**Лабораторная работа №5**

**Обработка исключений**

Механизм исключений (Exception), предусмотренный в C#, упрощает обработку ошибок. Вместо того чтобы проверять значение, возвращаемое функциями и методами, вы можете использовать для обнаружения и обработки ошибок структурные операторы, такие как try и catch.

Методы стандартных библиотек классов возбуждают Exception при возникновении ошибочных ситуаций. Заключив "ненадежный" с точки зрения возникновения ошибок код в блок try, вы можете перехватить и обработать Exception в блоке catch.

Заметим, что Exception возникают и в таких распространенных ситуациях, как выход за пределы массива или строки в процессе индексации, деление на нуль, попытка использования ссылки со значением null, недопустимое преобразование классов и так далее. С помощью Exception все подобные ошибки нетрудно обнаружить и обработать во время выполнения приложения.

К механизму обработки Exception в следующие ключевые слова: try, catch, throw и finally. Схема работы этого механизма следующая. Вы пытаетесь (try) выполнить блок кода, и, если при этом возникает ошибка, система возбуждает (throw) Exception, которое в зависимости от его типа вы можете перехватить (catch) или передать обработчику по-умолчанию (finally).

Ниже приведена общая форма блока обработки Exception.

try {

// блок кода }

catch (ТипException1) {

// обработчик Exception типа ТипException1 }

catch (ТипException2) {

// обработчик Exception типа ТипException2

throw(e) // повторное возбуждение Exception }

finally {

}

Пример:

try {

line = Сonsole.КeadLine();

if (line.Length() == 0)

throw new EmptyLineException("Строка, считанная с консоли, пустая!");

console.printLine("Привет, %s!" % line);

}

catch (EmptyLineException exception) {

console.printLine("Привет!");

}

catch (Exception exception) {

console.printLine("Ошибка: " + exception.message());

}

else {

console.printLine("Программа выполнилась без исключительных ситуаций");

}

finally {

console.printLine("Программа завершается");

}

Операторы try можно вкладывать друг в друга аналогично тому, как можно создавать вложенные области видимости переменных. Если у оператора try низкого уровня нет раздела catch, соответствующего возбужденному Exception будут проверены разделы catch внешнего оператора try. Вот пример, в котором два оператора try вложены друг в друга посредством вызова метода.

class MultiNest {

static void procedure() {

try {

int c[] = { 1 };

c[42] = 99;

}

catch(IndexOutOfRangeException e) {

System.out.println("array index oob: " + e);

} }

public static void main(String args[]) {

try {

int a = args.length();

System.out.println("a = " + a);

int b = 42 / a;

procedure();

}

catch (ArithmeticException e) {

System.out.println("div by 0: " + e);

}

} }

Хотя встроенные Exception обрабатывают большинство частых ошибок, вероятно, вам потребуется создать ваши собственные типы Exception для обработки ситуаций, специфичных для ваших приложений. Это достаточно просто сделать: просто определите подкласс Exception.

В следующем примере объявляется новый подкласс Exception, который затем используется для сигнализации об ошибочной ситуации в методе. Он переопределяет метод toString (), позволяя отобразить тщательно настроенное описание Exception.

Пример:

class MyException extends Exception {

private int detail;

MyException(int a) {

detail = a;

}

public String toString() {

return "MyException!" + detail ();"]";

}

}

class ExceptionDemo {

static void compute(int a)

{

System.out.println("Вызван compute(" + a + ")");

if(a > 10)

throw new MyException(a);

System.out.println("Нормальное завершение");

}

public static void main(String args[]) {

try {

compute(1);

compute(20);

}

catch (MyException e) {

System.out.println("Перехвачено " + e) ;

}

}

}

В этом примере определен подкласс Exception с именем MyException. Этот подкласс достаточно прост: он имеет только конструктор и перегруженный метод toString. Класс ExceptionDemo определяет метод под названием compute (), который возбуждает объект MyException. Это Exception возбуждается, когда целочисленный параметр compute () принимает значение больше 10.

