МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Гомельский государственный технический

университет имени П.О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчёт по лабораторной работе №6

По дисциплине «Объектно-ориентированное программирование»

**«Методы и механизмы инкапсуляции и организации доступа к элементам объекта»**

Выполнил: студент

группы ИТИ-21

Говядкова П. Ю.

Принял: преподаватель

Карабчикова Е. А.

Гомель 2020

**Цель работы:** изучить основы синтаксиса объектно-ориентированного языка программирования, реализацию свойств, методов класса.

**Задание:**

1. Необходимо решить задачу, согласно варианту (Рисунок 1).
2. При создании классов руководствоваться Code Convention.
3. Весь код должен содержать элементы документирования.
4. При реализации классов использовать в обязательном порядке перечисления и (или) вложенные классы.
5. Для хранения данных использовать массивы.
6. Разработать модульные тесты для верификации созданного класса.
7. Класс должен быть размещён в библиотеке классов.
8. Модульные тесты – в отдельном проекте.
9. В отдельном проекте реализовать интерфейс WindowsForm.
10. Windows приложение должно обеспечить ввод, редактирование и просмотр данных.

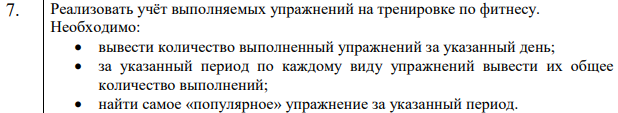


Рисунок 1 – Вариант задания

На рисунке 2 изображена структура решения.

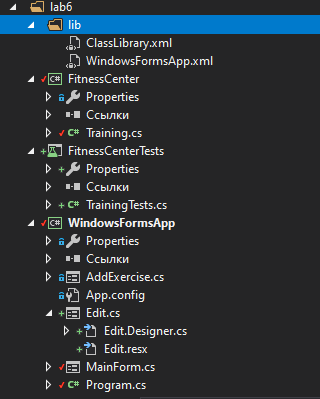


Рисунок 2 – Структура решения

На Рисунке 3 изображён вывод MainForm.

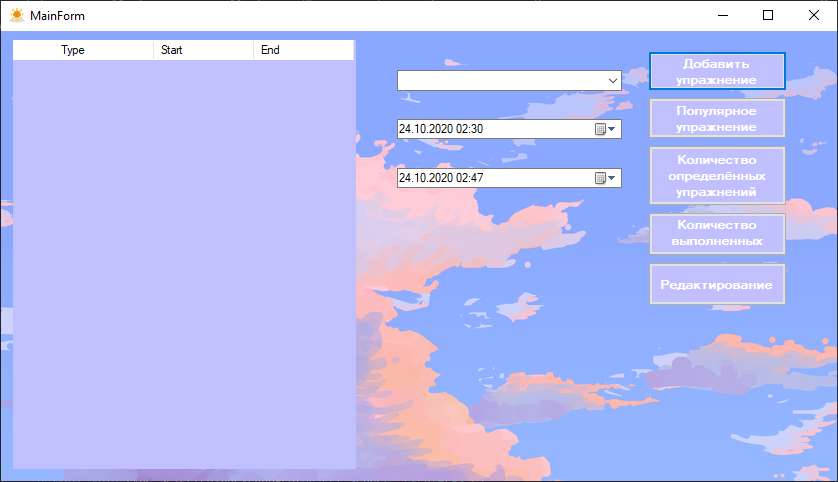


Рисунок 3 – Начало выполнения программы, вывод основной формы

На Рисунке 4 изображено выполнение пункта “Добавить упражнение”, где вводится данные об упражнении.

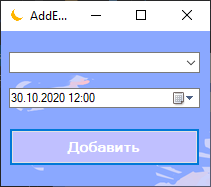


Рисунок 4 – Выполнение пункта “Добавить упражнение”

На Рисунке 5 изображено выполнение пункта “Редактировать упражнение”, где можно изменять данные.

