НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Дисциплина: «Программирование»

Контрольное домашнее задание
3 модуль
Вариант 12

Выполнил
Студент группы БПИ173
_____/Д. М. Торилов/
«___» _____ 2018 г.

Преподаватель: Максименкова О.В.,
старший преподаватель
департамента
программной инженерии
факультета компьютерных наук

Москва 2017

Содержание

1. Условие задачи

Программа контрольного домашнего задания (КДЗ) должна представлять собой небольшую информационно-справочную систему (ИСС), основанную на файлах. В стандартном файле содержатся данные о землетрясениях. Данные из него загружаются в основную таблицу ИСС.

Далее следует описание задания варианта №12:

Для представления данных о землетрясении использовать класс EarthQuake. Координаты землетрясения представлять объектом структуры. Класс QuakeInfo связан с объектами EarthQuake отношением агрегации и позволяет получать списки землетрясений, сгруппированные по количеству уловивших их станций; списки землетрясений с максимальной магнитурой; землетрясение произошедшее на минимальной и максимальной глубине. Модифицировать интерфейс так, чтобы указанные данные можно было отобразить.

2. Функции разрабатываемого приложения

2.1. Варианты использования

Данная ИСС может быть использована для проведения исследований в области изучения землетрясений, в том числе в научных и образовательных целях.

2.2. Описание интерфейса пользователя

Интерфейс программы реализован на русском языке. В реализации использована технология Windows Forms. В дальнейшем будет совершён переход на технологию WPF.
Интерфейс программы включает в себя:

2.2.1. Панель меню

1. Кнопка **Файл**, даёт доступ к кнопкам:
    1.1. Кнопка **Открыть**, позволяет вызвать меню выбора файла для открытия
    1.2. Кнопка **Сохранить**, позволяет вызвать меню выбора файла для сохранения, либо сохранить в уже открытый файл, даёт доступ к кнопкам:
        1.2.1. **Дозаписать**, позволяет вызвать меню выбора файла для дозаписи таблицы в его конец, либо сохранить в уже открытый файл
    1.3. Кнопка **Сохранить как**, позволяет вызвать меню выбора файла для сохранения, даёт доступ к кнопкам:
        1.3.1. **Дозаписать**, позволяет вызвать меню выбора файла для дозаписи таблицы в его конец

1.4. **Закрыть**, позволяет прекратить работу с файлом, вызывает диалоговое окно с предложением сохранить файл перед закрытием

2. Кнопка **Таблица**, даёт доступ к кнопкам:

2.1. Кнопка **Сортировка**, даёт доступ к кнопкам:

2.1.1. **По номеру**, позволяет отсортировать таблицу по возрастанию идентификационных номеров элементов

2.1.2. **По станциям**, позволяет отсортировать таблицу по вохрастанию числа в количественной характеристике станций, засёкших землетрясение, соответствующее элементу

2.2. Кнопка **Удалить выделенную строку**, позволяет удалить выделенную строку из таблицы

3. Кнопка **Информация**, даёт доступ к кнопкам:

3.1. **Предельные значения**, позволяет увидеть короткий список диапазонов изменения характеристических величин элементов таблицы

3.2. **О программе**, позволяет увидеть информацию об авторе программы

2.2.2. Панель управления

1. Поле **Добавление элемента** включает в себя

1.1. Поле ввода **Номер элемента**

1.2. Поле ввода **Широта элемента**

1.3. Поле ввода **Долгота элемента**

1.4. Поле ввода **Глубина элемента**

1.5. Поле ввода **Магнитуда элемента**

1.6. Поле ввода **Станции элемента**

1.7. Кнопка **Добавить**, создаёт элемент с характеристиками 1.1 - 1.6 и добавляет его в таблицу

2. Поле **Землетрясения** включает в себя два подполя, **Максимальная глубина** (информация о землетрясении с максимальной глубиной) и **Минимальная глубина** (информация о землетрясении с минимальной глубиной), несущие полную информацию о таковых в таблице в режиме реального времени

3. Поле **Фильтрация по магнитуде** включает в себя два

3.1. Поле ввода **Магнитуда** для ввода вещественного значения магнитуды

3.2. Кнопка **Фильтровать**, при нажатии удаляет из таблицы все данные о землетрясениях, магнитудой ниже чем в поле 3.1

4. Кнопка **Выход** позволяет выйти из приложения

2.2.3. Панель работы с таблицей

Представляет из себя матрицу неограниченного числа строк и 6 столбцов с возможностью редактирования

