Лабораторная работа №2

“Исследование типов данных, определяемые пользователем. Наследование. Обработка исключений в C#”

1 ЦЕЛЬ РАБОТЫ

* 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 C# и применение интерфейсов при наследовании;
* изучить механизм генерации и обработки исключений.

2 ПОСТАНОВКА ЗАДАЧИ

1. 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.
2. Разработать интерфейсные классы, добавляющие некоторые ме­тоды в класс-потомок. Изучить причины возникновения коллизий имен при наследовании и способы их устранения.
3. Разработать классы исключительных ситуаций и применить их для обработки, возникающих исключений.
4. Написать демонстрационную программу.

Вариант 4

ЭКЗАМЕН: ФИО студента, дата, оценка, перечень вопросов.

3 ТЕКСТ ПРОГРАММЫ

Файл Program.cs

using System;

namespace ConsoleApp2

{

class Program

{

static void Main(string[] args)

{

Exam exam1 = new Exam("Долженко Илья Андреевич", "20.12.2019", 5, "Дрозин А.Ю.");

exam1[0] = "Что такое массив?";

exam1[1] = "Виды массивов";

exam1[2] = "Добавление элементов в массив";

exam1[3] = "Удаление элементов в массиве";

exam1.Print();

exam1.Visit(new Random().Next(25));

exam1.PrintInfo();

try

{

exam1.FullName = "Юра";

}

catch (ValueException ex)

{

Console.WriteLine($"Попытка изменить поле ФИО. Ошибка: {ex.Message}");

}

Console.WriteLine("---------------------------------------");

Exam exam2 = new Exam("Волобуев Юра Сергеевич", "22.12.2019", 4, "Дрозин А.Ю.");

exam2[0] = "Что такое HTML?";

exam2[1] = "Что такое CSS?";

exam2[2] = "Как добавить событие?";

exam2.Print();

exam2.Visit(new Random().Next(11));

exam2.PrintInfo();

Console.WriteLine("\nНажмите любую клавишу...");

Console.ReadKey();

}

}

}

Файл Exam.cs

using System;

namespace ConsoleApp2

{

class Exam : AExam, IVisitable

{

private int visits = 0;

private string professor;

public string Professor

{

get { return professor; }

set { if (professor != "") professor = value; }

}

public Exam(string fullName, string date, int mark, string professor) : base(fullName, date, mark)

{

this.professor = professor;

}

new public void Print()

{

Console.WriteLine($"Студент: {FullName} | Дата: {Date} | Оценка: {Mark}");

base.PrintQuestions();

}

public override void PrintInfo()

{

Console.WriteLine($"Преподователь: {professor} | Посещений: {visits}");

}

public void Visit(int visits)

{

this.visits = visits;

}

public int GetVisits()

{

return visits;

}

}

}

Файл AExam.cs

using System;

namespace ConsoleApp2

{

abstract class AExam

{

private string fullName;

public string FullName

{

get { return fullName; }

set { if (fullName == null) { fullName = value; } else { throw new ValueException(FieldType.Name); } }

}

private string date;

public string Date

{

get { return date; }

set { if (date == null) { date = value; } else { throw new ValueException(FieldType.Date); } }

}

public int Mark { get; }

public int QuestionsCount { get; private set; } = 0;

const int questionsMaxCount = 8;

private string[] questions = new string[questionsMaxCount];

public string this[int i]

{

get

{

if (i >= 0 && i <= QuestionsCount) { return questions[i]; }

else { throw new ValueException(FieldType.Questions); }

}

set

{

if (i == QuestionsCount) { questions[i] = value; QuestionsCount++; }

else if (i >= 0 && i <= QuestionsCount) { questions[i] = value; }

else { throw new ValueException(FieldType.Questions); }

}

}

public AExam(string fullName, string date, int mark)

{

this.fullName = fullName;

this.date = date;

this.Mark = mark;

}

public virtual void Print()

{

Console.WriteLine($"Студент: {fullName} | Дата: {date} | Оценка: {Mark}");

}

public void PrintQuestions()

{

for (int i = 0; i < QuestionsCount; i++)

{

Console.WriteLine($"Вопрос {i + 1}: {this[i]}");

}

}

public abstract void PrintInfo();

}

}

Файл ValueException.cs

using System;

namespace ConsoleApp2

{

class ValueException : Exception

{

new public string Message { get; set; }

public ValueException(FieldType fieldType)

{

switch (fieldType)

{

case FieldType.Name:

Message = "Поле ФИО не может быть изменено";

break;

case FieldType.Date:

Message = "Поле Дата не может быть пустым";

break;

case FieldType.Questions:

Message = "Поле Вопросы не корректное";

break;

default:

Message = "Неизвестное поле";

break;

}

}

}

enum FieldType

{

Name,

Date,

Questions

}

}

Файл IVisitable.cs

namespace ConsoleApp2

{

interface IVisitable

{

void Visit(int visits);

int GetVisits();

}

}

4 РЕЗУЛЬТАТЫ

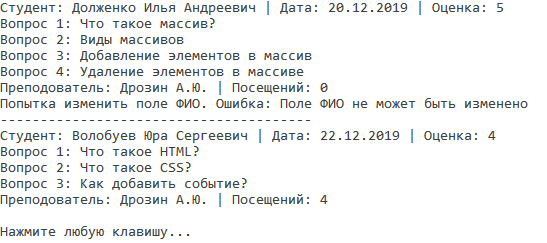


Рисунок 1 – Результат выполнения программы

ВЫВОДЫ

В процессе выполнения 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.