Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Уфимский государственный авиационный технический университет»

Факультет информатики и робототехники

Кафедра Вычислительной математики и кибернетики

**Лабораторная работа №4 Часть 2**

По дисциплине «Объектно-ориентированное программирование»

«MVC»

Выполнил: студент группы ПРО-222

Вальшин Руслан Ильдусович

УФА 2020

1. **Цель работы:**

Научиться разделять данные приложения, пользовательского интерфейса и управляющей логики на компоненты.

1. **Задачи:**

• Создать простейшее приложение с GUI, содержащее:

* два числа, одно из которых обязательно должно быть больше другого
* каждое из чисел должно отображаться и редактироваться в 3 разных компонентах, при этом редактирование числа в одном поле должно приводить к изменению отображения этого числа во всех других полях
* приложение не должно использовать MVC, все пересчёты должны выполняться напрямую в обработчиках событий

• Переделать созданное приложение с использованием идей MVC:

* хранение двух чисел должно быть организовано в виде отдельного объекта-модели
* все пересчёты и проверки должны выполняться в объекте-модели

1. **Ход работы:**

Реализация без MVC. Обработка событий 6 компонентов. Если изменение внесли в numericUpDown, это значение меньше или больше другого числа, то меняем значение во всех трёх компонентах. Если условие «Одно число обязательно больше другого» не выполняется, значение в numericUpDown меняется на первоначальное. То же самое происходит и для textbox и richTextBox, только там дополнительно проверяется существование числа.

private void numericUpDown1\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                if (numericUpDown1.Value < numericUpDown2.Value)

                {

                    textBox1.Text = numericUpDown1.Value.ToString();

                    richTextBox1.Text = numericUpDown1.Value.ToString();

                }

                else

                    numericUpDown1.Value = int.Parse(textBox1.Text);

            }

        }

        private void textBox1\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                int x = 0;

                if (int.TryParse(textBox1.Text, out x))

                {

                    if (x < numericUpDown2.Value)

                    {

                        numericUpDown1.Value = x;

                        richTextBox1.Text = textBox1.Text;

                    }

                    else

                        textBox1.Text = numericUpDown1.Value.ToString();

                }

            }

        }

        private void richTextBox1\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                int x = 0;

                if (int.TryParse(richTextBox1.Text, out x))

                {

                    if (x < numericUpDown2.Value)

                    {

                        textBox1.Text = richTextBox1.Text;

                        numericUpDown1.Value = x;

                    }

                    else

                        richTextBox1.Text = numericUpDown1.Value.ToString();

                }

            }

        }

        private void numericUpDown2\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                if (numericUpDown1.Value < numericUpDown2.Value)

                {

                    textBox2.Text = numericUpDown2.Value.ToString();

                    richTextBox2.Text = numericUpDown2.Value.ToString();

                }

                else

                    numericUpDown2.Value = int.Parse(textBox2.Text);

            }

        }

        private void textBox2\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                int x = 0;

                if (int.TryParse(textBox2.Text, out x))

                {

                    if (x > numericUpDown1.Value)

                    {

                        numericUpDown2.Value = x;

                        richTextBox2.Text = textBox2.Text;

                    }

                    else

                        textBox2.Text = numericUpDown2.Value.ToString();

                }

            }

        }

        private void richTextBox2\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                int x = 0;

                if (int.TryParse(richTextBox2.Text, out x))

                {

                    if (x > numericUpDown1.Value)

                    {

                        textBox2.Text = richTextBox2.Text;

                        numericUpDown2.Value = x;

                    }

                    else

                        richTextBox2.Text = numericUpDown2.Value.ToString();

                }

            }

        }

MVC. Создаётся класс модели. Модель хранит в себе массив значений и массив обработчиков событий. Конструктором задаются первоначальные значения. Метод setValue() устанавливает значение i-того элемента массива, если значение соответствует управляющей логике, и вызывает привязанный обработчик события. Метод getValue() возвращает значение i-того элемента.

    public class Model

    {

        private int[] value = new int[2];

        public EventHandler[] observers = new EventHandler[2];

        public Model(int value1, int value2)

        {

            value[0] = value1;

            value[1] = value2;

        }

        public void setValue(int value, int i)

        {

            if (i == 0 && value < this.value[1] || i == 1 && value > this.value[0])

                this.value[i] = value;

            observers[i].Invoke(this, null);

        }

        public int getValue(int i)

        {

            return value[i];

        }

    }

При создании формы создаётся модель, к модели подключаются обработчики событий.