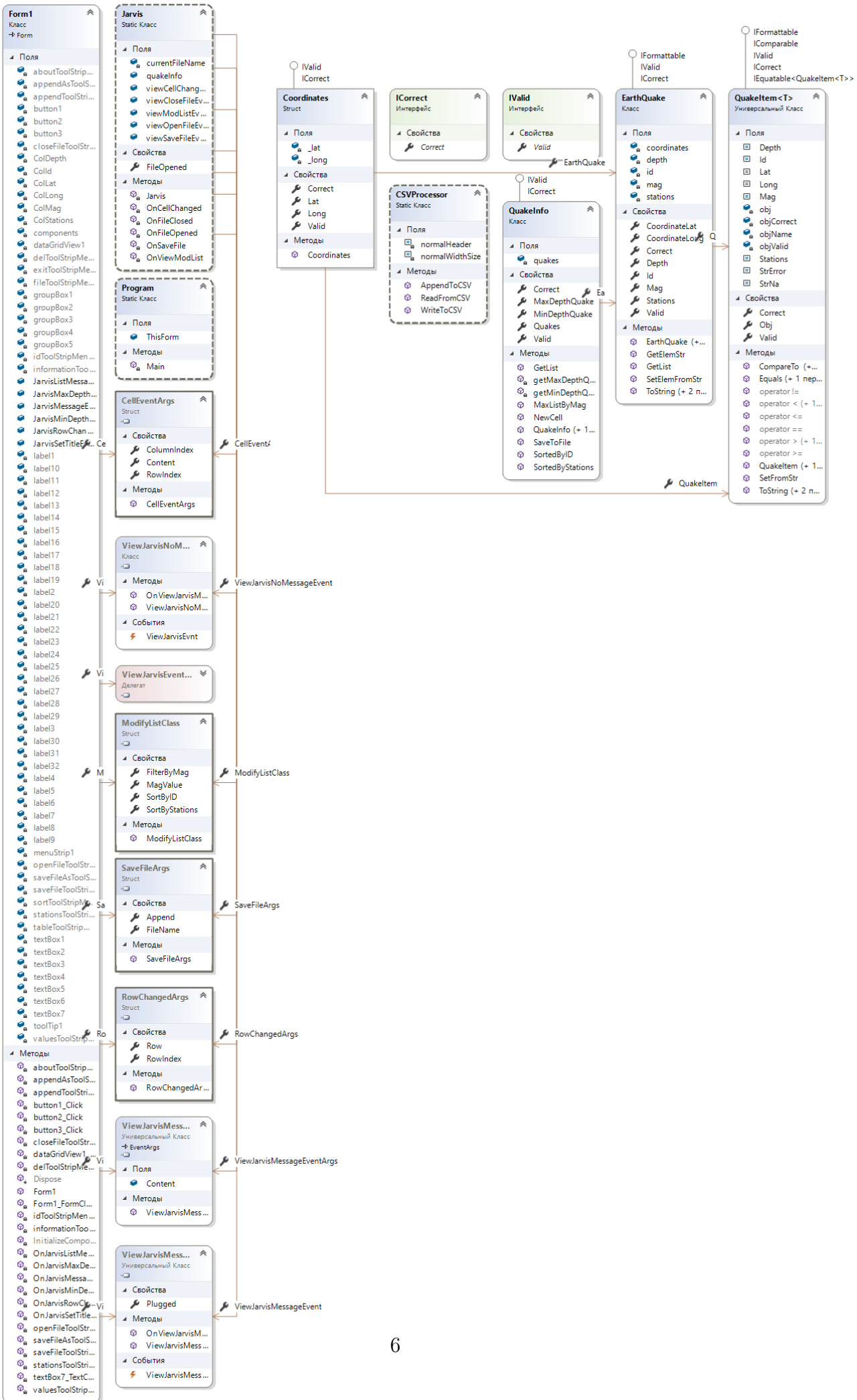
2.2.4. Всплывающие подсказки

Появляются при наведении курсора мыши на необходимый элемент. Несут дополнительную информацию об объекте.

3. Структура приложения

Приложение реализовано с использованием паттерна Model-View-Controller. Model реализована объектом класса QuakeInfo. View реализована объектом Form1. Задачу Controller выполняет статический класс Jarvis. Ниже приведены необходимые для понимания детали структуры приложения.

3.1. Диаграмма классов


Form1
Класс
Form
Поля
Методы
Jarvis
Static Класс
Поля
currentFileName
quakeInfo
viewCellChang...
viewCloseFileEv...
viewModListEv...
viewOpenFileEv...
viewSaveFileEv...
Свойства
FileOpened
Методы
Jarvis
OnCellChanged
OnFileClosed
OnFileOpened
OnSaveFile
OnViewModList
Program
Static Класс
Поля
ThisForm
Методы
Main
IValid
ICorrect
Coordinates
Struct
Поля
_lat
_long
Свойства
Correct
Lat
Long
Valid
Методы
Coordinates
ICorrect
Интерфейс
Свойства
Correct
IValid
Интерфейс
Свойства
Valid
EarthQuake
IFormattable
IValid
ICorrect
EarthQuake
Класс
Поля
coordinates
depth
id
mag
stations
Свойства
CoordinateLat
CoordinateLong
Correct
Depth
Id
Mag
Stations
Valid
Методы
EarthQuake (+...
GetElemStr
GetList
SetElemFromStr
ToString (+ 2 п...
IFormattable
IComparable
IValid
ICorrect
IEquatable<QuakeItem<T>>
QuakeItem<T>
Универсальный Класс
Поля
Depth
Id
Lat
Long
Mag
obj
objCorrect
objName
objValid
Stations
StrError
StrNa
Свойства
Correct
Obj
Valid
Методы
CompareTo (+...
Equals (+ 1 пер...
operator !=
operator < (+ 1...
operator <=
operator ==
operator > (+ 1...
operator >=
QuakeItem (+ 1...
SetFromStr
ToString (+ 2 п...
CSVProcessor
Static Класс
Поля
normalHeader
normalWidthSize
Методы
AppendToCSV
ReadFromCSV
WriteToCSV
IValid
ICorrect
QuakeInfo
Класс
Поля
quakes
Свойства
Correct
MaxDepthQuake
MinDepthQuake
Quakes
Valid
Методы
GetList
getMaxDepthQ...
getMinDepthQ...
MaxListByMag
NewCell
QuakeInfo (+ 1...
SaveToFile
SortedByID
SortedByStations
QuakeItem
CellEventArgs
Struct
Свойства
ColumnIndex
Content
RowIndex
Методы
CellEventArgs
CellEventA
ViewJarvisNoM...
Класс
Методы
OnViewJarvisM...
ViewJarvisNoM...
События
ViewJarvisEvnt
ViewJarvisNoMessageEvent
ViewJarvisEvent...
Делегат
ModifyListClass
Struct
Свойства
FilterByMag
MagValue
SortByID
SortByStations
Методы
ModifyListClass
SaveFileArgs
Struct
Свойства
Append
FileName
Методы
SaveFileArgs
RowChangedArgs
Struct
Свойства
Row
RowIndex
Методы
RowChangedAr...
ViewJarvisMess...
Универсальный Класс
EventArgs
Поля
Content
Методы
ViewJarvisMess...
ViewJarvisMessageEventArgs
ViewJarvisMess...
Универсальный Класс
Свойства
Plugged
Методы
OnViewJarvisM...
ViewJarvisMess...
События
ViewJarvisMess...
ViewJarvisMessageEvent
Методы
aboutToolStrip...
appendAsToolS...
appendToolStri...
button1_Click
button2_Click
button3_Click
closeFileToolStr...
dataGridView1...
delToolStripMe...
Dispose
Form1
Form1_FormCl...
idToolStripMen...
informationToo...
InitializeCompo...
OnJarvisListMe...
OnJarvisMaxDe...
OnJarvisMessa...
OnJarvisMinDe...
OnJarvisRowCh...
OnJarvisSetTitle...
openFileToolStr...
saveFileAsToolS...
saveFileToolStri...
stationsToolStri...
textBox7_TextC...
valuesToolStrip...