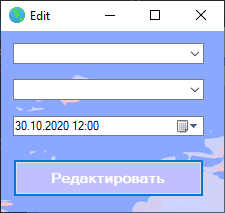


Рисунок 5 – Выполнение пункта “ Редактировать упражнение ”

На Рисунке 6 изображено выполнение пункта “Количество определённых выполненных упражнений”, где можно узнать количество определённых выполненных упражнений за указанный промежуток времени.

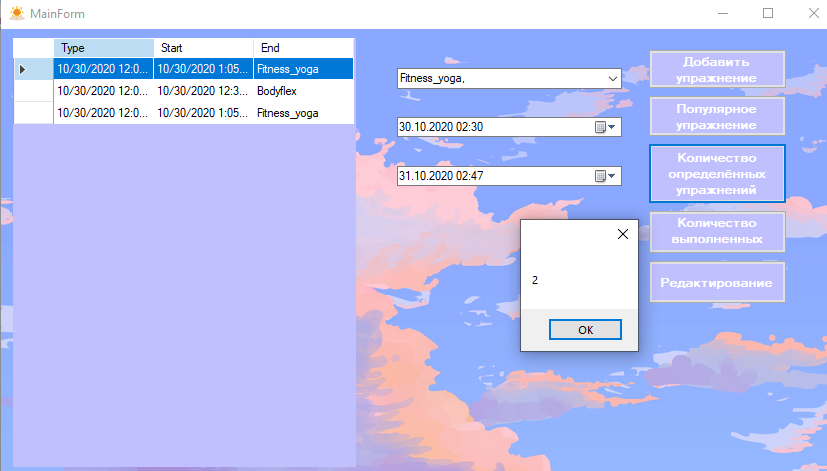


Рисунок 6 – Выполнение пункта “ Количество определённых выполненных упражнений ”

На Рисунке 7 изображено Выполнение пункта “Самое популярное упражнение”, где можно узнать упражнение, которое выполняется чаще всего.

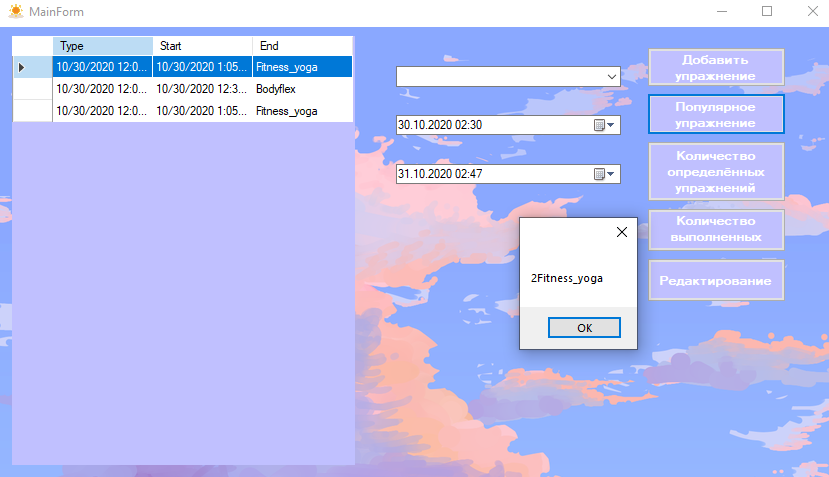


Рисунок 7 – Выполнение пункта “ Самое популярное упражнение”

На Рисунке 8 изображено выполнение пункта “Общее количество упражнений” за определённый период.