Model model;

        public Form1()

        {

            InitializeComponent();

            model = new Model(15, 34);

            model.observers[0] += new EventHandler(updateFromModel1);

            model.observers[1] += new EventHandler(updateFromModel2);

        }

Обработчики событий меняют значение элементов управления на значения, которые хранит в себе модель.

public void updateFromModel1(object sender, EventArgs e)

        {

            numericUpDown3.Value = model.getValue(0);

            textBox3.Text = model.getValue(0).ToString();

            richTextBox3.Text = model.getValue(0).ToString();

        }

        public void updateFromModel2(object sender, EventArgs e)

        {

            numericUpDown4.Value = model.getValue(1);

            textBox4.Text = model.getValue(1).ToString();

            richTextBox4.Text = model.getValue(1).ToString();

        }

В обработчиках событий элементов считываются данные и передаются в модель в корректном виде.

private void numericUpDown3\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                model.setValue(Convert.ToInt32(numericUpDown3.Value), 0);

            }

        }

        private void textBox3\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                int x = 0;

                if (int.TryParse(textBox3.Text, out x))

                    model.setValue(x, 0);

            }

        }

        private void richTextBox3\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                int x = 0;

                if (int.TryParse(richTextBox3.Text, out x))

                    model.setValue(x, 0);

            }

        }

        private void numericUpDown4\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                model.setValue(Convert.ToInt32(numericUpDown4.Value), 1);

            }

        }

        private void textBox4\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                int x = 0;

                if (int.TryParse(textBox4.Text, out x))

                    model.setValue(x, 1);

            }

        }

        private void richTextBox4\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                e.SuppressKeyPress = true;

                int x = 0;

                if (int.TryParse(richTextBox4.Text, out x))

                    model.setValue(x, 1);

            }

        }

Приложение А. Ссылка на репозиторий и скриншот истории коммитов

<https://github.com/Ruslan376M/OOP_Lab4.2>

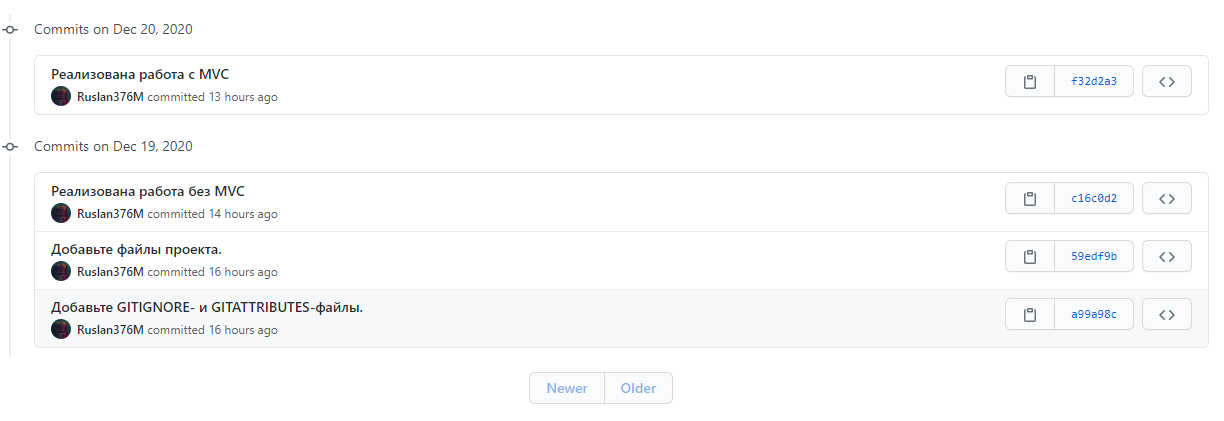


Рисунок 1 - Скриншот истории коммитов

Приложение B. Примеры работы программы.