3.2. Описание классов, их полей и методов

3.2.1. Jarvis

Выполняет задачу синхронизации Model и View.
Объект класса QuakeInfo связан со статическим классом Jarvis отношением агрегации. Планируется сделать их менее зависимыми с возможностью обмена данными через события.
С объектом Form1 реализовано гибкое общение через события.

Поля:

- public static QuakeInfo quakeInfo - Модель агрегирована в контроллер
- private static string currentFileName = null; - Имя файла, с которым ведётся работа

Свойства:

- public static bool FileOpened => currentFileName != null; - Свойство корректного открытия файла

Методы:

- private static void OnViewModList(object sender, ViewJarvisMessageEventArgs <ModifyListClass> messageEventArgs) - Обработчик вызова модификатора списка (сортировки и фильтрации)
- private static void OnFileOpened(object sender, ViewJarvisMessageEventArgs<string> messageEventArgs) - Обработчик открытия файла
- private static void OnSaveFile(object sender, ViewJarvisMessageEventArgs<SaveFileArgs> messageEventArgs) - Обработчик сохранения файла
- private static void OnFileClosed() - Обработчик закрытия файла
- private static void OnCellChanged(object sender, ViewJarvisMessageEventArgs <CellEventArgs> messageEventArgs) - Обработчик изменения ячейки
- static Jarvis() - Конструктор

События:

- public static ViewJarvisMessageEvent<string> viewOpenFileEvent = new ViewJarvisMessageEvent<string>(); - Событие обработчика открытия файла
- public static ViewJarvisMessageEvent<SaveFileArgs> viewSaveFileEvent = new ViewJarvisMessageEvent<SaveFileArgs>(); - Событие обработчика сохранения файла

- public static ViewJarvisNoMessageEvent viewCloseFileEvent = new ViewJarvisNoMessageEvent(); - Событие закрытия файла
- public static ViewJarvisMessageEvent<CellEventArgs> viewCellChangedEvent = new ViewJarvisMessageEvent<CellEventArgs>(); - Событие обработчика изменения ячейки таблицы
- public static ViewJarvisMessageEvent<ModifyListClass> viewModListEvent = new ViewJarvisMessageEvent<ModifyListClass>(); - Событие обработчика модификации списка

3.2.2. Form

Представляет реализацию View. Обменивается данными с Jarvis через события

Методы:

- private void OnJarvisMessageEvent(object sender, ViewJarvisMessageEventArgs<string> messageEventArgs) - Обработчик осбытия получения сообщения от класса Jarvis
- private void OnJarvisListMessageEvent(object sender, ViewJarvisMessageEventArgs <List<List<string»> messageEventArgs) - Обработчик события получения таблицы от класса Jarvis
- private void OnJarvisSetTitleEvent(object sender, ViewJarvisMessageEventArgs<string> messageEventArgs) - Обработчик события вывода сообщения
- private void OnJarvisRowChangedEvent(object sender, ViewJarvisMessageEventArgs <RowChangedArgs> messageEventArgs) - Обработчик события изменения строки
- private void OnJarvisMaxDepthUpdatedEvent(object sender, ViewJarvisMessageEventArgs <List<string» messageEventArgs) - Обработчик обновления информации об элементе с самой большой характеристикой глубины
- private void OnJarvisMinDepthUpdatedEvent(object sender, ViewJarvisMessageEventArgs <List<string» messageEventArgs) - Обработчик события обновления информации об элементе с минимальной характеристикой глубины
- public Form1() - Конструктор View
- private void openFileToolStripMenuItemClick(object sender, EventArgs e) - Обработчик нажатия кнопки открытия файла
- private void closeFileToolStripMenuItemClick(object sender, EventArgs e) - Обработчик нажатия кнопки закрытия файла
- private void dataGridView1CellValueChanged(object sender, DataGridViewCellEventArgs e) - Обработчик изменения ячейки таблицы

