# Лабораторная работа № 21

**Тема:** Разработка программ на языке С# с использованием обработчиков исключений

1. Формирование навыков разработки программ на языке С# с использованием обработчиков исключительных ситуаций
2. Изучение методов работы с исключениями на языке С#.

. **Оборудование:**

1. ПК
2. Программное обеспечение: ОС Windows, среда Visual Studio.Net

**Контрольные вопросы:**

1) Как создается защищенный блок кода?

Для создания защищенного блока кода используется оператор try. Оператор содержит три части:

Контролируемый блок – составной оператор, начинающийся ключевым словом try. В контролируемый блок включаются потенциально опасные операторы программы. Все функции, прямо или косвенно вызываемые из блока, также считаются ему принадлежащими

Один или несколько обработчиков исключений – блоков catch, в которых описывается, как обрабатываются ошибки различных типов

Блок завершения finally, выполняемый независимо от того, возникла ли ошибка в контролируемом блоке

Синтаксис оператора try:

try {блок} [catch{блоки}] [finally{блок}]

2) Какие описать процедуру обработки исключения в C#?

Для описания процедур обработки исключений используются блоки обработчики исключений. Они начинаются с ключевого слова catch, за которым в скобках следует тип обрабатываемого исключения. Можно записать один или несколько обработчиков в соответствии с типами обрабатываемых исключений.

Блоки catch просматриваются в том порядке, в котором они записаны, пока не будет найден соответствующий типу выброшенного исключения

Существуют три формы записи обработчиков:

catch(тип имя параметра) {}

catch(тип) {}

catch {}

**Первая форма** применяется, когда имя параметра используется в теле обработчика для выполнения каких-либо действий

**Вторая форма** не предполагает использования информации об исключении, играет роль только его тип

**Третья форма** применяется для перехвата всех исключений. Так как обработчики просматриваются в том порядке, в котором они записаны, обработчик третьего типа (он может быть только один) следует помещать после всех остальных

3) Как генерируется исключение в C#?

Для генерации исключения используется оператор throw с параметром, определяющим вид исключения.

Синтаксис:

throw [выражения]

Параметр должен быть объектом, порожденным от стандартного класса System.Exception. Этот объект используется для передачи информации об исключении его обработчику.

Форма без параметра применяется только внутри блока catch для повторной генерации исключения. Тип выражения, стоящего после throw, определяет тип исключения.

4) Какие стандартные классы исключений имеются в C#?

К стандартным классам исключений относятся:

ArithmeticException – ошибка в арифметических операциях

DivideByZeroException – попытка деления на ноль

FormatException – попытка передать в метод аргумент неверного формата

IndexOutOfRangeException – индекс массива выходит за границы диапазона

InvalidCastException – ошибка преобразования типа

OutOfMemoryException – недостаточно памяти для создания нового объекта

OverFlowException – переполнение при выполнении арифметических операций

StackOverFlowException – переполнение стека

5) Какой класс является базовым для всех стандартных классов исключений в C#?

System.Exception является базовым общим классом для всех создаваемых исключений

**Ход работы:**

1. В соответствии с вариантом задания для лабораторной работы №21 Разработать алгоритм задачи и представить его в виде  программы на С#
2. Используя средства Visual Studio.Net создать файл с программой , выполнить тестирование и отладку программу.
3. Результаты представить в виде отчета
4. Сделать вывод о проделанной работе

**Задание на лабораторную работу**

**Вариант № 7**

**Смоделировать структуру музыкальной коллекции:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Коллекция | название (get, set)  фио владельца (get, set) |
| Музыкальный носитель (альбом) | автор/группа (get, set)  жанр (get, set)  год выпуска (get, set)  общая продолжительность звучания (get, set) |
| Музыкальное произведение | название (get, set)  продолжительность (get, set)  сформировать описание() |
| Песня | текст (get, set)  автор текста (get, set)  сформировать описание() |
| Инструментальное произведение | инструменты (get, set)  сформировать описание() |

а) Обработать все Exception с помощью блока try…catch(Exception …) в методе «сформировать описание» классов Песня и Инструментальное произведение. При возникновении Exception выводить на экран сообщение об ошибке.

б) Описать собственный класс Exception ProdolzhitelnostException. В методе «сформировать описание» классов **Песня и Инструментальное** произведение осуществить проверку свойства Продолжительность и выбрасывать собственное Exception типа ProdolzhitelnostException при значении свойства меньшем или равном нулю. В этом же методе обработать ProdolzhitelnostException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

В основной программе (main) проверить работу блока обработки Exception метода «сформировать описание».

**Листинг с исходным кодом**

using System;

using static System.Net.Mime.MediaTypeNames;

using System.Xml.Linq;

class Коллекция

{

public string название { get; set; }

public string фио\_владельца { get; set; }

public Коллекция(string \_название, string \_фио\_владельца)

{

название = \_название;

фио\_владельца = \_фио\_владельца;

}

}

class МузыкальныйНоситель

{

public string автор\_группа { get; set; }

public string жанр { get; set; }

public int год\_выпуска { get; set; }

public int общая\_продолжительность { get; set; }

public МузыкальныйНоситель(string \_автор\_группа, string \_жанр, int \_год\_выпуска, int \_общая\_продолжительность)

{

автор\_группа = \_автор\_группа;

жанр = \_жанр;

год\_выпуска = \_год\_выпуска;

общая\_продолжительность = \_общая\_продолжительность;

