Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

Кафедра ИС

ОТЧЕТ

по лабораторной работе №2

ИССЛЕДОВАНИЕ ТИПОВ ДАННЫХ, ОПРЕДЕЛЯЕМЫХ ПОЛЬЗОВАТЕЛЕМ. НАСЛЕДОВАНИЕ. ОБРАБОТКА ИСПЛЮЧЕНИЙ В C#

Выполнил:

ст. гр. ИС/б-21-о

Куркчи А. Э.

Проверил:

Шишкевич В. Е.

Севастополь

2016

1. ЦЕЛЬ РАБОТЫ

В ходе выполнения данной лабораторной работы необходимо познакомиться с пользовательскими типами данных в языке С#: структура и перечисление; ознакомиться со структурой класса, его созданием и использова­нием, описанием членов класса: полей, свойств, инициализации объек­тов класса с помощью конструкторов; изучить механизм создания иерархий классов в С# и применение интерфейсов при наследовании; изучить механизм генерации и обработки исключений.

1. ПОСТАНОВКА ЗАДАЧИ

Вариант № 12

1. Проработать примеры программ 1-6, данные в теоретических сведениях. Создать на их основе программы. Получить результаты ра­боты программ и уметь их объяснить. Внести их в отчет по работе с комментариями.
2. Для заданной структуры данных разработать абстрактный класс и класс-наследник. В классе реализовать несколько конструкторов. Создать методы, работающие с полями класса. Часть из них должны быть виртуальными. Добавить методы-свойства и индексаторы.
3. Разработать интерфейсные классы, добавляющие некоторые ме­тоды в класс-потомок. Изучить причины возникновения коллизий имен при наследовании и способы их устранения.
4. Разработать классы исключительных ситуаций и применить их для обработки, возникающих исключений.
5. Написать демонстрационную программу.
6. ТЕКСТ ПРОГРАММЫ

Файл Program.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Text**.**RegularExpressions**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Lab**.**\_02

**{**

class Program

**{**

static void Main**(**string**[]** args**)**

**{**

Console**.**WriteLine**(**"\t\tSevSU semestr.04 .NET Lab.02"**);**

Museum m **=** **new** Museum**(**"Центральный"**,** "ул. Пушкина"**,** "Город"**);**

m**[**0**]** **=** "Холл"**;**

m**[**1**]** **=** "Зал A"**;**

m**[**2**]** **=** "Зал Б"**;**

m**[**3**]** **=** "Зал В"**;**

m**[**4**]** **=** "Зал Г"**;**

Random rnd **=** **new** Random**();**

**for** **(**int i **=** 0**;** i **<** rnd**.**Next**(**100**);** i**++)**

**{**

m**.**visit**();**

**}**

**try**

**{**

m**.**Name **=** "Областной"**;**

**}**

**catch** **(**InvalidValueException ex**)**

**{**

Console**.**WriteLine**(**"Exception: {0}"**,** ex**.**Message**);**

**}**

m**.**print**();**

Console**.**WriteLine**(**"\tPress any key to exit..."**);**

Console**.**ReadKey**();**

**}**

**}**

**}**

Файл InvalidValueException.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Lab**.**\_02

**{**

class InvalidValueException**:** Exception

**{**

**new** **public** string Message **{** get**;** set**;** **}**

**public** InvalidValueException**(**FieldType fieldType**)**

**{**

**switch** **(**fieldType**)**

**{**

**case** FieldType**.**Name**:**

Message **=** "Name field can`t be changed"**;**

**break;**

**case** FieldType**.**Address**:**

Message **=** "Address field can`t be empty"**;**

**break;**

**case** FieldType**.**Index**:**

Message **=** "Index field is invalid"**;**

**break;**

**default:**

Message **=** "Unknown field is invalid"**;**

**break;**

**}**

**}**

**}**

enum FieldType

**{**

Name**,**

Address**,**

Index

**}**

**}**

Файл AMuseum.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Lab**.**\_02

**{**

**abstract** class AMuseum

**{**

**private** string name**;**

**public** string Name

**{**

get **{** **return** name**;** **}**

set **{** **if** **(**name **==** **null)** **{** name **=** value**;** **}** **else** **{** **throw** **new** InvalidValueException**(**FieldType**.**Name**);** **}** **}**