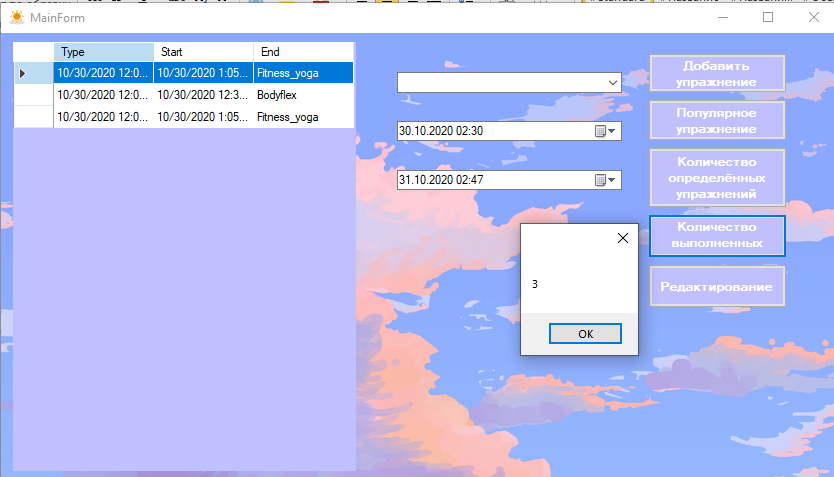


Рисунок 8 ­­­– Выполнение пункта “ Общее количество упражнений за определённый период ”

На Рисунке 9 изображено выполнение тестов

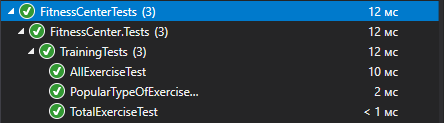


Рисунок 9 – Прохождение модульных тестов

На Рисунках 10–15 изображено выполнение программы.

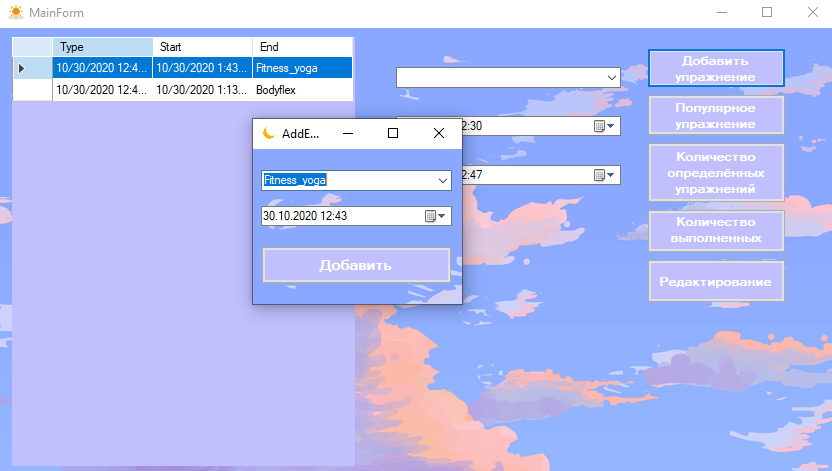


Рисунок 10 – Добавление упражнений с датой, указанной на рисунке

Добавим упражнение в другом временном диапазоне.

На Рисунке – 11 изображено добавление такого упражнения.

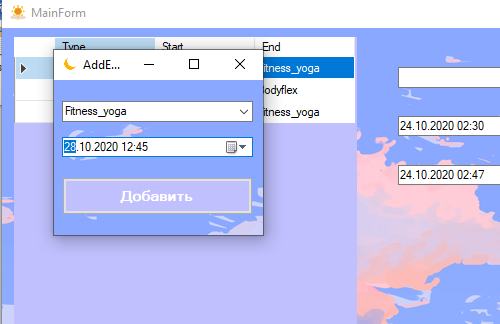


Рисунок 11 – Добавление упражнения в другом временном диапазоне

Найдём упражнения в определённом временном диапазоне и определенное упражнение. Упражнение 10/28/2020 не учитывается.