- private void aboutToolStripMenuItemClick(object sender, EventArgs e) - Справка
- private void valuesToolStripMenuItemClick(object sender, EventArgs e) - Предельные величины
- private void button1 Click(object sender, EventArgs e) - Кнопка выхода - обработчик
- private void button2 Click(object sender, EventArgs e) - Обработчик нажатия кнопки добавления строки
- private void Form1 FormClosing1(object sender, FormClosingEventArgs e) - Обработчик закрытия формы
- private void idToolStripMenuItemClick(object sender, EventArgs e) - Обработчик вызова сортировки по ID
- private void stationsToolStripMenuItemClick(object sender, EventArgs e) - Обработчик вызова сортировки по станциям
- private void button3 Click(object sender, EventArgs e) - Обработчик нажатия кнопки фильтрации
- private void delToolStripMenuItemClick(object sender, EventArgs e) - Обработчик нажатия кнопки удаления строки в таблице
- private void appendToolStripMenuItemClick(object sender, EventArgs e) - Обработчик нажатия кнопки дозаписи в файл
- private void appendAsToolStripMenuItem1Click(object sender, EventArgs e) - Обработчик нажатия кнопки дозаписи в файл
- private void saveFileAsToolStripMenuItemClick(object sender, EventArgs e) - Обработчик нажатия кнопки сохранения файла
- private void saveFileToolStripMenuItemClick(object sender, EventArgs e) - Обработчик нажатия кнопки сохранения файла

События:

- public static ViewJarvisMessageEvent<string> JarvisMessageEvent = new ViewJarvisMessageEvent<string>(); - Событие получения сообщения от класса Jarvis
- public static ViewJarvisMessageEvent<List<List<string»> JarvisListMessageEvent = new ViewJarvisMessageEvent<List<List<string»>(); - Событие получения таблицы от класса Jarvis
- public static ViewJarvisMessageEvent<string> JarvisSetTitleEvent = new ViewJarvisMessageEvent<string>(); - Событие вывода сообщения

- public static ViewJarvisMessageEvent<RowChangedArgs> JarvisRowChangedEvent = new ViewJarvisMessageEvent<RowChangedArgs>(); - Событие изменения строки
- public static ViewJarvisMessageEvent<List<string» JarvisMaxDepthUpdatedEvent = new ViewJarvisMessageEvent<List<string»(); - Событие обновления информации об элементе с самой большой характеристикой глубины
- public static ViewJarvisMessageEvent<List<string» JarvisMinDepthUpdatedEvent = new ViewJarvisMessageEvent<List<string»(); - Событие обновления информации об элементе с минимальной характеристикой глубины

4. Распределение исходного кода по файлам проекта

Описание модели содержится в файле QuakeInfo.cs. Все файлы из библиотеки классов ModelLibrary являются вспомогательными.
Описание контроллера содержится в файле Jarvis.cs.
Работа с интерфейсом описана в Form1.cs. События для обмена информацией между Jarvis и Form1 описаны в библиотеке классов EventsLibrary.

5. Контрольный пример и описание результатов

(Контрольный пример – это аккуратно описанная последовательность действий, позволяющая проверить корректность работы функций программы по шагам. Это предполагает для каждого шага наличие входных и выходных данных или состояния интерфейса и т.п.)

6. Текст (код) программы

6.1. Events Library

6.1.1. ViewJarvisMessage

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace EventsLibrary
{
    public class ViewJarvisMessageEventArgs<T> : EventArgs
    {
        public T Content;
```

```
    }
    public class ViewJarvisMessageEvent<T>
    {
        private bool plugged = true;
        public bool Plugged {
            get { return plugged; }
            set { plugged = value; }
        }

        public event EventHandler<ViewJarvisMessageEventArgs<T>> ViewJarv

        public void OnViewJarvisMessage(T content)
        {
            ViewJarvisMessageEventArgs<T> args = new ViewJarvisMessageEve

            if (ViewJarvisMessageEvnt != null && plugged)
            {
                args.Content = content;
                ViewJarvisMessageEvnt(this, args);
            }
        }
    }
}
```

6.2. KDZ

6.2.1. Form1

```
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;
using EventsLibrary;

namespace KDZ_2_12_
{
    /// <summary>
```

```
/// View
/// </summary>
public partial class Form1 : Form
{
    public static ViewJarvisMessageEvent<string> JarvisMessageEvent =
    private void OnJarvisMessageEvent(object sender, ViewJarvisMessag
    {
        string message = messageEventArgs.Content;
        MessageBox.Show(message, " C                    ");
    }

    public static ViewJarvisMessageEvent<List<List<string>>> JarvisLi


    private void OnJarvisListMessageEvent(object sender, ViewJarvisMe
    {
        dataGridView1.Rows.Clear();
        dataGridView1.Refresh();

        List<List<string>> list = messageEventArgs.Content;
        foreach (List<string> row in list)
        {
            dataGridView1.Rows.Add(row.ToArray());
        }
    }

    public static ViewJarvisMessageEvent<string> JarvisSetTitleEvent

    private void OnJarvisSetTitleEvent(object sender, ViewJarvisMessa
    {
        string title = messageEventArgs.Content;
        Program.ThisForm.Text = $"
    }

    public static ViewJarvisMessageEvent<RowChangedArgs> JarvisRowCha

    private void OnJarvisRowChangedEvent(object sender, ViewJarvisMes
    {
        RowChangedArgs args = messageEventArgs.Content;
        Jarvis.viewCellChangedEvent.Plugged = false;
```

