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 3

Печатное издание: название, ФИО автора, стоимость, оглавление.

3 ТЕКСТ ПРОГРАММЫ

Publish.cs

using System;

namespace ConsoleApp2

{

class Publish : Pub, ICircutable

{

private int copies = 0;

private string status;

public string Status

{

get

{

Random rand = new Random();

int n = rand.Next(100);

if (n < 50)

status = "Да";

else

status = "Нет";

return status;

}

set { status = value; }

}

public Publish(string article, string author, int price, string status) : base(article, author, price)

{

this.status = status;

}

new public void Print()

{

Console.WriteLine($"\nНазвание: {Article}\nИмя автора: {Author}\nСтоимость: {price}\n");

Contents();

}

public override void PrintСirculation()

{

Console.WriteLine($"Осталось: {copies} копий");

Console.WriteLine($"Доступно к продаже: {status}");

}

public void Circulation(int copies)

{

this.copies = copies;

}

public int GetCirc()

{

return copies;

}

}

}

Pub.cs

using System;

namespace ConsoleApp2

{

abstract class Pub

{

private string article;

private string author;

private string[] chapter = new string[5];

public string Article

{

get { return article; }

set { if (article == null) { article = value; } else { throw new ExeptionHandler(Field.Article); } }

}

public string Author

{

get { return author; }

set { if (author == null) { author = value; } else { throw new ExeptionHandler(Field.Author); } }

}

public int price { get; set; }

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

public string this[int i]

{

get

{

if (i >= 0 && i <= ChaptersCount) { return chapter[i]; }

else { throw new ExeptionHandler(Field.chapter); }

}

set

{

if (i == ChaptersCount) { chapter[i] = value; ChaptersCount++; }

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

else { throw new ExeptionHandler(Field.chapter); }

}

}

public Pub(string article, string author, int price)

{

this.article = article;

this.author = author;

this.price = price;

}

public virtual void Print()

{

Console.WriteLine($"\nКнига: {article}\nИмя автора: {author}\nСтоимость: {price}\n");

}

public void Contents()

{

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

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

{

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

}

}

public abstract void PrintСirculation();

}

}

ICircutable.cs

namespace ConsoleApp2

{

interface ICircutable

{

void Circulation(int copies);

int GetCirc();

}

}

ExceptionHandler.cs

using System;

namespace ConsoleApp2

{

class ExceptionHandler : Exception

{

public string warn { get; set; }

public ExeptionHandler(Field type)

{

switch (type)

{

case Field.Article:

warn = "Поле Название не может быть изменено";

break;

case Field.Author:

warn = "Поле Имя автора не может быть изменено";

break;

case Field.chapter:

warn = "Некооректные данные в поле Глава";

break;

default:

warn = "Некорректные данные";

break;

}

}

}

Файл Program.cs

using System;

namespace ConsoleApp2

{

class Program

{

static void Main(string[] args)

{

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

Publish book\_1 = new Publish("Поваренная книга анархиста", "Уильям Пауэлл", 1200, "Нет");

book\_1[0] = "Введение";

book\_1[1] = "История";

book\_1[2] = "Создание";

book\_1[3] = "Применение";

book\_1.Print();

book\_1.Circulation(new Random().Next(100));

book\_1.PrintСirculation();

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

Publish book\_2 = new Publish("Сделай сам", "Николай Степанович Амеба", 150, "Да");

book\_2[0] = "Введение";

book\_2[1] = "Сделай";

book\_2[2] = "Выброси";

book\_2.Print();

book\_2.Circulation(new Random().Next(100));

book\_2.PrintСirculation();

try

{

int b = 0;

book\_2.Article = "Сделай сам";

}

catch (ExceptionHandler exception)

{

Console.WriteLine($"\nПопытка изменить поле Название.\nОшибка: {exception.warn}");

}

Console.ReadKey();

}

}

}

4 РЕЗУЛЬТАТЫ

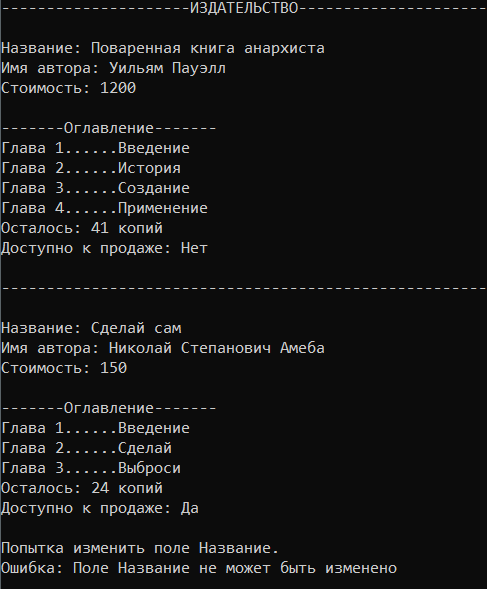


Рисунок 1 – Результат выполнения программы

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.