Метод main () устанавливает обработчик Exception MyException, затем вызывает compute () с правильным параметром (меньше 10), и с неправильным, чтобы продемонстрировать оба пути выполнения кода. Ниже показан результат.

Вызван compute(1)

Нормальное завершение

Вызван compute(20)

Перехвачено MyException[20]

Выше были рассмотрены способы обработки Exception и передача Exception в вызывающий метод. Однако имеется возможность совмещения этих подходов, т.е. Exception может быть обработано и выброшено в вызывающий метод:

public String сформироватьОписание() {

try {

...

if (nazvanie = = null || nazvanie.equals(“”)) throw new MyException (“Не указано название дерева”);

...

} catch (MyException ex) {

System.out.println(ex.getMessage());

throw ex; // повторный выброс Exception (операция return отсутствует)

}

}

Повторное создание Exception

Throw можно также использовать в блоке catch для повторного создания Exception, обрабатываемого в блоке catch. В этом случае оператор throw не принимает операнд Exception. Это наиболее полезно, когда метод передает аргумент от вызывающего объекта в другой метод библиотеки, а метод библиотеки создает Exception, которое должно быть передано вызывающему объекту. Например, в следующем примере повторно создается Exception NullReferenceException, возникающее при попытке получить первый символ неинициализированной строки.

Повторное генерирование Exception

Exception, перехваченное в одном блоке catch, может быть повторно сгенерировано в другом блоке, чтобы быть перехваченным во внешнем блоке catch. Наиболее вероятной причиной для повторного генерирования Exception служит предоставление доступа к Exception нескольким обработчикам. Допустим, что один обработчик оперирует каким-нибудь одним аспектом Exception, а другой обработчик — другим его аспектом. Для повторного генерирования Exception достаточно указать оператор throw без сопутствующего выражения, как в приведенной ниже форме:

throw;

Не следует, однако, забывать, что когда Exception генерируется повторно, то оно не перехватывается снова тем же самым блоком catch, а передается во внешний блок catch. Пример:

using System;

namespace ConsoleApplication1

{

class Program

{

static void Del(int x, int y)

{

try

{

int result = x / y;

}

catch(DivideByZeroException)

{

Console.WriteLine("Деление на ноль!");

throw;

}

}

static void Main()

{

try

{

Del(5, 0);

}

catch (DivideByZeroException)

{

Console.WriteLine("Программная ошибка!");

}

Console.ReadLine();

}

}

}

**Варианты заданий**

Вариант № 1

**Смоделировать структуру предприятия:**

|  |  |
| --- | --- |
| **Классы** | **Свойства и методы** |
| Фирма | название (get, set) |
| Отдел | название (get, set)  количество сотрудников (get, set) |
| Сотрудник | фио (get, set)  должность (get, set)  оклад (get, set)  рассчитать зарплату() |
| Штатный сотрудник | премия (get, set)  рассчитать зарплату() |
| Сотрудник по контракту | рассчитать зарплату() |

а) Обработать все Exception с помощью блока try…catch(Exception …) в методе «рассчитать зарплату» классов Штатный сотрудник и Сотрудник по контракту. При возникновении Exception выводить на экран сообщение об ошибке.

Описать собственный класс Exception PremiyaException. В методе «рассчитать зарплату» класса Штатный сотрудник выбрасывать собственное Exception типа PremiyaException при отрицательном значении свойства Премия. В этом же методе обработать PremiyaException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

В основной программе проверить работу блока обработки Exception метода «рассчитать зарплату».

б) Описать собственный класс Exception OkladException. В конструкторе реализовать проверку значения оклада, при отрицательном значении выбрасывать собственное Exception типа OkladException. Обработать Exception OkladException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать сотрудника – указан отрицательный оклад: <оклад> » и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Сотрудник и проверить работу обработчика Exception.