```
        for (int i = 0; i < args.Row.Count; i++)
        {
            dataGridView1.Rows[args.RowIndex].Cells[i].Value = args.F
        }
        Jarvis.viewCellChangedEvent.Plugged = true;
        dataGridView1.Refresh();
    }

    public static ViewJarvisMessageEvent<List<string>> JarvisMaxDepth

    private void OnJarvisMaxDepthUpdatedEvent(object sender, ViewJarv
    {
        List<string> args = messageEventArgs.Content;
        label23.Text = args[0];
        label3.Text = args[1];
        label5.Text = args[2];
        label7.Text = args[3];
        label9.Text = args[4];
        label11.Text = args[5];
    }

    public static ViewJarvisMessageEvent<List<string>> JarvisMinDepth

    private void OnJarvisMinDepthUpdatedEvent(object sender, ViewJarv
    {
        List<string> args = messageEventArgs.Content;
        label24.Text = args[0];
        label20.Text = args[1];
        label18.Text = args[2];
        label16.Text = args[3];
        label14.Text = args[4];
        label12.Text = args[5];
    }

    public Form1()
    {
        InitializeComponent();
        JarvisMessageEvent.ViewJarvisMessageEvnt += OnJarvisMessageEv
        JarvisListMessageEvent.ViewJarvisMessageEvnt += OnJarvisListM
        JarvisRowChangedEvent.ViewJarvisMessageEvnt += OnJarvisRowCha
```

```
        JarvisSetTitleEvent.ViewJarvisMessageEvnt += OnJarvisSetTitle
        JarvisMaxDepthUpdatedEvent.ViewJarvisMessageEvnt += OnJarvisM
        JarvisMinDepthUpdatedEvent.ViewJarvisMessageEvnt += OnJarvisM

        label23.Text = "NA";
        label3.Text = "NA";
        label5.Text = "NA";
        label7.Text = "NA";
        label9.Text = "NA";
        label11.Text = "NA";
        label24.Text = "NA";
        label20.Text = "NA";
        label18.Text = "NA";
        label16.Text = "NA";
        label14.Text = "NA";
        label12.Text = "NA";
    }

    private void openFileToolStripMenuItem_Click(object sender, Event
    {
        if (Jarvis.FileOpened)
        {
            DialogResult dialogResult = MessageBox.Show("", "", Messa
            switch (dialogResult)
            {
                case DialogResult.Yes:
                    break;
                case DialogResult.No:
                    break;
                case DialogResult.Cancel:
                    return;
            }
        }

        dataGridView1.Rows.Clear();
        dataGridView1.Refresh();

        using (var openFileDialog = new OpenFileDialog())
        {
            openFileDialog.Filter = "Comma Separated Value(*.csv) | *
```

```
            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                Jarvis.viewOpenFileEvent.OnViewJarvisMessage(openFile
            }
        }
    }

    private void closeFileToolStripMenuItem_Click(object sender, Even
    {
        DialogResult dialogResult = MessageBox.Show("", "", MessageBo
        switch (dialogResult)
        {
            case DialogResult.Yes:
                dataGridView1.Rows.Clear();
                dataGridView1.Refresh();
                Jarvis.viewCloseFileEvent.OnViewJarvisMessage();
                break;
            case DialogResult.No:
                dataGridView1.Rows.Clear();
                dataGridView1.Refresh();
                Jarvis.viewCloseFileEvent.OnViewJarvisMessage();
                break;
            case DialogResult.Cancel:
                return;
        }
    }

    private void dataGridView1_CellValueChanged(object sender, DataG
    {
        try
        {
            Jarvis.viewCellChangedEvent.OnViewJarvisMessage(new CellE
        }
        catch
        {

        }
    }
```

```
private void button1_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("", "",
        MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        Application.ExitThread();
    }
}

private void button2_Click(object sender, EventArgs e)
{
    dataGridView1.Rows.Add("NA", "NA", "NA", "NA", "NA", "NA");
    List<string> list = new List<string>
    {
        textBox1.Text,
        textBox2.Text,
        textBox3.Text,
        textBox4.Text,
        textBox5.Text,
        textBox6.Text
    };

    int rowIndex = dataGridView1.RowCount - 2;

    for (int i = 0; i < 6; i++)
    {
        Jarvis.viewCellChangedEvent.OnViewJarvisMessage(new CellE
    }

    Jarvis.viewCellChangedEvent.Plugged = false;
    dataGridView1.Refresh();
    //dataGridView1.Rows.Add();
    //dataGridView1.Refresh();
    Jarvis.viewCellChangedEvent.Plugged = true;
}

private void Form1_FormClosing_1(object sender, FormClosingEventA
{
    if (MessageBox.Show("",
        MessageBoxButtons.YesNo) == DialogResult.No)
```

```
    {
        e.Cancel = true;
    }
}

private void idToolStripMenuItem_Click(object sender, EventArgs e
{
    Jarvis.viewModListEvent.OnViewJarvisMessage(new ModifyListCla

private void stationsToolStripMenuItem_Click(object sender, Event
{
    Jarvis.viewModListEvent.OnViewJarvisMessage(new ModifyListCla
}

private void textBox7_TextChanged(object sender, EventArgs e)
{

}

private void button3_Click(object sender, EventArgs e)
{
    double val;
    if (double.TryParse(textBox7.Text, out val) && val >= 1 && va
    {
        if (MessageBox.Show($" {val}", "", MessageBoxButtons.OKC
        {
            Jarvis.viewModListEvent.OnViewJarvisMessage(new Modi
        }
    }
    else
    {
        MessageBox.Show("", MessageBoxButtons.OK);
    }
}

private void delToolStripMenuItem_Click(object sender, EventArgs
{
    try
    {
        if ((dataGridView1.RowCount < 1) || (dataGridView1.Column
```

