**TAXI**

**Program.cs**

using System;

using System.Linq;

using System.IO;

using System.Data;

namespace ConsoleApp2

{

class Program

{

static void Main(string[] args)

{

Park<Taxi> uber = new Park<Taxi>();

Taxi taxi1 = new Taxi(1, new Location(111, 222, 60));

Taxi taxi3 = new Taxi(3, new Location(123, 223, 30));

Taxi taxi4 = new Taxi(4, new Location(145, 234, 50));

Taxi taxi5 = new Taxi(5, new Location(120, 215, 70));

uber.Add(taxi1);

uber.Add(taxi5);

uber.Add(taxi4);

uber.Add(taxi3);

Console.WriteLine("Введите значение x:");

int x = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите значение y:");

int y = Convert.ToInt32(Console.ReadLine());

foreach (var item in uber.List)

{

Console.WriteLine(Math.Sqrt(Math.Pow(item.loc.lat - x, 2) + Math.Pow(item.loc.longe - y, 2)));

}

uber.List = uber.GetList().OrderBy(t =>

Math.Sqrt(Math.Pow(t.loc.lat - x, 2) + Math.Pow(t.loc.longe - y, 2))).ToList();

foreach (var item in uber.List)

{

Console.WriteLine(Math.Sqrt(Math.Pow(item.loc.lat - x, 2) + Math.Pow(item.loc.longe - y, 2)));

}

using (StreamWriter sw = new StreamWriter("D:\\belstu\\2k1s\\OOP\\exam\\ConsoleApp2\\min.txt", false))

{

Taxi taxi = uber.List[0];

sw.WriteLine(taxi.ToString());

}

}

}

}

**Location.cs**

namespace ConsoleApp2

{

class Location

{

public int lat { get; set; }

public int longe { get; set; }

public int speed { get; set; }

public Location (int \_lat, int \_longe, int \_speed)

{

lat = \_lat;

longe = \_longe;

speed = \_speed;

}

}

}

**Park.cs**

namespace ConsoleApp2

{

class Park<T>

{

public List<T> List = new List<T>();

public void Add(T item)

{

List.Add(item);

}

public void Delete(T item)

{

List.Remove(item);

}

public void DeleteAll()

{

List.Clear();

}

public T? Find(Predicate<T> item)

{

return List.Find(item);

}

*public object Find(Predicate<T> predicate)*

*{*

*foreach (T item in list)*

*{*

*if (predicate(item))*

*{*

*return item;*

*}*

*}*

*}*

public List<T> GetList()

{

return List;

}

}

}

**Taxi.cs**

namespace ConsoleApp2

{

class Taxi

{

public int number { get; set; }

private Location Loc;

public Location loc

{

get { return Loc; }

set { Loc = value; }

}

enum Status

{

free,

busy,

}

private Status status = Status.free;

public void ChangeStatus()

{

if (status == Status.free)

{

status = Status.busy;

}

else

{

status = Status.free;

}

}

public Taxi(int \_number, Location \_loc)

{

number = \_number;

loc = \_loc;

}

}

}

**Addres**

**Program.cs**

using System;

namespace \_1\_2

{

internal static class Program

{

public static void Main()

{

Address a1 = new Address("Иванов Иван Иванович","email1");

Address a2 = new Address("Петров Пётр Пертровис", "email2");

Letter l1 = new Letter("Заголовок1", "Текст1", a1, a2, "Подпись1");

Letter l2 = new Letter("Заголовок2", "Текст2", a2, a1, "Подпись2");

Letter l3 = new Letter("Заголовок3", "Текст2", a1, a2, "Подпись2");

Console.WriteLine(l1.ToString());

Console.WriteLine(l2.ToString());

*IGetIt mew = new Letter("Email", "Hello", "lobanov@mail.com", "drozd@mail.com", "Vladimir");*

*mew.Print();*

*mew.Save(path, name1);*

l1.Print();

l1.Read();

Console.WriteLine(l1.Equals(l2));