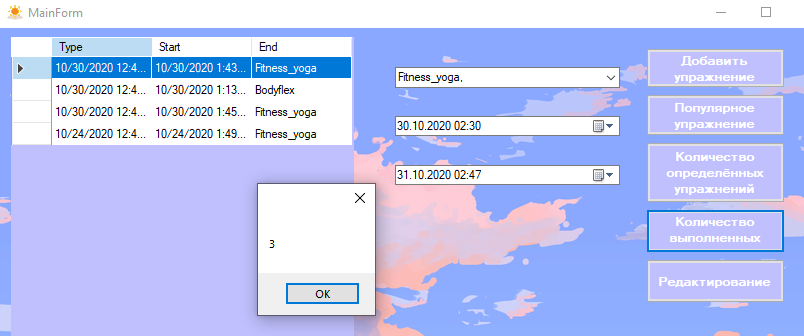


Рисунок 12 – Вывод количества выполненных упражнений

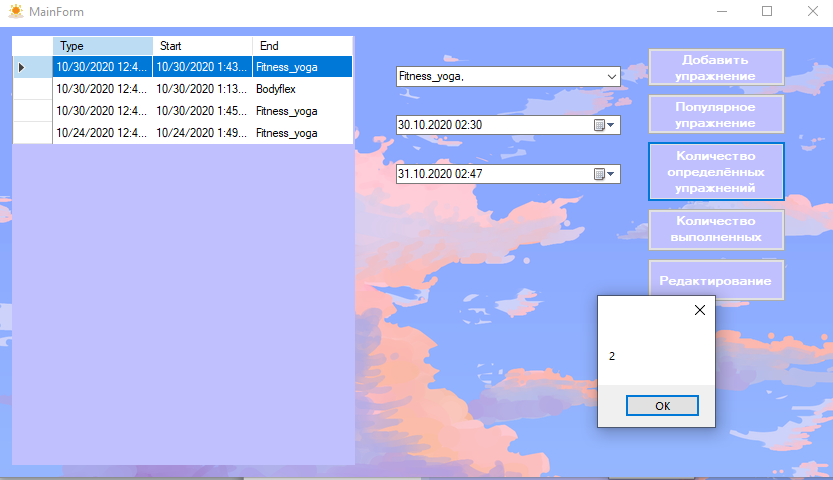


Рисунок 13 – Вывод количества определённых выполненных упражнений

На Рисунках 14–15 изображено редактирование упражнения.