Вариант № 2

**Смоделировать структуру банка:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Банк | название (get, set) |
| Филиал | название (get, set)  общая сумма вкладов (get, set) |
| Вклад | фио вкладчика (get, set)  сумма вклада (get, set)  рассчитать сумму вклада (количество месяцев) |
| Долгосрочный вклад | рассчитать сумму вклада (количество месяцев) |
| Вклад до востребования | рассчитать сумму вклада (количество месяцев) |

а) Обработать все Exception с помощью блока try…catch(Exception …) в методе «рассчитать сумму вклада (количество месяцев)» классов Долгосрочный вклад и Вклад до востребования. При возникновении Exception выводить на экран сообщение об.

Описать собственный класс Exception KolichestvoException. В методе «рассчитать сумму вклада (количество месяцев)» классов Долгосрочный вклад и Вклад до востребования осуществить проверку значения параметра Количество месяцев - выбрасывать собственное Exception типа KolichestvoException при отрицательном значении. В этом же методе обработать KolichestvoException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

В основной программе проверить работу блока обработки Exception метода «рассчитать сумму вклада (количество месяцев)».

б) Описать собственный класс Exception VkladException. В конструкторе реализовать проверку значения суммы вклада, при отрицательном значении выбрасывать собственное Exception типа VkladException. Обработать Exception VkladException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать вклад – указана отрицательная сумма вклада: <сумма вклада> » и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Вклад и проверить работу обработчика Exception.

Вариант № 3

**Смоделировать структуру аэропорта:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Аэропорт | название (get, set) |
| Летательный аппарат | название (get, set)  макс. количество пассажиров (get, set)  взлет() |
| Самолет | длина полосы разгона (get, set)  взлет() |
| Вертолет | взлет() |
| Пассажир | фио (get, set)  №посадочного места (get, set) |

а) Обработать все Exception с помощью блока try…catch(Exception …) в методе «взлет» классов Самолет Вертолет. При возникновении Exception выводить на экран сообщение об ошибке.

Описать собственный класс Exception KolichestvoException. В методе «добавить пассажира» класса Летательный аппарат выбрасывать собственное Exception типа KolichestvoException при превышении максимального количества пассажиров (сравнивать текущее количество пассажиров и значение свойства «макс. количество пассажиров»). В этом же методе обработать KolichestvoException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

В основной программе (main) проверить работу блока обработки Exception методов «взлет» и «добавить пассажира».

б) Описать собственный класс Exception PolosaRazgonaException. В конструкторе реализовать проверку значения длины полосы разгона, при отрицательном значении или нулевом значении выбрасывать собственное Exception типа PolosaRazgonaException. Обработать Exception PolosaRazgonaException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать самолет – указана некорректная длина полосы разгона: <длина полосы разгона> » и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Самолет и проверить работу обработчика Exception.

Вариант № 4

**Смоделировать структуру библиотеки:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Библиотека | название (get, set) |
| Отдел (по жанрам) | название жанра (get, set)  количество изданий (get, set) |
| Издание | название (get, set)  автор (get, set)  год издания (get, set)  сформировать описание() |
| Книга | резюме (get, set)  сформировать описание() |
| Журнал | статьи (get, add)  сформировать описание() |

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

Описать собственный класс Exception BookException. В методе «сформировать описание» классов Книга и Журнал осуществить проверку значений свойств Название и Автор – выбрасывать собственное Exception типа BookException при значении null или пустой строке (“”).. В этом же методе обработать BookException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

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

б) Описать собственный класс Exception GodIzdaniyaException. В конструкторе реализовать проверку значения года издания, при отрицательном или нулевом значении выбрасывать собственное Exception типа GodIzdaniyaException. Обработать Exception GodIzdaniyaException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать издание – указан некорректный год издания: <год издания>» и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Издания и проверить работу обработчика Exception.