Console.WriteLine(l2.Equals(l3));

Box b1 = new Box();

b1.Add(l1);

b1.Add(l2);

b1.Add(l3);

Box b2 = new Box();

b2.letters = b1.letters

.Where(l => l.addressTo.Email == "email2")

.ToList<Letter>();

Task[] tasks = new Task[3]

{

new Task(() => l1.Print()),

new Task(() => l2.Print()),

new Task(() => l3.Print())

};

foreach(Task t in tasks)

{

t.Start();

}

}

}

}

**Addres.cs**

namespace \_1\_2

{

internal class Address

{

public string Fio { get; set; }

public string Email { get; set; }

public Address(string fio, string email)

{

this.Fio = fio;

this.Email = email;

}

}

}

**Letter.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_1\_2

{

internal class Letter: IGetIt

{

public string Subject { get; set; }

public string Text { get; set; }

public Address addressTo { get; set; }

public Address addressFrom { get; set; }

public string Signature { get; set; }

public Letter(string subject, string text, Address addressTo, Address addressFrom, string signature)

{

Subject = subject;

Text = text;

this.addressTo = addressTo;

this.addressFrom = addressFrom;

Signature = signature;

}

public void Read()

{

Console.WriteLine(Text);

}

public void Print()

{

using (StreamWriter sw = new StreamWriter("C:\\Users\\looog\\source\\repos\\\_1\_2\\\_1\_2\\TextFile1.txt", true))

{

sw.WriteLine(Text);

}

}

*void IGetIt.Print()*

*{*

*string rez = "Subject: " + this.Subject + " /Text: " + this.Text + " /addresTo: " + this.addresTo + " /addresFrom: " + this.addresFrom + " /Signature: " + this.Signature;*

*Console.WriteLine(rez);*

*}*

*void IGetIt.Save(string path, Letter name1)*

*{*

*using (StreamWriter writer = new StreamWriter(path, false))*

*{*

*writer.WriteLine(name1);*

*}*

*}*

public override bool Equals(object? obj)

{

if (obj is Letter)

{

if (this.Text.CompareTo(((Letter)obj).Text) == 0 && this.Signature.CompareTo(((Letter)obj).Signature) == 0)

{

return true;

}

else

{

return false;

}

}

else

{

return false;

}

}

public override string ToString()

{

return $"{Subject} {Text} {addressTo.Fio} {addressFrom.Fio} {Signature}";

}

}

}

**Interface.cs**

namespace \_1\_2

{

internal interface IGetIt

{

public void Read();

public void Print();

}

}

**Box.cs**

namespace ConsoleApp1

{

class Box

{

public List<Letter> letters = new List<Letter>();

public void Add(Letter name)

{

letters.Add(name);

}

public void Print()

{

foreach (object n in letters)

{

Console.WriteLine(n);

}

}

}

}

**2DPoint**

**2DPath.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace pr2

{

internal class \_2DPath

{

private List<\_2DPoint> path = new List<\_2DPoint>();

public void Add(\_2DPoint point)

{

path.Add(point);

}

public void Delete(\_2DPoint point)

{

if (path == null)

{

throw new DeleteException("Список пустой!");

}

path.Remove(point);

}

public void Clear()

{

path.Clear();

}

public List<\_2DPoint> GetPath()

{

return this.path;

}

public static bool operator ==(\_2DPath p1, \_2DPath p2)

{

return p1.GetPath().Count() == p2.GetPath().Count();

}

public static bool operator !=(\_2DPath p1, \_2DPath p2)

{

return p1.GetPath().Count() != p2.GetPath().Count();

}

public (int, int, int, int) CountPoints()

{

int count1 = (from p in this.GetPath()

where p.X > 0 && p.Y > 0

select p).Count();

int count2 = (from p in this.GetPath()

where p.X < 0 && p.Y > 0

select p).Count();

int count3 = (from p in this.GetPath()

where p.X < 0 && p.Y < 0

select p).Count();

int count4 = (from p in this.GetPath()

where p.X > 0 && p.Y < 0

select p).Count();

return (count1, count2, count3, count4);