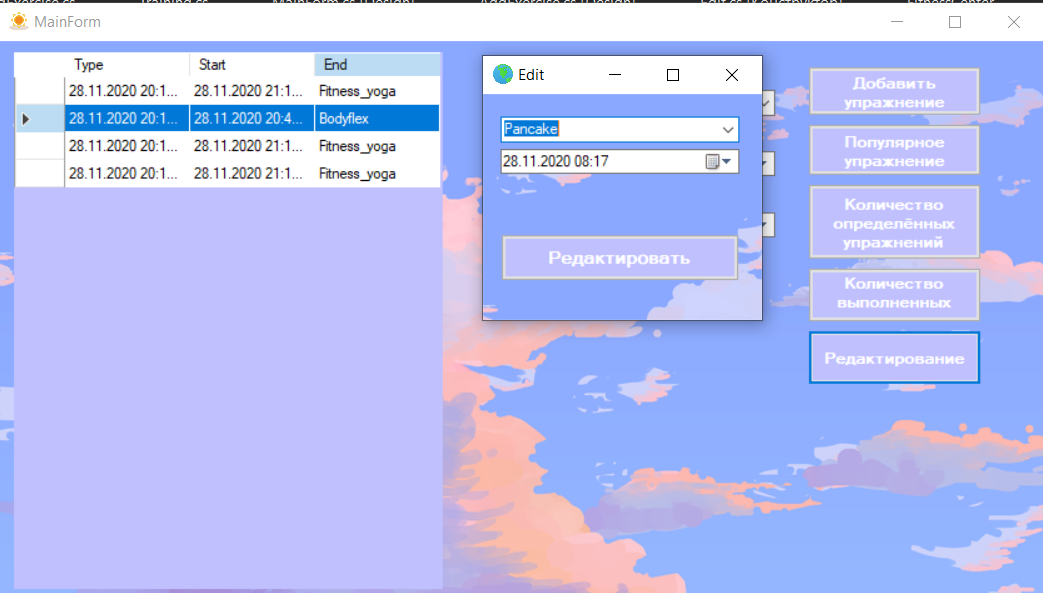


Рисунок 14 – Редактирование упражнений

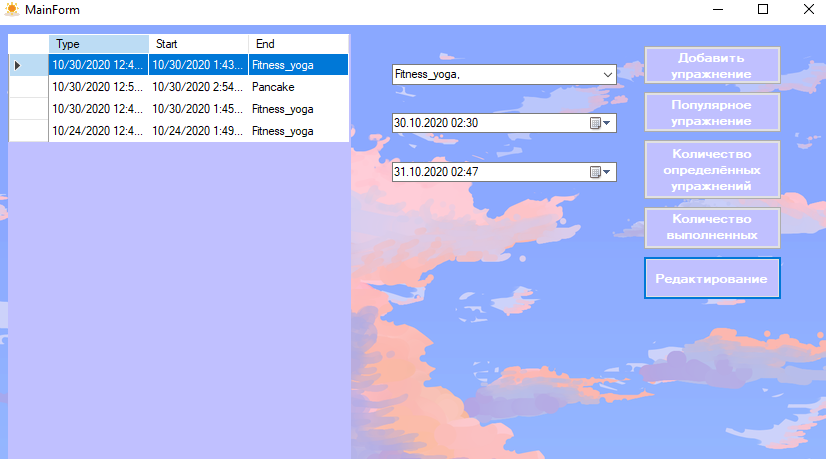


Рисунок 15 – Редактирование упражнений

**Вывод:** в ходе лабораторной работы были изучены основы синтаксиса объектно-ориентированного языка программирования. Реализован класс для обработки перечислений и вложенных в него объектов.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

Training

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace FitnessCenter

{

/// <summary>

///

/// </summary>

enum ExerciseType

{

/// <summary>

/// The fitness yoga

/// </summary>

Fitness\_yoga,

/// <summary>

/// The stretching

/// </summary>

Stretching,

/// <summary>

/// The bodyflex

/// </summary>

Bodyflex,

/// <summary>

/// The fitball

/// </summary>

Fitball,

/// <summary>

/// The nordic walking

/// </summary>

Nordic\_walking,

/// <summary>

/// The pilates

/// </summary>

Pilates,

/// <summary>

/// The callanetic

/// </summary>

Callanetic,

/// <summary>

/// The anaerobic load

/// </summary>

Anaerobic\_load,

/// <summary>

/// The butterfly

/// </summary>

Butterfly,

/// <summary>

/// The burpee

/// </summary>

Burpee,

/// <summary>

/// The pancake

/// </summary>

Pancake,

/// <summary>

/// The gainer

/// </summary>

Gainer

}

/// <summary>

/// Training

/// </summary>

public class Training

{

/// <summary>

///

/// </summary>

private class Exercise

{

/// <summary>

/// The data

/// </summary>

private DateTime \_data;

/// <summary>

/// The end data

/// </summary>

private DateTime \_end\_data;

/// <summary>

/// The type of exercise

/// </summary>

private string \_type\_of\_exercise;

/// <summary>

/// The exercise time

/// </summary>

private int[] \_exercise\_time = { 60, 120, 30, 50, 30, 60, 45, 120, 30, 60, 120, 30 };

/// <summary>

/// The number of exercise

/// </summary>

private int \_number\_of\_exercise;

/// <summary>

/// Initializes a new instance of the <see cref="Exercise"/> class.

/// </summary>

/// <param name="data">The data.</param>

/// <param name="number\_of\_exercise">The number of exercise.</param>

public Exercise(DateTime data, int number\_of\_exercise)

{

this.\_number\_of\_exercise = number\_of\_exercise;

this.\_data = data;

this.\_end\_data = data.AddMinutes(Convert.ToDouble(\_exercise\_time[\_number\_of\_exercise]));

this.\_type\_of\_exercise = Enum.GetName(typeof(ExerciseType), number\_of\_exercise);

}

/// <summary>

/// Gets or sets the data.

/// </summary>

/// <value>

/// The data.

/// </value>

internal DateTime Data

{

get

{

return \_data;

}

set

{

\_data = value;

}

}

/// <summary>

/// Gets or sets the end data.

/// </summary>

/// <value>

/// The end data.

/// </value>

internal DateTime EndData

{

get

{

return \_end\_data;

}

set

{

\_end\_data = value;

}

}

/// <summary>

/// Gets or sets the type of exercise.

/// </summary>

/// <value>

/// The type of exercise.

