**ПРИЛОЖЕНИЕ 11**

Практическая работа №12

Задание 2

2.1

namespace TemplateM

{

//Создадим необобщённый класс

class InfoObject

{

//Обобщённый метод использующий также ограничение

public static string Info<T>(T obj)

where T : Biblioteka

{

return obj.ToString();

}

}

class Biblioteka

{

public Biblioteka(int reg\_nomer, string avtor, string nazvanie, int got\_izdania, string izdatelstvo, int kol\_str)

{

this.reg\_nomer = reg\_nomer;

this.avtor = avtor;

this.nazvanie = nazvanie;

this.got\_izdania= got\_izdania;

this.izdatelstvo=izdatelstvo;

this.kol\_str=kol\_str;

}

public Biblioteka()

{

Console.WriteLine("Введите регестрационный номер: ");

this.reg\_nomer = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите автора: ");

this.avtor = Console.ReadLine();

Console.WriteLine("Введите название: ");

this.nazvanie = Console.ReadLine();

Console.WriteLine("Введите год издания:");

this.got\_izdania = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите издательство:");

this.izdatelstvo = Console.ReadLine();

Console.WriteLine("Введите количество страниц:");

this.kol\_str = Convert.ToInt32(Console.ReadLine());

}

public int reg\_nomer { get; set; }

public string avtor { get; set; }

public string nazvanie { get; set; }

public int got\_izdania { get; set; }

public string izdatelstvo { get; set; }

public int kol\_str { get; set; }

}

class BibliotekaIncome : Biblioteka

{

public BibliotekaIncome(int reg\_nomer, string avtor, string nazvanie, int got\_izdania, string izdatelstvo, int kol\_str,string Income)

: base(reg\_nomer, avtor, nazvanie, got\_izdania,izdatelstvo, kol\_str)

{

IncomePFM = Income;

}

public BibliotekaIncome()

: base()

{

Console.WriteLine("Введите доход студента: ");

IncomePFM = Console.ReadLine();

}

public string IncomePFM { get; set; }

public override string ToString()

{

return String.Format(@"Список студентов :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Регестрационный номер: {0}

Автор: {1}

Название: {2}

Год издания: {3}

Издательство: {4}

Количество страниц: {5}

Доход студента: {6}", reg\_nomer, avtor, nazvanie, got\_izdania, izdatelstvo, kol\_str, IncomePFM);

}

}

class Program

{

static void Main()

{

BibliotekaIncome[] student1 = new BibliotekaIncome[3];

for (int i = 0; i < 3; i++)

{

student1[i] = new BibliotekaIncome();

}

//Вызываем обобщённый метод

string s;

for (int i = 0; i < 2; i++)

{

s = InfoObject.Info<BibliotekaIncome>(student1[i]);

Console.WriteLine(s);

}

Console.ReadLine();

2.2

namespace TemplateK

{

class PhoneInfo

{

public PhoneInfo(string ФИО, string адрес, int номер ,int время\_внутр\_разговоров)

{

this.ФИО = ФИО;

this.адрес = адрес;

this.номер = номер;

this.время\_внутр\_разговоров = время\_внутр\_разговоров;

}

public PhoneInfo()

{

Console.WriteLine("Введите ФИО");

this.ФИО = Console.ReadLine();

Console.WriteLine("Введите адрес");

this.адрес = Console.ReadLine();

Console.WriteLine("Введите номер");

this.номер = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите время внутригородских разговоров");

this.время\_внутр\_разговоров = Convert.ToInt32(Console.ReadLine());

}

public string ФИО { get; set; }

public string адрес { get; set; }

public int номер { get; set; }

public int время\_внутр\_разговоров{ get; set; }

}

//Создадим класс, унаследованный от UserInfo

class AllInfoPhone : PhoneInfo

{

public AllInfoPhone(string ФИО ,string адрес, int номер , int время\_внутр\_разговоров, int время\_между\_разговоров)

: base(ФИО, адрес, номер, время\_внутр\_разговоров)

{

this.время\_между\_разговоров = время\_между\_разговоров;

}

public AllInfoPhone()

: base()

{

Console.WriteLine("Введите время междугородских разговоров ");

this. время\_между\_разговоров = Convert.ToInt32(Console.ReadLine());

}

public int время\_между\_разговоров { get; set; }

public override string ToString()

{

string s = String.Format("Информация о телефоне: \n{0} {1} {2} {3} {4}\n",

this.ФИО, this.адрес, this.номер, this.время\_внутр\_разговоров, время\_между\_разговоров);

return s;

}

}

//Обобщённый класс, использующий ограничение на базовый класс

class Info<T> where T : PhoneInfo

{

T[] PhoneList;

int i;

public Info()

{

PhoneList = new T[2];

i = 0;

}

public void Add(T obj)

{

if (i == 2) return;

PhoneList[i] = obj;

i++;

return;

}

public void ReWrite()

{

foreach (T t in PhoneList)

Console.WriteLine(t.ToString());

}

}

class Program

{

static void Main()

{

Info<AllInfoPhone> database1 = new Info<AllInfoPhone>();

Console.WriteLine("Используя ограничение цикла for ");

for (int i = 0; i < 2; i++)

database1.Add(new AllInfoPhone());

database1.ReWrite();

Info<AllInfoPhone> database2 = new Info<AllInfoPhone>();

Console.WriteLine("Используя ограничение обобщённого класса ");

database2.Add(new AllInfoPhone(ФИО: "Фалевич Алексей Викторович", адрес: Жупраны, номер:6358963, время\_внутр\_разговоров: "8", время\_между\_разговоров: "9"));

database2.Add(new AllInfoPhone(ФИО: "Еваровский Андрей Ромуальдович", адрес: Ошмяны, номер: 6897512, время\_внутр\_разговоров: "6", время\_между\_разговоров: "10"));