```
            {
                throw new ArgumentException("
            }
            else
            {
                if (dataGridView1.SelectedRows.Count > 0)
                {
                    Jarvis.quakeInfo.Quakes.RemoveAt(dataGridView1.Se
                    dataGridView1.Rows.RemoveAt(dataGridView1.Selected
                    JarvisMinDepthUpdatedEvent.OnViewJarvisMessage(Ja
                    JarvisMaxDepthUpdatedEvent.OnViewJarvisMessage(Ja
                }
                else
                {
                    throw new ArgumentException("
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "                    ");
        }
    }

    private void appendToolStripMenuItem_Click(object sender, EventAr
    {
        if (Jarvis.FileOpened)
        {
            Jarvis.viewSaveFileEvent.OnViewJarvisMessage(new SaveFile
        }
        else
        {
            using (var saveFileDialog = new SaveFileDialog())
            {
                saveFileDialog.Filter = "Comma Separated Value(*.csv)

                if (saveFileDialog.ShowDialog() == DialogResult.OK)
                {
                    Jarvis.viewSaveFileEvent.OnViewJarvisMessage(new
```

```
            }
        }
    }
}

private void appendAsToolStripMenuItem1_Click(object sender, Even
{
    using (var saveFileDialog = new SaveFileDialog())
    {
        saveFileDialog.Filter = "Comma Separated Value(*.csv) | *

        if (saveFileDialog.ShowDialog() == DialogResult.OK)
        {
            Jarvis.viewSaveFileEvent.OnViewJarvisMessage(new Save
        }
    }
}

private void saveFileAsToolStripMenuItem_Click(object sender, Eve
{
    using (var saveFileDialog = new SaveFileDialog())
    {
        saveFileDialog.Filter = "Comma Separated Value(*.csv) | *

        if (saveFileDialog.ShowDialog() == DialogResult.OK)
        {
            Jarvis.viewSaveFileEvent.OnViewJarvisMessage(new Save
        }
    }
}

private void saveFileToolStripMenuItem_Click(object sender, Event
{
    if (Jarvis.FileOpened)
    {
        Jarvis.viewSaveFileEvent.OnViewJarvisMessage(new SaveFile
    }
    else
    {
        using (var saveFileDialog = new SaveFileDialog())
```

```
            {
                saveFileDialog.Filter = "Comma Separated Value(*.csv)

                if (saveFileDialog.ShowDialog() == DialogResult.OK)
                {

                    Jarvis.viewSaveFileEvent.OnViewJarvisMessage(new
                }
            }
        }
    }

    public ViewJarvisMessageEvent<object> ViewJarvisMessageEvent
    {
        get => default(ViewJarvisMessageEvent<object>);
        set
        {
        }
    }

    public RowChangedArgs RowChangedArgs
    {
        get => default(RowChangedArgs);
        set
        {
        }
    }

    public ViewJarvisMessageEventArgs<object> ViewJarvisMessageEventA
    {
        get => default(ViewJarvisMessageEventArgs<object>);
        set
        {
        }
    }

    public SaveFileArgs SaveFileArgs
    {
        get => default(SaveFileArgs);
        set
```

```
            {
            }
        }

        public ModifyListClass ModifyListClass
        {
            get => default(ModifyListClass);
            set
            {
            }
        }

        public ViewJarvisEventDelegate ViewJarvisEventDelegate
        {
            get => default(ViewJarvisEventDelegate);
            set
            {
            }
        }

        public ViewJarvisNoMessageEvent ViewJarvisNoMessageEvent
        {
            get => default(ViewJarvisNoMessageEvent);
            set
            {
            }
        }