/// </value>

internal string TypeOfExercise

{

get

{

return \_type\_of\_exercise;

}

set

{

\_type\_of\_exercise = value;

}

}

/// <summary>

/// Gets or sets the number of exercise.

/// </summary>

/// <value>

/// The number of exercise.

/// </value>

internal int NumberOfExercise

{

get

{

return \_number\_of\_exercise;

}

set

{

\_number\_of\_exercise = value;

}

}

}

/// <summary>

/// The exercise time

/// </summary>

int[] \_exercise\_time = { 60, 120, 30, 50, 30, 60, 45, 120, 30, 60, 120, 30 };

/// <summary>

/// The exercise array

/// </summary>

private Exercise[] \_exercise\_array = new Exercise[] { };

/// <summary>

/// The index

/// </summary>

public int index = 0;

/// <summary>

/// Gets the get lenght.

/// </summary>

/// <value>

/// The get lenght.

/// </value>

public int GetLenght

{

get

{

return \_exercise\_array.Length;

}

}

/// <summary>

/// Gets or sets the get data.

/// </summary>

/// <value>

/// The get data.

/// </value>

public DateTime GetData

{

get

{

return \_exercise\_array[index].Data;

}

set

{

\_exercise\_array[index].Data = value;

}

}

/// <summary>

/// Gets or sets the get end data.

/// </summary>

/// <value>

/// The get end data.

/// </value>

public DateTime GetEndData

{

get

{

return \_exercise\_array[index].EndData;

}

set

{

\_exercise\_array[index].EndData = value;

}

}

/// <summary>

/// Gets or sets the get type of exercise.

/// </summary>

/// <value>

/// The get type of exercise.

/// </value>

public string GetTypeOfExercise

{

get

{

return \_exercise\_array[index].TypeOfExercise;

}

set

{

\_exercise\_array[index].TypeOfExercise = value;

}

}

/// <summary>

/// Alls the exercise.

/// </summary>

/// <param name="low\_date">The low date.</param>

/// <param name="high\_date">The high date.</param>

/// <returns></returns>

public string AllExercise(DateTime low\_date, DateTime high\_date)

{

int count\_of\_exercise = 0;

for (int i = 0; i < \_exercise\_array.Length; i++)

{

if (\_exercise\_array[i].Data >= low\_date && \_exercise\_array[i].EndData <= high\_date)

{

count\_of\_exercise++;

}

}

File.WriteAllText("H:\\ AllExercise.txt", Convert.ToString(count\_of\_exercise));

return Convert.ToString(count\_of\_exercise);

}

/// <summary>

/// Populars the type of exercise.

/// </summary>

/// <param name="low\_date">The low date.</param>

/// <param name="high\_date">The high date.</param>

/// <returns></returns>

public string PopularTypeOfExercise(DateTime low\_date, DateTime high\_date)

{

int[] count\_of\_exercise = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

List<string> type = new List<string>() { };

for (int i = 0; i < \_exercise\_array.Length; i++)

{

if (\_exercise\_array[i].Data >= low\_date && \_exercise\_array[i].EndData <= high\_date)

{

count\_of\_exercise[\_exercise\_array[i].NumberOfExercise]++;

type.Add(\_exercise\_array[i].TypeOfExercise);

}

}

//SelectMany(x => x.Split(...)) в качестве х последовательность,

//которую надо проецировать, а в качестве x.Split(...) - функцию преобразования, т.е.

// разделяем по пространству: с помощью String.Split и удаляем пустые строки, затем группируем по каждоиу слову

// и упорядочиваем по убыванию "употребления?" слова.

// в конце, получем первый элемент (слово) по кол-ву употребления.

string exercise\_type = type.SelectMany(x => x.Split(new[] { " " },

StringSplitOptions.RemoveEmptyEntries)).

GroupBy(x => x).

OrderByDescending(x => x.Count()).

Select(x => x.Key).

First();

File.WriteAllText("H:\\PopularTypeOfExercise.txt", exercise\_type);

return Convert.ToString(count\_of\_exercise.Max()) + exercise\_type;

}

/// <summary>

/// Totals the exercise.