**}**

**private** string address**;**

**public** string Address

**{**

get **{** **return** address**;** **}**

set **{** **if** **(**address **!=** ""**)** **{** address **=** value**;** **}** **else** **{** **throw** **new** InvalidValueException**(**FieldType**.**Address**);** **}** **}**

**}**

**private** int hall\_count **=** 0**;**

**public** int Hall\_count

**{**

get **{** **return** hall\_count**;** **}**

**}**

const int hall\_max\_count **=** 100**;**

**private** string**[]** halls **=** **new** string**[**hall\_max\_count**];**

**public** string **this[**int i**]**

**{**

get

**{**

**if** **(**i **>=** 0 **&&** i **<=** hall\_count**)**

**{**

**return** halls**[**i**];**

**}**

**else**

**{**

**throw** **new** InvalidValueException**(**FieldType**.**Index**);**

**}**

**}**

set

**{**

**if** **(**i **==** hall\_count**)**

**{**

halls**[**i**]** **=** value**;**

hall\_count**++;**

**}**

**else** **if** **(**i **>=** 0 **&&** i **<=** hall\_count**)**

**{**

halls**[**i**]** **=** value**;**

**}**

**else**

**{**

**throw** **new** InvalidValueException**(**FieldType**.**Index**);**

**}**

**}**

**}**

**public** AMuseum**(**string name**,** string address**)**

**{**

**this.**name **=** name**;**

**this.**address **=** address**;**

**}**

**public** **virtual** void print**()**

**{**

Console**.**WriteLine**(**"Museum \"{0}\", {1}"**,** name**,** address**);**

**}**

**public** void printHalls**()**

**{**

**for** **(**int i **=** 0**;** i **<** hall\_count**;**i**++)**

**{**

Console**.**WriteLine**(**"Hall #{0}: {1}"**,** i**+**1**,** **this[**i**]);**

**}**

**}**

**public** **abstract** void printInfo**();**

**}**

**}**

Файл IVisitable.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Lab**.**\_02

**{**

**interface** IVisitable

**{**

void visit**();**

int getVisits**();**

**}**

**}**

Файл Museum.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Lab**.**\_02

**{**

class Museum**:** AMuseum**,** IVisitable

**{**

**private** string owner**;**

**public** string Owner

**{**

get **{** **return** owner**;** **}**

set **{** **if** **(**owner **!=** ""**)** owner **=** value**;** **}**

**}**

**private** int visits **=** 0**;**

**public** Museum**(**string name**,** string address**,** string owner**):** **base(**name**,** address**)** **{**

**this.**owner **=** owner**;**

**}**

**new** **public** void print**()**

**{**

Console**.**WriteLine**(**"Museum \"{0}\", {1}; Owner: {2}, Visits: {3}"**,** Name**,** Address**,** owner**,** visits**);**

**base.**printHalls**();**

**}**

**override** **public** void printInfo**()**

**{**

Console**.**WriteLine**(**"Owner: {0}, Visits: {1}"**,** owner**,** visits**);**

**}**

**public** void visit**()**

**{**

visits**++;**

**}**

**public** int getVisits**()**

**{**

**return** visits**;**

**}**

**}**

**}**

1. ТЕСТОВЫЙ ПРИМЕР

На рисунке 1 показана работа программы.

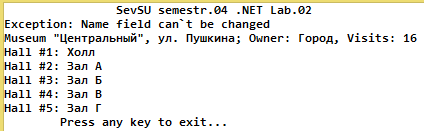


Рисунок 1 – работа программы

ВЫВОДЫ

В ходе данной лабораторной работы были изучены основы программирования пользовательских типов данных на языке C#. Освоена работа с наследованием, свойствами и индексаторами классов. Изучен механизм генерации и обработки исключений.