        public CellEventArgs CellEventArgs
        {
            get => default(CellEventArgs);
            set
            {
            }
        }
    }
}
```

6.2.2. Jarvis

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ModelLibrary;
using EventsLibrary;
using System.Globalization;

namespace KDZ_2_12_
{
    public static class Jarvis
    {
        public static QuakeInfo quakeInfo;
        private static string currentFileName = null;
        public static bool FileOpened => currentFileName != null;

        public static ViewJarvisMessageEvent<string> viewOpenFileEvent =
        public static ViewJarvisMessageEvent<SaveFileArgs> viewSaveFileEv
        public static ViewJarvisNoMessageEvent viewCloseFileEvent = new V
        public static ViewJarvisMessageEvent<CellEventArgs> viewCellChang
        public static ViewJarvisMessageEvent<ModifyListClass> viewModList

        private static void OnViewModList(object sender, ViewJarvisMessag
        {
            ModifyListClass mod = messageEventArgs.Content;
            if (mod.SortByID)
            {
                quakeInfo.Quakes = quakeInfo.SortedByID();
            }
            if (mod.SortByStations)
            {
                quakeInfo.Quakes = quakeInfo.SortedByStations();
            }
            if (mod.FilterByMag)
            {
                quakeInfo.Quakes = quakeInfo.MaxListByMag(mod.MagValue);
            }

            Form1.JarvisListMessageEvent.OnViewJarvisMessage(quakeInfo.G
            Form1.JarvisMinDepthUpdatedEvent.OnViewJarvisMessage(quakeInf
            Form1.JarvisMaxDepthUpdatedEvent.OnViewJarvisMessage(quakeInf
```

```
}

private static void OnFileOpened(object sender, ViewJarvisMessage
{
    string fileName = messageEventArgs.Content;
    try
    {
        quakeInfo = new QuakeInfo(fileName);
        Form1.JarvisListMessageEvent.OnViewJarvisMessage(quakeInf
        currentFileName = fileName;
        Form1.JarvisSetTitleEvent.OnViewJarvisMessage(fileName);
        if (!quakeInfo.Valid || !quakeInfo.Correct)
        {
            Form1.JarvisMessageEvent.OnViewJarvisMessage($"");
        }
    }
    catch (ArgumentException e)
    {
        switch (e.ParamName)
        {
            case "file":
                Form1.JarvisMessageEvent.OnViewJarvisMessage($""
                OnFileClosed();
                break;
        }
    }
    Form1.JarvisMinDepthUpdatedEvent.OnViewJarvisMessage(quakeInf
    Form1.JarvisMaxDepthUpdatedEvent.OnViewJarvisMessage(quakeInf
}

private static void OnSaveFile(object sender, ViewJarvisMessageEv
{
    SaveFileArgs args = messageEventArgs.Content;
    if (args.FileName != null && args.Append == false)
    {
        currentFileName = args.FileName;
    }
    if (args.Append)
```

```
        {
            quakeInfo.SaveToFile(args.FileName, "append");
        }
        else
        {
            quakeInfo.SaveToFile(args.FileName, "rewrite");
            Form1.JarvisSetTitleEvent.OnViewJarvisMessage(currentFileN
        }
    }

    private static void OnFileClosed()
    {
        quakeInfo = new QuakeInfo();
        currentFileName = null;
        Form1.JarvisSetTitleEvent.OnViewJarvisMessage("");
        Form1.JarvisMinDepthUpdatedEvent.OnViewJarvisMessage(quakeInf
        Form1.JarvisMaxDepthUpdatedEvent.OnViewJarvisMessage(quakeInf
    }

    private static void OnCellChanged(object sender, ViewJarvisMessag
    {
        // newCell.content.tostring
        CellEventArgs newCell = messageEventArgs.Content;
        if (newCell.RowIndex != -1)
        {
            List<string> list = quakeInfo.NewCell(newCell.Content.To
            Form1.JarvisRowChangedEvent.OnViewJarvisMessage(new RowC
            Form1.JarvisMinDepthUpdatedEvent.OnViewJarvisMessage(qual
            Form1.JarvisMaxDepthUpdatedEvent.OnViewJarvisMessage(qual
        }
    }

    static Jarvis()
    {
        viewOpenFileEvent.ViewJarvisMessageEvnt += OnFileOpened;
        viewCloseFileEvent.ViewJarvisEvnt += OnFileClosed;
        viewCellChangedEvent.ViewJarvisMessageEvnt += OnCellChanged;
        quakeInfo = new QuakeInfo();
        Form1.JarvisMinDepthUpdatedEvent.OnViewJarvisMessage(quakeInf
        Form1.JarvisMaxDepthUpdatedEvent.OnViewJarvisMessage(quakeInf
```

```
        viewModListEvent.ViewJarvisMessageEvnt += OnViewModList;
        viewSaveFileEvent.ViewJarvisMessageEvnt += OnSaveFile;
    }

    public static CellEventArgs CellEventArgs
    {
        get => default(CellEventArgs);
        set
        {
        }
    }

    public static ViewJarvisNoMessageEvent ViewJarvisNoMessageEvent
    {
        get => default(ViewJarvisNoMessageEvent);
        set
        {
        }
    }

    public static ModifyListClass ModifyListClass
    {
        get => default(ModifyListClass);
        set
        {
        }
    }

    public static SaveFileArgs SaveFileArgs
    {
        get => default(SaveFileArgs);
        set
        {
        }
    }

    public static RowChangedArgs RowChangedArgs
    {
        get => default(RowChangedArgs);
        set
```

```
            {
            }
        }

        public static ViewJarvisMessageEventArgs<object> ViewJarvisMessag
        {
            get => default(ViewJarvisMessageEventArgs<object>);
            set
            {
            }
        }

        public static ViewJarvisMessageEvent<object> ViewJarvisMessageEve
        {
            get => default(ViewJarvisMessageEvent<object>);
            set
            {
            }
        }
    }
}
```

6.3. ModelLibrary

6.3.1. QuakeInfo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Globalization;


namespace ModelLibrary
{
    public class QuakeInfo : IValid, ICorrect
    {
        private List<EarthQuake> quakes = new List<EarthQuake>();
        public List<EarthQuake> Quakes {
            set { quakes = value; }
            get { return quakes; }
```