}

}

}

**2DPoint.cs**

namespace pr2

{

internal class \_2DPoint

{

public delegate void EventHandler(\_2DPoint p);

public int X { get; set; }

public int Y { get; set; }

public \_2DPoint(int x, int y)

{

X = x;

Y = y;

}

public void ChangeValue()

{

Change(this);

}

public event EventHandler Change;

public void ChangeSign()

{

X = -X;

Y = -Y;

}

}

}

**DeleteExeption.cs**

namespace pr2

{

internal class DeleteException: Exception

{

public DeleteException(string message): base(message) { }

}

}

**Program.cs**

using System;

using System.Reflection;

namespace pr2

{

internal static class Program

{

public static void Main(string[] args)

{

\_2DPoint p1 = new \_2DPoint(1, 2);

\_2DPoint p2 = new \_2DPoint(3, 4);

\_2DPoint p3 = new \_2DPoint(5, 6);

\_2DPath path = new \_2DPath();

path.Add(p1);

path.Add(p2);

path.Add(p3);

Console.WriteLine(path.CountPoints());

p1.Change += ChangeSigns;

p1.ChangeValue();

Console.WriteLine($"{p1.X} {p1.Y}");

Console.WriteLine(path.CountPoints());

ConstructorInfo[] c = path.GetType().GetConstructors();

Console.WriteLine(c.Length);

FieldInfo[] f = path.GetType().GetFields();

Console.WriteLine(f.Length);

}

public static void ChangeSigns(\_2DPoint p)

{

p.ChangeSign();

}

}

}

**SomeString**

**Program.cs**

using System;

using System.Xml;

namespace \_3\_3

{

internal static class Program

{

public static void Main(string[] args)

{

using (StreamWriter sw = new StreamWriter("C:\\Users\\looog\\source\\repos\\\_3\_3\\\_3\_3\\TextFile1.txt", false))

{

SomeString<string> s1 = new SomeString<string>(" ujc!... djc");

SomeString<string> s2 = new SomeString<string>(" ujc df c");

SomeString<string> s3 = new SomeString<string>(" uj c djc");

SomeString<string> s4 = new SomeString<string>(" uj c djc");

SomeString<string> s5 = new SomeString<string>(" uj c dj c");

Console.WriteLine(s1.CountProbel());

Console.WriteLine(s1.RemoveSigns().someString);

SomeString<string>[] array = { s1, s2, s3, s4, s5 };

int countSpace = array.Select(s => s.CountProbel()).Sum();

} }}

**SomeString.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3\_3

{

internal class SomeString<T> : IComparer<T>

{

public string someString;

public SomeString(string someString)

{

this.someString = someString;

}

public int Compare(T s1, T s2)

{

return 1;

}

public override bool Equals(object? obj)

{

if (obj is string)

return someString.Length == ((string)obj).Length && someString[0] == ((string)obj)[0] && someString[someString.Length - 1] == ((string)obj)[((string)obj).Length - 1];

else

return false;

}

public static SomeString<T> operator +(SomeString<T> main, SomeString<T> content)

{

return main + content;

}

public static SomeString<T> operator -(SomeString<T> main, int a)

{

if (main != null)

return new SomeString<T>(main.someString.Remove(0, a));

else

throw new Exception("String is empty!");

}

}

}

**StaticOperators.cs**

namespace \_3\_3

{

internal static class StaticOperators

{

public static int CountProbel(this SomeString<string> someStr)

{

return someStr.someString

.ToArray()

.Where(ch => ch == ' ')

.Count();

}

public static SomeString<string> RemoveSigns(this SomeString<string> str)

{

return new SomeString<string>(String.Join("", str.someString.Split('.', ',', ';', '!', ':', '-')));}}}

**USERS(1)**

using System.Runtime.Serialization.Formatters.Binary;

using System.Xml.Serialization;

using System.Runtime.Serialization.Json;