Вариант № 5

**Смоделировать структуру компании сотовой связи:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Компания | Название (get, set) |
| Тариф | название (get, set)  количество абонентов (get, set)  рассчитать стоимость разговора (количество секунд) |
| Посекундный тариф | стоимость секунды разговора» (get, set)  рассчитать стоимость разговора (количество секунд) |
| Поминутный тариф | стоимость минуты разговора» (get, set)  рассчитать стоимость разговора (количество секунд) |
| Абонент | фио (get, set)  номер телефона (get, set)  остаток на счете (get, set) |

а) Обработать все Exception с помощью блока try…catch(Exception …) в методе «рассчитать стоимость разговора (количество секунд)» классов Посекундный тариф и Поминутный тариф. При возникновении Exception выводить на экран сообщение об ошибке.

Описать собственный класс Exception KolichestvoSekundException. В методе «рассчитать стоимость разговора (количество секунд)» классов Посекундный тариф и Поминутный тариф осуществить проверку значения параметра Количество секунд – выбрасывать собственное Exception типа KolichestvoSekundException при отрицательном значении. В этом же методе обработать KolichestvoSekundException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

В основной программе (main) проверить работу блока обработки Exception метода «рассчитать стоимость разговора (количество секунд)».

б) Описать собственный класс Exception NazvanieException. В конструкторе реализовать проверку значения свойства Название, при значении null или пустой строке выбрасывать собственное Exception типа NazvanieException. Обработать Exception NazvanieException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать тариф – не указано название» и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Тариф и проверить работу обработчика Exception.

Вариант № 6

**Смоделировать структуру автосалона:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Автосалон | название (get, set) |
| Автомобиль | марка (get, set)  макс. количество пассажиров (get, set)  стоимость (get, set)  количество на складе(get, set)  boolean наличие (get, set) |
| Заявка на покупку | фио покупателя (get, set)  номер телефона (get, set)  автомобили (add, remove, get)  рассчитать стоимость заказа() |
| Заявка на приобретение со стенда | рассчитать стоимость заказа() |
| Заявка на отложенную поставку | процент скидки (get, set)  рассчитать стоимость заказа() |

а) Обработать все Exception с помощью блока try…catch(Exception …) в методе «рассчитать стоимость заказа» классов Заявка на приобретение со стенда и Заявка на отложенную поставку. При возникновении Exception выводить на экран сообщение об ошибке.

Описать собственный класс Exception KolichestvoAvtomobileyException. В методе «рассчитать стоимость заказа» классов Заявка на приобретение со стенда и Заявка на отложенную поставку выбрасывать собственное Exception типа KolichestvoAvtomobileyException, если количество автомобилей в заявке равно нулю. В этом же методе обработать KolichestvoAvtomobileyException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

В основной программе (main) проверить работу блока обработки Exception метода «рассчитать стоимость заказа».

б) Описать собственный класс Exception FIOException. В конструкторе реализовать проверку значения свойства ФИО покупателя, при значении null или пустой строке выбрасывать собственное Exception типа FIOException. Обработать Exception FIOException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать заявку – не указано ФИО покупателя» и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Заявка на покупку и проверить работу обработчика Exception.

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.

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».

б) Описать собственный класс Exception NazvanieException. В конструкторе реализовать проверку значения свойства Название, при значении null или пустой строке выбрасывать собственное Exception типа NazvanieException. Обработать Exception NazvanieException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать музыкальное произведение – не указано название» и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Музыкальное произведение и проверить работу обработчика Exception.

Вариант № 8

**Смоделировать структуру реестра городского жилья:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Город | название (get, set) |
| Здание | название улицы (get, set)  номер дома (get, set)  общая площадь (get)  базовая ежемесячная оплата за кв.м площади (get, set) |
| Помещение | номер (get, set)  площадь (get, set)  рассчитать ежемес. оплату() |
| Квартира | ФИО жильцов (get, add, remove)  рассчитать ежемес. оплату() |
| Офис | название фирмы-владельца (get, set)  вид деятельности (get, set)  рассчитать ежемес. оплату() |

а) Обработать все Exception с помощью блока try…catch(Exception …) в методе «рассчитать ежемес. оплату» классов Квартира и Офис. При возникновении Exception выводить на экран сообщение об ошибке.