/// </summary>

/// <param name="low\_date">The low date.</param>

/// <param name="high\_date">The high date.</param>

/// <param name="number\_of\_exercise">The number of exercise.</param>

/// <returns></returns>

public string TotalExercise(DateTime low\_date, DateTime high\_date, int number\_of\_exercise)

{

int count = 0;

for (int i = 0; i < \_exercise\_array.Length; i++)

{

if (\_exercise\_array[i].Data >= low\_date && \_exercise\_array[i].EndData <= high\_date)

{

if (\_exercise\_array[i].NumberOfExercise == number\_of\_exercise)

{

count++;

}

}

}

File.WriteAllText("H:\\TotalExercise.txt", Convert.ToString(count));

return Convert.ToString(count);

}

/// <summary>

/// Adds the exercise.

/// </summary>

/// <param name="new\_data">The new data.</param>

/// <param name="number\_of\_exercise">The number of exercise.</param>

public void AddExercise(DateTime new\_data, int number\_of\_exercise)

{

index = \_exercise\_array.Length + 1;

Array.Resize(ref \_exercise\_array, index--);

\_exercise\_array[index] = new Exercise(new\_data, number\_of\_exercise);

}

/// <summary>

/// Edits the exercise.

/// </summary>

/// <param name="new\_data">The new data.</param>

/// <param name="number\_of\_exercise">The number of exercise.</param>

/// <param name="i">The i.</param>

public void EditExercise(DateTime new\_data, int number\_of\_exercise, int i)

{

\_exercise\_array[i].Data = new\_data;

\_exercise\_array[i].EndData = new\_data.AddMinutes(Convert.ToDouble(\_exercise\_time[number\_of\_exercise]));

\_exercise\_array[i].TypeOfExercise = Enum.GetName(typeof(ExerciseType), number\_of\_exercise);

\_exercise\_array[i].NumberOfExercise = number\_of\_exercise;

}

}

}

MainForm

using FitnessCenter;

using System;

using System.Windows.Forms;

namespace WindowsFormsApp

{

/// <summary>

/// MainForm

/// </summary>

/// <seealso cref="System.Windows.Forms.Form" />

public partial class MainForm : Form

{

/// <summary>

/// The training

/// </summary>

public Training training;

/// <summary>

/// Initializes a new instance of the <see cref="MainForm"/> class.

/// </summary>

public MainForm()

{

this.training = new Training();

InitializeComponent();

}

/// <summary>

/// Handles the Load event of the MainForm control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void MainForm\_Load(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the 1 event of the dataGridView1\_CellContentClick control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="DataGridViewCellEventArgs"/> instance containing the event data.</param>

private void dataGridView1\_CellContentClick\_1(object sender, DataGridViewCellEventArgs e)

{

}

/// <summary>

/// Handles the Click event of the button1 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void button1\_Click(object sender, EventArgs e)

{

AddExercise addExercise = new AddExercise(this);

addExercise.Show();

}

/// <summary>

/// Handles the Click event of the button2 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void button2\_Click(object sender, EventArgs e)

{

try

{

MessageBox.Show(training.PopularTypeOfExercise(date1.Value, date2.Value));

}

catch

{

MessageBox.Show("Последовательность не содержит элементов");

}

}

/// <summary>

/// Handles the Click event of the button3 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void button3\_Click(object sender, EventArgs e)

{

try

{

if(training.TotalExercise(date1.Value, date2.Value, comboBox1.SelectedIndex) == "0")

{

MessageBox.Show("Надо выбрать упражнение из списка");

}

else MessageBox.Show(training.TotalExercise(date1.Value, date2.Value, comboBox1.SelectedIndex));

}

catch (Exception ex)

{

MessageBox.Show(Convert.ToString(ex));

}

}

/// <summary>

/// Handles the Click event of the button4 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void button4\_Click(object sender, EventArgs e)

{

try

{

if (training.AllExercise(date1.Value, date2.Value) == "0")

{

MessageBox.Show("Нет выполенных упражнений");

}

else MessageBox.Show(training.AllExercise(date1.Value, date2.Value));

}

catch (Exception ex)

{

MessageBox.Show(Convert.ToString(ex));

}

}

/// <summary>

/// Handles the SelectedIndexChanged event of the comboBox1 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the ValueChanged event of the dateTimePicker2 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void dateTimePicker2\_ValueChanged(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the ValueChanged event of the dateTimePicker1 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the CellContentClick event of the dataGridView1 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="DataGridViewCellEventArgs"/> instance containing the event data.</param>

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

/// <summary>

/// Creates new table.