**Interface.cs**

namespace UserEmailPassword

{

public interface IComparable

{

bool CompareTo(User user2);

}

**User.cs**

[Serializable]

public class User : IComparable

{

private string Email { get; set; }

private string Password { get; set; }

public enum myEnum

{

signin,

signup

}

public myEnum status;

public User(string email, string password, myEnum Status)

{

this.Email = email;

this.Password = password;

status = Status;

}

public override bool Equals(object? obj)

{

return base.Equals(obj);

}

public override int GetHashCode()

{

return base.GetHashCode();

}

public override string ToString()

{

return base.ToString();

}

bool IComparable.CompareTo(User user2)

{

return Email == user2.Email;

}

}

**WebNet.cs**

[Serializable]

class WebNet

{

public static LinkedList<User> users = new LinkedList<User>();

public void Add(User user)

{

users.AddLast(user);

}

public void Remove(User user)

{

users.Remove(user);

}

}

**Program.cs**

class Program

{

static void Main()

{

User user1 = new User("123@mail.com", "123", User.myEnum.signin);

User user2 = new User("123@mail.com", "123", User.myEnum.signup);

User user3 = new User("13@mail.com", "12323", User.myEnum.signin);

User user4 = new User("1223@mail.com", "12233", User.myEnum.signup);

User user5 = new User("12333@mail.com", "121233", User.myEnum.signin);

Console.WriteLine(((IComparable)user1).CompareTo(user2));

WebNet github = new WebNet();

github.Add(user1);

github.Add(user2);

github.Add(user3);

github.Add(user4);

github.Add(user5);

var myQuery = from m in WebNet.users

where m.status == User.myEnum.signin

select m;

var xmlFormatter = new XmlSerializer(typeof(WebNet));

using (FileStream fs = new FileStream("data.xml", FileMode.OpenOrCreate))

{

xmlFormatter.Serialize(fs, github);

}

var binatyFormatter = new BinaryFormatter();

using (FileStream fs = new FileStream("binary.bin", FileMode.OpenOrCreate))

{

binatyFormatter.Serialize(fs, user1);

}

var jsonFormatter = new DataContractJsonSerializer(typeof(User));

using (FileStream fs = new FileStream("data.json", FileMode.OpenOrCreate))

{

jsonFormatter.WriteObject(fs, user1);

}

}

}

}

**Bill(Wallet)**

using System.Runtime.Serialization;

using System.Runtime.Serialization.Json;

using System.Runtime.Serialization.Formatters.Binary;

using System.Xml.Serialization;

**Bill.cs**

namespace INumber

{

interface INumber

{

public int Number { get; set; }

}

[Serializable]

public class Bill : INumber

{

public int num;

public Bill(int userNum)

{

num = userNum;

}

public Bill() { }

public int Number

{

get

{

return num;

}

set

{

try

{

if ((num == 5 || num == 10 || num == 20 || num == 50 || num == 100) && num > 0)

num = value;

else

throw new Error();

}

catch (Error ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

**Wallet.cs**

[Serializable, DataContract]

public class Wallet<T> where T : Bill

{

public List<T> cashMoney = new List<T>();

public Wallet() { }

public void Add(T ad)

{

try

{

if (cashMoney.Count < 200)

cashMoney.Add(ad);

else

throw new TooMuchMoney();

}

catch (TooMuchMoney ex)

{

Console.WriteLine(ex.Message);

}

}

public void Remove()

{

try

{

if (cashMoney.Count > 0)

{

var minBill = (from i in cashMoney

orderby i.Number

select i).First();

cashMoney.Remove(minBill);

}

else

{

throw new NoBillInWallet();

}

}

catch (NoBillInWallet ex)

{

Console.WriteLine(ex.Message);

}

}

public void Show()

{

foreach (var i in cashMoney)

Console.WriteLine(i.Number);