}

}

class МузыкальноеПроизведение

{

public string название { get; set; }

public int продолжительность { get; set; }

public МузыкальноеПроизведение(string \_название, int \_продолжительность)

{

название = \_название;

продолжительность = \_продолжительность;

}

public virtual void сформировать\_описание()

{

Console.WriteLine("Название: " + название);

Console.WriteLine("Продолжительность: " + продолжительность + " секунд");

}

}

class Песня : МузыкальноеПроизведение

{

public string текст { get; set; }

public string автор\_текста { get; set; }

public Песня(string \_название, int \_продолжительность, string \_текст, string \_автор\_текста)

: base(\_название, \_продолжительность)

{

текст = \_текст;

автор\_текста = \_автор\_текста;

}

public override void сформировать\_описание()

{

try

{

if (продолжительность<=0)

{

throw new ProdolzhitelnostException("Невозможно создать музыкальное произведение – не верная продолжительность");

}

Console.WriteLine("Название песни: " + название);

Console.WriteLine("Продолжительность: " + продолжительность + " секунд");

Console.WriteLine("Текст песни: " + текст);

Console.WriteLine("Автор текста: " + автор\_текста);

}

catch (ProdolzhitelnostException ex)

{

Console.WriteLine(ex.Message);

}

}

}

class ИнструментальноеПроизведение : МузыкальноеПроизведение

{

public string инструменты { get; set; }

public ИнструментальноеПроизведение(string \_название, int \_продолжительность, string \_инструменты)

: base(\_название, \_продолжительность)

{

инструменты = \_инструменты;

}

public override void сформировать\_описание()

{

try

{

if (продолжительность <= 0)// условие для исключения

{

throw new ProdolzhitelnostException("Невозможно создать музыкальное произведение – не верная продолжительность");// программируем собственное исключение

}

Console.WriteLine("Название инструментального произведения: " + название);

Console.WriteLine("Продолжительность: " + продолжительность + " секунд");

Console.WriteLine("Инструменты: " + инструменты);

}

catch (ProdolzhitelnostException ex)//Базовый для всех типов исключений Exception

{

Console.WriteLine(ex.Message);//Message: хранит сообщение об исключении, текст ошибки

}

}

}

class ProdolzhitelnostException : Exception//Базовый для всех типов исключений Exception

{

public ProdolzhitelnostException(string message) : base(message)

{

}

}

class Program

{

static void Main(string[] args)

{

try

{

Console.Write("Введите название песни: ");

string название\_песни = Console.ReadLine();

Console.Write("Введите продолжительность песни (в секундах): ");

int продолжительность\_песни = int.Parse(Console.ReadLine());

Console.Write("Введите текст песни: ");

string текст\_песни = Console.ReadLine();

Console.Write("Введите автора текста: ");

string автор\_текста = Console.ReadLine();

Песня песня = new Песня(название\_песни, продолжительность\_песни, текст\_песни, автор\_текста);

песня.сформировать\_описание();

}

catch (ProdolzhitelnostException ex)

{

Console.WriteLine("Произошла ошибка: " + ex.Message);//В данной строке ex представляет объект исключения

}

try

{

Console.Write("Введите название инструментального произведения: ");

string название\_произведения = Console.ReadLine();

Console.Write("Введите продолжительность инструментального произведения (в секундах): ");

int продолжительность\_произведения = int.Parse(Console.ReadLine());//используется для преобразования строки в целое число типа int.

Console.Write("Введите инструменты: ");

string инструменты = Console.ReadLine();

ИнструментальноеПроизведение произведение = new ИнструментальноеПроизведение(название\_произведения, продолжительность\_произведения, инструменты);

произведение.сформировать\_описание();

}

catch (ProdolzhitelnostException ex)

{

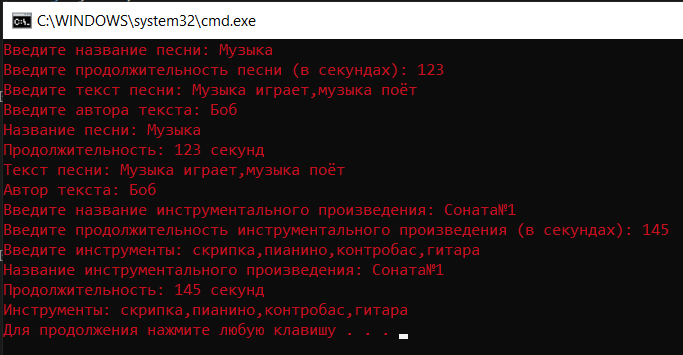
Console.WriteLine("Произошла ошибка: " + ex.Message);//В данной строке ex представляет объект исключения

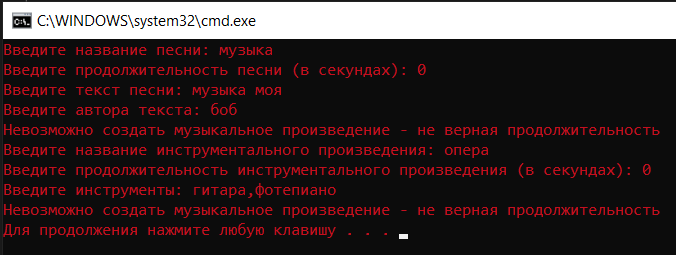
}

}

}

**Результат тестирования**

****

****

**Вывод по работе.** Код работает корректно