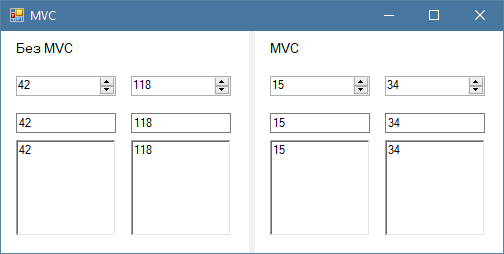


Рисунок 2 - Пример работы программы

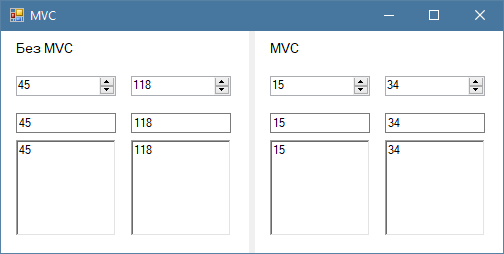


Рисунок 3 - Пример работы программы

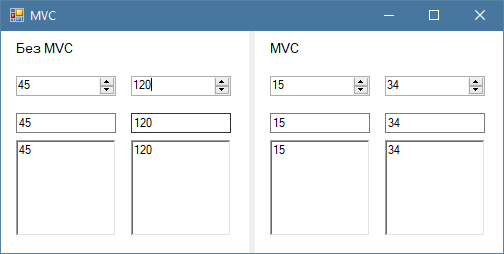


Рисунок 4 - Пример работы программы

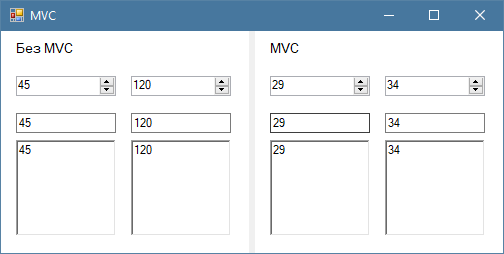


Рисунок 5 - Пример работы программы

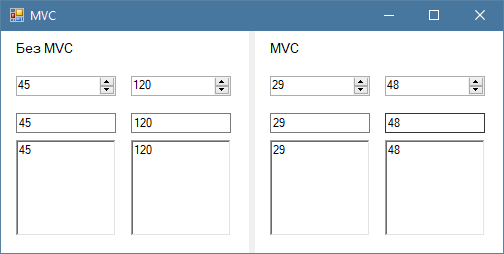


Рисунок 6 - Пример работы программы

Приложение C. Исходный код программы.

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 Лабораторная\_работа\_\_4\_ч.\_2

{

public partial class Form1 : Form

{

Model model;

public Form1()

{

InitializeComponent();

model = new Model(15, 34);

model.observers[0] += new EventHandler(updateFromModel1);

model.observers[1] += new EventHandler(updateFromModel2);

}

// Начало без MVC

private void numericUpDown1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

if (numericUpDown1.Value < numericUpDown2.Value)

{

textBox1.Text = numericUpDown1.Value.ToString();

richTextBox1.Text = numericUpDown1.Value.ToString();

}

else

numericUpDown1.Value = int.Parse(textBox1.Text);

}

}

private void textBox1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

int x = 0;

if (int.TryParse(textBox1.Text, out x))

{

if (x < numericUpDown2.Value)

{

numericUpDown1.Value = x;

richTextBox1.Text = textBox1.Text;

}

else

textBox1.Text = numericUpDown1.Value.ToString();

}

}

}

private void richTextBox1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

int x = 0;

if (int.TryParse(richTextBox1.Text, out x))

{

if (x < numericUpDown2.Value)

{

textBox1.Text = richTextBox1.Text;

numericUpDown1.Value = x;

}

else

richTextBox1.Text = numericUpDown1.Value.ToString();

}

}

}

private void numericUpDown2\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

if (numericUpDown1.Value < numericUpDown2.Value)

{

textBox2.Text = numericUpDown2.Value.ToString();

richTextBox2.Text = numericUpDown2.Value.ToString();

}

else

numericUpDown2.Value = int.Parse(textBox2.Text);

}

}

private void textBox2\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

int x = 0;

if (int.TryParse(textBox2.Text, out x))

{

if (x > numericUpDown1.Value)

{

numericUpDown2.Value = x;

richTextBox2.Text = textBox2.Text;

}

else

textBox2.Text = numericUpDown2.Value.ToString();

}

}

}

private void richTextBox2\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

int x = 0;

if (int.TryParse(richTextBox2.Text, out x))

{

if (x > numericUpDown1.Value)

{

textBox2.Text = richTextBox2.Text;

numericUpDown2.Value = x;

}

else

richTextBox2.Text = numericUpDown2.Value.ToString();

}

}

}

// MVC

public void updateFromModel1(object sender, EventArgs e)

{

numericUpDown3.Value = model.getValue(0);

textBox3.Text = model.getValue(0).ToString();

richTextBox3.Text = model.getValue(0).ToString();

}

public void updateFromModel2(object sender, EventArgs e)

{

numericUpDown4.Value = model.getValue(1);

textBox4.Text = model.getValue(1).ToString();

richTextBox4.Text = model.getValue(1).ToString();

}

private void numericUpDown3\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

model.setValue(Convert.ToInt32(numericUpDown3.Value), 0);

}

}

private void textBox3\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

int x = 0;

if (int.TryParse(textBox3.Text, out x))

model.setValue(x, 0);

}

}

private void richTextBox3\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

int x = 0;

if (int.TryParse(richTextBox3.Text, out x))

model.setValue(x, 0);

}

}

private void numericUpDown4\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

model.setValue(Convert.ToInt32(numericUpDown4.Value), 1);

}

}

private void textBox4\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

int x = 0;

if (int.TryParse(textBox4.Text, out x))

model.setValue(x, 1);

}

}

private void richTextBox4\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.SuppressKeyPress = true;

int x = 0;

if (int.TryParse(richTextBox4.Text, out x))

model.setValue(x, 1);

}

}

}

public class Model

{

private int[] value = new int[2];

public EventHandler[] observers = new EventHandler[2];

public Model(int value1, int value2)

{

value[0] = value1;

value[1] = value2;

}

public void setValue(int value, int i)

{

if (i == 0 && value < this.value[1] || i == 1 && value > this.value[0])

this.value[i] = value;

observers[i].Invoke(this, null);

}

public int getValue(int i)

{

return value[i];

}

}

}