}

}

**Exeption.cs??**

class TooMuchMoney : Exception

{

public TooMuchMoney() : base("Too much money!") { }

}

class NoBillInWallet : Exception

{

public NoBillInWallet() : base("No Bill In the Wallet!") { }

}

class Error : Exception

{

public Error() : base("The cost of the bill doesnt exist!") { }

}

**Progrma.cs**

class Program

{

static void Main()

{

Wallet<Bill> wallet = new Wallet<Bill>();

Bill dollar = new(100);

Bill euro = new(50);

Bill byn = new(200);

for (int i = 0; i < 201; i++)

wallet.Add(new(i));

for (int i = 0; i < 201; i++)

wallet.Remove();

wallet.Add(dollar);

wallet.Add(euro);

wallet.Add(byn);

wallet.Remove();

wallet.Show();

var fiveDollarBill = from bill in wallet.cashMoney

where bill.Number == 5

select bill;

Console.WriteLine(fiveDollarBill.Count());

var tenDollarBill = from bill in wallet.cashMoney

where bill.Number == 10

select bill;

Console.WriteLine(tenDollarBill.Count());

var twentyDollarBill = from bill in wallet.cashMoney

where bill.Number == 20

select bill;

Console.WriteLine(twentyDollarBill.Count());

var fiftyDollarBill = from bill in wallet.cashMoney

where bill.Number == 50

select bill;

Console.WriteLine(fiftyDollarBill.Count());

var hundreedDollarBill = from bill in wallet.cashMoney

where bill.Number == 100

select bill;

Console.WriteLine(hundreedDollarBill.Count());

var jsonFormatter = new DataContractJsonSerializer(typeof(Wallet<Bill>));

using (var sw = new FileStream("data.json", FileMode.OpenOrCreate))

{

jsonFormatter.WriteObject(sw, wallet);

}

var xmlFormatter = new XmlSerializer(typeof(Wallet<Bill>));

using (var sw = new FileStream("data.xml", FileMode.OpenOrCreate))

{

xmlFormatter.Serialize(sw, wallet);

}

var binaryFormatter = new BinaryFormatter();

using (var sw = new FileStream("data.xml", FileMode.OpenOrCreate))

{

binariyFormatter.Serialize(sw, wallet);

}

}}}

**User(PASSWORD)**

class AbstractUser

{

public DateTime Data { get; set; }

}

class Exception1 : Exception

{

public Exception1() : base("Пароль слишком короткий или слишком длинный") { }

}

class Exception2 : Exception

{

public Exception2() : base("Пароль не может состоять только из цифр") { }

}

class User : AbstractUser

{

public override string ToString()

{

return (" LOGIN " + Login + " PASSWORD " + password);

}

public string Login { get; set; }

private string password;

public string lowList = "abcdefghijklmnoprstuvwxyz";

public string Password

{

get { return password; }

set

{

if (value.Length < 6 || value.Length > 12)

{

throw new Exception1();

}

else

{

if (value.IndexOfAny(lowList.ToCharArray()) == -1)

{

throw new Exception2();

}

else

{

password = value;

}

}

}

}

public User(string login, string password)

{

Password = password;

Login = login;

}

static void Main(string[] args)

{

try

{

User user1 = new User("flex", "1242vv323");

User user2 = new User("Podliva", "qweqweqw");

User user3 = new User("vlad", "123asdsa2");

User user4 = new User("help", "111d3123");

List<User> users = new List<User>(4);

users.Add(user1);

users.Add(user2);

users.Add(user3);

users.Add(user4);

// linq не выходит

Console.WriteLine(user1.ToString());

Console.WriteLine(user2.ToString());

Console.WriteLine(user3.ToString());

Console.WriteLine(user4.ToString());

}

catch (Exception1 e)

{

Console.WriteLine(e.Message);

}

catch (Exception2 e)

{

Console.WriteLine(e.Message);

}

}

**Скидка**

class Item

{

public string Name { get; set; }

public int ID { get; set; }

public double Price { get; set; }

public Item(string name, int ID, double price)

{

Name = name;

this.ID = ID;

Price = price;

}

public override string ToString()

