this.measurement = value; }

}

public string History

{

get { return this.history; }

set { this.history = value; }

}

}

}