```
}
public bool Valid {
    get
    {
        bool res = true;
        for (int i = 0; i < quakes.Count && res; i++)
        {
            res = res && quakes[i].Valid;
        }
        return res;
    }
}
public bool Correct
{
    get
    {
        bool res = true;
        for (int i = 0; i < quakes.Count && res; i++)
        {
            res = res && quakes[i].Correct;
        }
        return res;
    }
}

public QuakeInfo() {
    quakes = new List<EarthQuake>();
}
public QuakeInfo(string filename)
{
    quakes = CSVProcessor.ReadFromCSV(filename);
}
public List<EarthQuake> SortedByID()
{
    List<EarthQuake> res = quakes;
    res.Sort((q1, q2) => q1.Id.CompareTo(q2.Id));
    return res;
}

public List<EarthQuake> SortedByStations()
```

```
{
    List<EarthQuake> res = quakes;
    res.Sort((q1, q2) => q1.Stations.CompareTo(q2.Stations));
    return res;
}


public List<EarthQuake> MaxListByMag(double minValue)
{
    List<EarthQuake> res = new List<EarthQuake>();

    foreach (EarthQuake quake in quakes)
    {
        if (quake.Mag > minValue)
        {
            res.Add(quake);
        }
    }
    return res;
}


public List<string> MinDepthQuake => getMinDepthQuake().GetList()

public List<string> MaxDepthQuake => getMaxDepthQuake().GetList()

private EarthQuake getMinDepthQuake()
{
    EarthQuake minDepthQuake = new EarthQuake();
    foreach (EarthQuake quake in quakes)
    {
        if (quake.Correct)
        {
            if (!minDepthQuake.Correct || quake.Depth < minDepthQ
            {
                minDepthQuake = quake;
            }
        }
    }
    return minDepthQuake;
```

```
}

private EarthQuake getMaxDepthQuake()
{
    EarthQuake maxDepthQuake = new EarthQuake();
    foreach (EarthQuake quake in quakes)
    {
        if (quake.Correct)
        {
            if (!maxDepthQuake.Correct || quake.Depth > maxDepthQ
            {
                maxDepthQuake = quake;
            }
        }
    }
    return maxDepthQuake;
}

public List<List<string>> GetList(CultureInfo cultureInfo=null)
{
    List<List<string>> res = new List<List<string>>();
    foreach (EarthQuake quake in quakes)
    {
        res.Add(quake.GetList(cultureInfo));
    }
    return res;
}

public List<string> NewCell(string val, int columnIndex, int rowI
{
    if (rowIndex >= quakes.Count)
    {
        EarthQuake earthQuake = new EarthQuake();
        earthQuake.SetElemFromStr(val, columnIndex, cultureInfo);
        quakes.Add(earthQuake);
        return earthQuake.GetList(cultureInfo);
    }
    else
    {
        quakes[rowIndex].SetElemFromStr(val, columnIndex, culture
```

```
                return quakes[rowIndex].GetList(cultureInfo);
            }
        }

        public void SaveToFile(string filename, string mode)
        {
            switch (mode)
            {
                case "rewrite":
                    CSVProcessor.WriteToCSV(quakes, filename);
                    break;
                case "append":
                    CSVProcessor.AppendToCSV(quakes, filename);
                    break;
            }
        }

        public EarthQuake EarthQuake
        {
            get => default(EarthQuake);
            set
            {
            }
        }
    }
}
```

6.3.2. CSVProcessor

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using System.Globalization;


namespace ModelLibrary
{
    static class CSVProcessor
    {

        const string normalHeader = "\"id\",\"lat\",\"long\",\"depth\",\"
```

```
const int normalWidthSize = 6;


public static List<EarthQuake> ReadFromCSV(string filename)
{
    List<EarthQuake> res = new List<EarthQuake>();

    using (StreamReader reader = new StreamReader(filename))
    {
        if(!reader.EndOfStream)
        {
            string headerLine = reader.ReadLine();
            if (headerLine != normalHeader)
            {
                throw new ArgumentException($"file {filename} is
            }
        }

        while (!reader.EndOfStream)
        {
            string line = reader.ReadLine();
            List<string> values = line.Split(',').ToList();
            if (values.Capacity != normalWidthSize)
            {
                throw new ArgumentException($"file {filename} is
            }
            res.Add(new EarthQuake(values, CultureInfo.GetCulture
        }
    }
    return res;
}

public static void WriteToCSV(List<EarthQuake> table, string filen
{
    using (StreamWriter writer = new StreamWriter(filename))
    {
        writer.WriteLine(normalHeader);
        foreach (EarthQuake item in table)
        {
```

```
                writer.WriteLine(item.ToString(CultureInfo.GetCulture
            }
        }
    }


    public static void AppendToCSV(List<EarthQuake> table, string file
    {
        using (StreamWriter writer = new StreamWriter(filename, appen
        {
            writer.WriteLine(normalHeader);
            foreach (EarthQuake item in table)
            {
                writer.WriteLine(item.ToString(CultureInfo.GetCulture
            }
        }
    }
}
```

7. Список литературы

1. Охота на мифический MVC. Построение пользовательского интерфейса - cobiot 27 февраля 2017 - https://habrahabr.ru/post/322700/
2. Changing the form title text programmatically - February 04, 2008
- https://social.msdn.microsoft.com/Forums/en-US/b958c565-94bb-464d-9075-5163384d3887
/changing-the-form-title-text-programmatically?forum=winforms
3. Географические координаты - Википедия - https://ru.wikipedia.org/wiki/
Географические_координаты
4. Землетрясение - Википедия - https://ru.wikipedia.org/wiki/Землетрясение
5. Магнитуда землетрясения - Википедия - https://ru.wikipedia.org/wiki/
Магнитуда_землетрясения
6. Обобщенные классы - https://professorweb.ru -
https://professorweb.ru/my/csharp/charp_theory/level11/11_2.php
7.openFileDialog, saveFileDialog - stackoverflow.com -
https://ru.stackoverflow.com/questions/87480/openfiledialog-savefiledialog
8. C 6.0. Справочник. Полное описание языка - Джозеф Албахари, Бен Албахари - O'Reilly
- C 6.0 in a Nutshell: The Definitive Reference - Joseph Albahari, Ben Albahari