{

return base.ToString() + " " + Name + " " + ID + " " + Price;

}

public override int GetHashCode()

{

return base.GetHashCode();

}

public void Start()

{

this.Price = this.Price \* 0.5;

Console.WriteLine("sale");

class Manager

{

public delegate void Sale();

public event Sale sales;

public void SaleOn()

{

sales();

}

}

class Shop : IEnumerable

{

public Queue<Item> queue = new Queue<Item>();

public Queue<Shop> shop = new Queue<Shop>();

public Queue<Item> GetQueue { get { return queue; } }

public void qAdd(Item obj)

{

queue.Enqueue(obj);

}

public void qRemove(Item obj)

{

queue.Dequeue();

}

public void qClear(Item obj)

{

queue.Clear();

}

public IEnumerator GetEnumerator()

{

return queue.GetEnumerator();

}

public static int operator +(Shop obj)

{

obj.sAdd(obj);

return 0;

}

public static int operator -(Shop obj)

{

obj.sRemove(obj);

return 0;

}

public void sAdd(Shop obj)

{

shop.Enqueue(obj);

}

public void sRemove(Shop obj)

{

shop.Dequeue();

}

static void Main(string[] args)

{

Item item1 = new Item("Диван", 1, 32);

Item item2 = new Item("Диван", 2, 40);

Item item3 = new Item("Диван", 3, 30);

Item item4 = new Item("Диван", 4, 36);

Shop que = new Shop();

que.qAdd(item1);

que.qAdd(item2);

que.qAdd(item3);

que.qAdd(item4);

Console.WriteLine(item1.ToString());

Console.WriteLine(item1.GetHashCode());

foreach (var item in que)

{

Console.WriteLine(item);

}

Manager manager = new Manager();

manager.sales += item1.Start;

manager.sales += item2.Start;

manager.sales += item3.Start;

manager.SaleOn();

foreach (var item in que)

{

Console.WriteLine(item);

}

string whatName = "Диван";

var itemCount = (from t in que.queue where t.Name == whatName select t).Count();

Console.WriteLine(itemCount);

}

Создать класс **Item** со свойствами Name, ID,Price. Создать класс Manager с событием sale(распродажа). Созд вещи и добавить их в обобщ коллекцию типа очередь. Подпишите некоторые вещи на событие sale

//Реакция на событие следующая, цена Item уменьшается на 70%. Продемонстр ситуацию события и вывести содерж очереди на консоль

using System;

using System.Collections.Generic;

using System.Linq;

using System.IO;

namespace Exam

{

class MainClass

{

public static void Main(string[] args)

{

Queue<Item> a = new Queue<Item>();

Item first = new Item("Мыло", 12, 2000);

Item second = new Item("Maslo", 122, 9000);

Item third = new Item("Vika", 178, 90);

Manager b = new Manager();

b.sale += (sender, e) =>

{

first.price = (int)(first.price \* 0.3);

second.price = (int)(second.price \* 0.3);

third.price = (int)(third.price \* 0.3);

};

b.Sale();

a.Enqueue(first);

a.Enqueue(second);

a.Enqueue(third);

}

}

public class Item

{

public string name { get; set; }

public int id { get; set; }

public int price { get; set; }

public Item(string name, int ID, int price)

{

this.name = name;

this.id = ID;

this.price = price;

}

}

public class Manager

{

public event EventHandler sale;

public void Sale()

{

if (sale != null)

sale(this, new EventArgs());

}

}}

**USER(2)**

создать клас User с закрытымип полями login, password. переопределить в классе все Public методы object. перегрузить метод CompareTo стандартного унаследованного интерфейса IComparable который сравнивает пользователей по логину и паролю. создать и сравнить 3-х юзеров, создать LinkedList<user> с 5-ю юзерами. используя LINQ найти в коллекции юзеров, у которых длина пароля меньше 8 и содержит только цифры

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

using System.Collections.Generic;

using System.IO;

using System.Runtime.Serialization;

namespace OOP

{

class User : IComparable<User>

{

public readonly string login;

public readonly string pasword;

public User(string l, string p)

