Лабораторная работа 20  
Вариант 16

Класс Book

using System;

namespace Lab20\_HW

{

public class Book

{

private string \_name;

private string \_authorName;

private string \_authorSurname;

private int \_pageNumber;

private string \_coverType;

private decimal \_price;

public Book() { }

public Book(string name = "книга не указана", string authorName = "имя автора не указано", string authorSurname = "фамилия автора не указана", int numberOfPages = 0, string coverType = "тип обложки не указан", decimal price = 0)

{

Name = name;

AuthorName = authorName;

AuthorSurname = authorSurname;

PageNumber = numberOfPages;

CoverType = coverType;

Price = price;

}

public string Name

{

get => \_name;

set

{

\_name = Validator.ValidateVarchar(value, 45) ?

value : throw new MyException2();

}

}

public string AuthorName

{

get => \_authorName;

set => \_authorName = Validator.ValidateVarchar(value, 45) ?

value : throw new NumberException(value);

}

public string AuthorSurname

{

get => \_authorSurname;

set => \_authorSurname = Validator.ValidateVarchar(value, 45) ?

value : throw new NumberException(value);

}

public int PageNumber

{

get => \_pageNumber;

set => \_pageNumber = Validator.ValidateNumber(value) ?

value : throw new NumberException(value.ToString());

}

public string CoverType

{

get => \_coverType;

set => \_coverType = Validator.ValidateCover(value) ?

value : throw new NumberException(value);

}

public decimal Price

{

get => \_price;

set => \_price = Validator.ValidatePrice(value) ?

Math.Round(value, 2) : throw new Exception("Цена книги не может быть отрицательной ");

}

public override string ToString()

{

return $"книга: {\_name} автор: {\_authorName} {\_authorSurname} количество страниц {\_pageNumber} тип обложки: {\_coverType} стоимость: {\_price}руб";

}

}

}

Класс ошибки MyException

using System;

namespace Lab20\_HW

{

internal class NumberException: Exception

{

public NumberException(string exceptionData) : base()

{

if (!string.IsNullOrEmpty(exceptionData)) Console.WriteLine($"ваша ошибка в {exceptionData}");

else Console.WriteLine("введенная строка пуста");

}

}

}

Класс ошибки MyException2

using System;

namespace Lab20\_HW

{

internal class MyException2: Exception

{

public MyException2() : base()

{

Console.WriteLine("не удалось создать объект из-за некорректных данных");

}

}

}

Класс program

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace Lab20\_HW

{

internal class Program

{

static void Main(string[] args)

{

List<Book> books = new List<Book>

{

//new Book(),

//new Book(),

//new Book(),

//new Book(),

//new Book(),

//new Book("", "Агата", "Кристи", 256, "мягкая", 256),

//new Book(),

//new Book(),

//new Book(),

//new Book(),

};

AdToList(ref books, "", "Агата", "Кристи", 256.ToString(), "мягкая", 256.ToString());

AdToList(ref books, "Война и мир", "Лев", "Толстой", 600.ToString(), "мягкая", 1225.ToString());

AdToList(ref books, "Война и мир", "Лев", "Толстой", 671.ToString(), "мягкая", 671.ToString());

AdToList(ref books, "1984", "Джордж", "Оруэлл", 328.ToString(), "твердая", 328.ToString());

AdToList(ref books, "Гарри Поттер и философский камень", "Джоан", "Роулинг", 223.ToString(), "мягкая", 10.ToString());

AdToList(ref books, "", "Михаил", "Булгаков", 1000.ToString(), "твердая", 384.ToString());

AdToList(ref books, "Алиса в Стране чудес", "Льюис", "Кэрролл", 96.ToString(), "мягкая", 96.ToString());

AdToList(ref books, "Маленький принц", "Антуан", "Де Сент-Экзюпери", 91.ToString(), "твердая", 91.ToString().ToString());

AdToList(ref books, "Триумфальная арка", "Эрих Мария", "Ремарк", 632.ToString(), "твердая", 632.ToString());

AdToList(ref books, "Гордость и предубеждение", "Джейн", "Остин", 1000.ToString(), "мягкая", 432.ToString());

var newBooksList1 = (from book in books

orderby book.Name, book.Price

select book).ToList();

Console.WriteLine("отсортированный список по 2 критериям");

foreach (Book book in newBooksList1)

Console.WriteLine(book);

Console.WriteLine();

Console.WriteLine();

var newBooksList2 = books.Where(book => book.CoverType == "мягкая");

Console.WriteLine("список книг в мягкой обложкой");

foreach (Book book in newBooksList2)

Console.WriteLine(book);

Console.WriteLine();

Console.WriteLine();

var newBooksList3 = books.Where(book => book.CoverType == "твердая");

Console.WriteLine("список книг в твердой обложкой");

foreach (Book book in newBooksList3)

Console.WriteLine(book);

Console.WriteLine();

Console.WriteLine();

var newBooksList4 = books.Where(book => book.CoverType == "мягкая" && book.Price < 100);

Console.WriteLine("список книг в мягкой обложкой со стоимостью ниже 100руб");

foreach (Book book in newBooksList4)

Console.WriteLine(book);

Console.WriteLine();

Console.WriteLine();

int MaxNumberOfPage = books.Max(book => book.PageNumber);

var newBooksList5 = from book in books where book.PageNumber == MaxNumberOfPage select book;

Console.WriteLine("список книг с наибольшим количеством страниц ");

foreach (Book book in newBooksList5)

Console.WriteLine(book);

Console.ReadKey();

}

public static void AdToList(ref List<Book> books, params string[] args)

{

bool create = true;

Book newbook = null;

try

{

newbook = new Book(args[0], args[1], args[2], int.Parse(args[3]), args[4], decimal.Parse(args[5]));

}

catch (MyException2)

{

create = false;

}

if (create) books.Add(newbook);

}

}

}

Класс для валидации данных Validator

using System;

namespace Lab20\_HW

{

public static class Validator

{

public static bool ValidateVarchar(string value, int maxLength)

{

if (string.IsNullOrEmpty(value))

{

Console.WriteLine("1");

return false;

}

if (string.IsNullOrWhiteSpace(value))

{

Console.WriteLine("2");

return false;

}

if (value.Length > maxLength)

{

Console.WriteLine("3");

return false;

}

if (char.IsUpper(value[0]) && char.IsDigit(value[0]))

{

Console.WriteLine("4");

return false;

}

return true;

}

public static bool ValidateNumber(int value)

=> (value > 0);

public static bool ValidateCover(string value)

{

if (value != "мягкая" && value != "твердая")

return false;

return true;

}

public static bool ValidatePrice(decimal value)

{

if (value <= 0)

return false;

return true;

}

}

}