/// </summary>

public void Update()

{

training.index = 0;

Table.RowCount = training.GetLenght;

for (int i = 0; i < training.GetLenght; i++)

{

training.index = i;

Table.Rows[i].Cells[0].Value = training.GetData;

Table.Rows[i].Cells[1].Value = training.GetEndData;

Table.Rows[i].Cells[2].Value = training.GetTypeOfExercise;

}

}

/// <summary>

/// Handles the Click event of the button6 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void button6\_Click(object sender, EventArgs e)

{

Edit edit= new Edit(this);

edit.Show();

}

}

}

AddForm.

using System;

using System.Windows.Forms;

namespace WindowsFormsApp

{

/// <summary>

/// <br />

/// </summary>

/// <seealso cref="System.Windows.Forms.Form" />

public partial class AddExercise : Form

{

/// <summary>

/// The main form

/// </summary>

private MainForm mainForm;

/// <summary>

/// Initializes a new instance of the <see cref="AddExercise"/> class.

/// </summary>

/// <param name="mainForm">The main form.</param>

public AddExercise(MainForm mainForm)

{

this.mainForm = mainForm;

InitializeComponent();

}

/// <summary>

/// Handles the Load event of the AddExercise control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void AddExercise\_Load(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the SelectedIndexChanged event of the comboBox1 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the ValueChanged event of the dateTimePicker2 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void dateTimePicker2\_ValueChanged(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the Click event of the button1 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void button1\_Click(object sender, EventArgs e)

{

int selectedType = type.SelectedIndex;

try

{

if(type.SelectedItem == null)

{

MessageBox.Show("Выберите упражнение");

}

else

{

mainForm.training.AddExercise(dateTime.Value, selectedType);

mainForm.Update();

}

}

catch (Exception ex)

{

MessageBox.Show(Convert.ToString(ex));

}

Close();

}

/// <summary>

/// Handles the Click event of the label1 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void label1\_Click(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the Click event of the label2 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void label2\_Click(object sender, EventArgs e)

{

}

}

}

EditExersise

using FitnessCenter;

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 WindowsFormsApp

{

/// <summary>

/// Edit

/// </summary>

/// <seealso cref="System.Windows.Forms.Form" />

public partial class Edit : Form

{

/// <summary>

/// The main form

/// </summary>

private MainForm mainForm;

/// <summary>

/// Initializes a new instance of the <see cref="Edit"/> class.

/// </summary>

/// <param name="mainForm">The main form.</param>

public Edit(MainForm mainForm)

{

this.mainForm = mainForm;

InitializeComponent();

Initializer();

}

/// <summary>

/// Handles the Load event of the Edit control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void Edit\_Load(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the SelectedIndexChanged event of the type control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void type\_SelectedIndexChanged(object sender, EventArgs e)

{

}

/// <summary>

/// Handles the ValueChanged event of the dateTime control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void dateTime\_ValueChanged(object sender, EventArgs e)

{

}

private void Initializer()

{

try

{

int index = mainForm.Table.CurrentRow.Index;

dateTime.Value = (DateTime)mainForm.Table.Rows[index].Cells[0].Value;

type.Text = (string)mainForm.Table.Rows[index].Cells[2].Value;

}

catch

{

MessageBox.Show("Выберите упражнение");

}

}

/// <summary>

/// Handles the Click event of the button1 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void button1\_Click(object sender, EventArgs e)

{

try

{

mainForm.training.EditExercise(dateTime.Value, type.SelectedIndex, mainForm.Table.CurrentRow.Index);

mainForm.Update();

}

catch

{

MessageBox.Show("Индекс находится вне границ");

}

Close();

}

/// <summary>

/// Handles the SelectedIndexChanged event of the comboBox2 control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}