{

login = l;

pasword = p;

}

//перегрузка открытых метов Object

public override int GetHashCode()

{

return base.GetHashCode();

}

public override bool Equals(object obj)

{

return base.Equals(obj);

}

public override string ToString()

{

return base.ToString();

}

//перегрузка метода IComparable

public int CompareTo(User obj)

{

int result = -1;

if (obj != null)

{

if (login == obj.login && pasword == obj.pasword)

result = 1;

}

return result;

}

}

class Program

{

static void Main(string[] args)

{

User user1 = new User("Dasha", "556156847a");

User user2 = new User("Vasya58", "8544");

User user3 = new User("Vasya58", "8544");

User user4 = new User("Dima1", "8544gs");

User user5 = new User("Vika", "8er564644");

LinkedList<User> linkedList = new LinkedList<User>();

LinkedListNode<User> listNode = new LinkedListNode<User>(user1);

linkedList.AddFirst(listNode);

linkedList.AddAfter(listNode, user2);

linkedList.AddAfter(listNode, user3);

linkedList.AddAfter(listNode, user4);

linkedList.AddAfter(listNode, user5);

for (int i = 0; i < linkedList.Count; i++)

{

Console.WriteLine(listNode.Value.login);

listNode = listNode.Next;

}

IEnumerable<User> newCollection = linkedList.Where(n => (n.pasword.Length) < 8).Select(n => n).ToArray();

Console.WriteLine("Отбор по паролям");

foreach (User item in newCollection)

{

Console.WriteLine(item.login);

}

Console.WriteLine($"user1 vs user2 {user1.CompareTo(user2)}");

Console.WriteLine($"user2 vs user3 {user2.CompareTo(user3)}");

}

}

}

**Самолеты**

public interface IManage

{

float MaxAvg();

}

public enum Form

{

our = 1,

your,

my

}

public class ZiroException : Exception

{

public ZiroException(string message) : base(message)

{

Console.WriteLine(message);

}

}

public class Company : IManage

{

public string name { get; set; }

public int count { get; set; }

Form form { get; set; }

public int year1 { get; set; }

public int year2 { get; set; }

public int year3 { get; set; }

public int year4 { get; set; }

public Company(string \_name, int \_count, Form \_form, int \_year1, int \_year2, int \_year3, int \_year4)

{

this.name = \_name;

this.count = \_count;

this.form = \_form;

this.year1 = \_year1;

this.year2 = \_year2;

this.year3 = \_year3;

this.year4 = \_year4;

}

public override string ToString()

{

return $"{name} {count} {form} {year1} {year2} {year3} {year4}";

}

public (int, int) MinMaxMoney()

{

List<int> money = new List<int>();

money.Add(year1);

money.Add(year2);

money.Add(year3);

money.Add(year4);

int min = money.Min();

int max = money.Max();

var result = (min, max);

return result;

}

float IManage.MaxAvg()

{

float sum = 0;

float result;

sum = (float)(year1 + year2 + year3 + year4);

result = sum / 4;

return result;

}

public static Company operator ++(Company obj)

{

obj.count++;

return obj;

}

public static Company operator --(Company obj)

{

try

{

if (obj.count == 0)

throw new ZiroException("Null");

}

catch (ZiroException ex)

{

Console.WriteLine(ex.Message);

}

obj.count--;

return obj;

}

public static Company operator +(Company obj, int i)

{

obj.count = obj.count + i;

return obj;

}

}

public static class Extension

{

public static Company DeleteInfo(Company company)

{

company.year1 = 0;

company.year2 = 0;

company.year3 = 0;

company.year4 = 0;

return company;

}

}

class Program

{

static void Main(string[] args)

{

Company company = new Company("EPAM", 450, Form.your, 45, 57, 38, 39);

Console.WriteLine(company.MinMaxMoney());

Console.WriteLine(((IManage)company).MaxAvg());

Console.WriteLine(company.ToString());

company++;

Console.WriteLine(company.ToString());

company--;

Extension.DeleteInfo(company);

Console.WriteLine(company.ToString());

}

}

}

**BSTU**

Создать класс BSTUStudent, в котором следующие поля: имя, группа, курс, специальность- задача перечислением poit,isit,web,mobile.И 4 отметки за экзамены

Создать метод в классе BSTUStudent, которой возвращает кортеж, содержащий. Мин , макс и ср отметки за экзы.