//Ограничение обобщённого класса не даст внести больше обхектов, чем нужно

database2.Add(new AllInfoPhone(ФИО: "Авгуль Владислав Павлович", адрес: Жупраны, номер:5631233, время\_внутр\_разговоров: "2", время\_между\_разговоров: "13"));

database2.ReWrite();

Console.ReadLine();

2.3

namespace TemplateI

{

//Используем обобщённый интерфейс IComparable

class Abityrient : IComparable<object>

{

public string Фамилия;

public string Имя;

public string Отчество;

public string Адрес;

public int Оценка;

byte sort;//Переменная, принимающая возможные значения типа сортировки

public Abityrient(string Фамилия, string Имя, string Отчество, string Адрес, int Оценка)

{

this.Фамилия = Фамилия;

this.Имя = Имя;

this.Отчество = Отчество;

this.Адрес = Адрес;

this.Оценка = Оценка;

}

public Abityrient()

{

Console.WriteLine("Введите фамилию: ");

this.Фамилия = Console.ReadLine();

Console.WriteLine("Введите имя: ");

this.Имя = Console.ReadLine();

Console.WriteLine("Введите отчество: ");

this.Отчество = Console.ReadLine();

Console.WriteLine("Введите адрес: ");

this.Адрес = Console.ReadLine();

Console.WriteLine("Введите оценку: ");

this.Оценка = Convert.ToInt32(Console.ReadLine());

}

public Abityrient (int d)

{

this.Фамилия = "";

this.Имя = "";

this.Отчество = "";

this.Адрес = "";

this.Оценка = 0;

}

//Реализуем интерфейс IComparable<T>

public int CompareTo(object other)

{

//Проверим тип объекта obj

Abityrient db = other as Abityrient;

if (db != null)

{

switch (db.sort)

{

case 1:

{

if (String.Compare(this.Фамилия, db.Фамилия) > 0)

return 1;

if (String.Compare(this.Фамилия, db.Фамилия) < 0)

return -1;

else

return 0;

}

case 2:

{

if (String.Compare(this.Имя, db.Имя) > 0)

return 1;

if (String.Compare(this.Имя, db.Имя) < 0)

return -1;

else

return 0;

}

case 3:

{

if (String.Compare(this.Отчество, db.Отчество) > 0)

return 1;

if (String.Compare(this.Отчество, db.Отчество) < 0)

return -1;

else

return 0;

}

case 4:

{

if (String.Compare(this.Адрес, db.Адрес) > 0)

return 1;

if (String.Compare(this.Адрес, db.Адрес) < 0)

return -1;

else

return 0;

}

case 5:

{

if (this.Оценка> db.Оценка)

return 1;

if (this.Оценка < db.Оценка)

return -1;

else

return 0;

}

default:

throw new FormatException("Такой сортировки не существует.");

}

}

else

throw new FormatException("Данный параметр не относится к базе данных.");

}

//Метод, реализующий сортировку

public void Sort(ref Abityrient[] db\_object)

{

try

{

Array.Sort(db\_object);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

foreach (Abityrient d in db\_object)

Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}", d.Фамилия, d.Имя, d.Отчество, d.Адрес,d.Оценка);

}

public void NumberSort(byte j, ref Abityrient[] arr)

{

for (int i = 0; i < arr.Length; i++)

arr[i].sort = j;

}

}

class Program

{

static void Main()

{

Abityrient[] db\_arr = new Abityrient[3];

for (int i = 0; i < 3; i++)

{

db\_arr[i] = new Abityrient();

}

Console.WriteLine("Исходная таблица: \n-----------------\n\n" +

"Фамилия\tИмя\tОтчество\t Адрес\tОценка\n");

foreach (Abityrient d in db\_arr)

Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}", d.Фамилия, d.Имя, d.Отчество, d.Адрес, d.Оценка);

Console.WriteLine("\nОтсортированная таблица БД по фамилии: " +

"\n----------------------------------------------------------------------------------------------------\n" +

"Фамилия\tИмя\tОтчество\tАдрес\tОценка\n");

Abityrient sr1 = new Abityrient(0);

sr1.NumberSort(1, ref db\_arr);

sr1.Sort(ref db\_arr);

Console.WriteLine("\nОтсортированная таблица БД по имени: " +

"\n----------------------------------------------------------------------------------------------------\n" +

"Фамилия\tИмя\tОтчество\tАдрес\tОценка\n");

sr1.NumberSort(2, ref db\_arr);

sr1.Sort(ref db\_arr);

Console.WriteLine("\nОтсортированная таблица БД по отчеству: " +

"\n----------------------------------------------------------------------------------------------------\n" +

"Фамилия\tИмя\tОтчество\tАдрес\tОценка\n");

sr1.NumberSort(3, ref db\_arr);

sr1.Sort(ref db\_arr);

Console.WriteLine("\nОтсортированная таблица БД по адресу: " +

"\n----------------------------------------------------------------------------------------------------\n" +

"Фамилия\tИмя\tОтчество\tАдрес\tОценка\n");

sr1.NumberSort(4, ref db\_arr);

sr1.Sort(ref db\_arr);

Console.WriteLine("\nОтсортированная таблица БД по оценкам: " +

"\n----------------------------------------------------------------------------------------------------\n" +

"Фамилия\tИмя\tОтчество\tАдрес\tОценка\n");

sr1.NumberSort(5, ref db\_arr);

sr1.Sort(ref db\_arr);

Console.ReadLine();

}