Описать собственный класс Exception KolichestvoException. В методе «рассчитать ежемес. оплату» класса Квартира осуществить проверку свойства «ФИО жильцов» и выбрасывать собственное Exception типа KolichestvoException при отсутствии жильцов в квартире (кол-во эл-тов коллекции равно нулю). В этом же методе обработать KolichestvoException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

В основной программе (main) проверить работу блока обработки Exception метода «рассчитать ежемес. оплату».

б) Описать собственный класс Exception PloschadException. В конструкторе реализовать проверку значения свойства Площадь, при значении 0 или отрицательном значении выбрасывать собственное Exception типа PloschadException. Обработать Exception PloschadException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать помещение – указано некорректное значение площади: <площадь>» и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Помещение и проверить работу обработчика Exception.

Вариант № 9

**Смоделировать структуру зоопарка:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Зоопарк | название (get, set) |
| Вольер/клетка | номер (get, set)  размер (get, set)  макс. количество животных (get, set)  текущее количество животных (get, set) |
| Животное | название (get, set)  boolean хищник (get, set)  сформировать описание() |
| Рыба | глубоководная (get, set)  сформировать описание() |
| Птица | скорость полета (get, set)  сформировать описание() |
| Зверь | среда обитания (get, set)  сформировать описание() |

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

Описать собственный класс Exception HischnikException. В методе «добавить животное» в классе Клетка при попытке добавить хищника в клетку, где уже содержатся не хищные животные выбрасывать собственное Exception типа HischnikException. В этом же методе обработать HischnikException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

В основной программе (main) проверить работу блока обработки Exception методов «сформировать описание» классов Рыба, Птица и Зверь и метода «добавить животное» класса Клетка.

б) Описать собственный класс Exception NazvanieException. В конструкторе реализовать проверку значения свойства Название, при значении null или пустой строке выбрасывать собственное Exception типа NazvanieException. Обработать Exception NazvanieException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать животное – не указано название» и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Животное и проверить работу обработчика Exception.

Вариант № 10

**Смоделировать структуру автоматизированного банкомата:**

|  |  |
| --- | --- |
| **Классы** | **Свойства** |
| Банк | название (get, set) |
| Счет | номер (get, set)  PIN-код (get, set)  остаток (get, set)  снять со счета (сумма) |
| Обычный счет | снять со счета (сумма) |
| Льготный счет | снять со счета (сумма) |
| Банкомат | идентификационный номер (get, set)  адрес (get, set) |

а) Обработать все Exception с помощью блока try…catch(Exception …)в методе «снять со счета (сумма)» классов Обычный счет и Льготный счет. При возникновении Exception выводить на экран сообщение об ошибке.

Описать собственный класс Exception SnyatSoSchetaException. В методе «снять со счета (сумма)» классов Обычный счет и Льготный счет осуществить проверку значения параметра «сумма» и выбрасывать собственное Exception типа SnyatSoSchetaException при отрицательном значении, а также при попытке снять со счета сумму большую, чем остаток на счете. В этом же методе обработать SnyatSoSchetaException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран.

В основной программе (main) проверить работу блока обработки Exception метода «снять со счета (сумма)».

б) Описать собственный класс Exception OstatokNaScheteException. В конструкторе реализовать проверку значения остатка на счете, при отрицательном значении выбрасывать собственное Exception типа OstatokNaScheteException. Обработать Exception OstatokNaScheteException с помощью блока try…catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать счет – указано некорректное значение остатка на счете: <остаток на счете>» и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса Счет и проверить работу обработчика Exception.

Приложение 1

Справочные материалы

Исключения и обработка Исключений

<https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/exceptions/>

Операторы обработки Исключений

<https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/exception-handling-statements>

Создание классов Исключений

<https://metanit.com/sharp/tutorial/3.17.php>

throw и повторное создание Исключений

<https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/throw>