Создать класс Group, который хранит студентов в одной из необобщенных коллекций .Net и присвоить ей 4 студентов.

С помощью Linq.Выведите два элемента с наибольшим ср. Баллом

Создать интерфейс IClearnable с методом Clearn , который очищает коллекцию в классе Group. Реализовать сам интерфейс

enum Specialization

{

isit,

poibms,

poit,

deivi

}

class BSTUStudent

{

public override string ToString()

{

return ("Имя " + Name + " Группа " + Group + " Специальность " + Specialization);

}

public string Name { get; set; }

public int Group { get; set; }

public int Course { get; set; }

public Specialization Specialization { get; set; }

public int Mark1 { get; set; }

public int Mark2 { get; set; }

public int Mark3 { get; set; }

public int Mark4 { get; set; }

public BSTUStudent(Specialization specialization, string name, int group, int course, int mark1, int mark2, int mark3, int mark4)

{

Specialization = specialization;

Name = name;

Group = group;

Course = course;

Mark1 = mark1;

Mark2 = mark2;

Mark3 = mark3;

Mark4 = mark4;

}

public static (int, int, double) Marks(BSTUStudent obj)

{

int[] marks = { obj.Mark1, obj.Mark2, obj.Mark3, obj.Mark4 };

int max = marks.Max();

int min = marks.Min();

double average = marks.Average();

var result = (min, max, average);

return result;

}

}

}

public interface IClearnable

{

void lClearn();

}

class Groups : IClearnable

{

public List<BSTUStudent> Array = new List<BSTUStudent>();

public List<BSTUStudent> GetArray { get { return Array; } }

public void lAdd(BSTUStudent obj)

{

Array.Add(obj);

}

public void lClearn()

{

Array.Clear();

}

static void Main(string[] args)

{

BSTUStudent student1 = new BSTUStudent(Specialization.poibms, "Vlad", 8, 2, 4, 4, 6, 5);

BSTUStudent student2 = new BSTUStudent(Specialization.poit, "Katya", 4, 3, 7, 4, 5, 5);

BSTUStudent student3 = new BSTUStudent(Specialization.poibms, "Nikita", 8, 2, 6, 5, 6, 5);

BSTUStudent student4 = new BSTUStudent(Specialization.poibms, "Kostya", 8, 2, 4, 4, 4, 4);

BSTUStudent.Marks(student1);

Groups mobilki = new Groups();

mobilki.lAdd(student1);

mobilki.lAdd(student2);

mobilki.lAdd(student3);

mobilki.lAdd(student4);

//var average = mobilki.Array.Select((Mark1, Mark2) => Mark1 + Mark2).Sum() / ratings.Sum();

//var maxAverage = (from t in mobilki.Array where t.Marks select t).Take(2);

//Console.WriteLine(maxAverage);

**The first**

using System.Runtime.Serialization.Json;

namespace ExampleOne

{

class Program

{

public static void Main()

{

List<Rectangle> list = new List<Rectangle>();

Rectangle rectangleFirst = new Rectangle(12, 12, 12, 12, "cyan");

Rectangle rectangleSecond = new Rectangle(1, 1, 2, 23, "green");

Rectangle rectangleThird = new Rectangle(12, 12, 22, 32, "grey");

Rectangle rectangleFourth = new Rectangle(12, 12, 12, 12, "yellow");

Rectangle rectangleFifth = new Rectangle(12, 12, 12, 12, "orange");

Rectangle rectangleSixth = new Rectangle(12, 52, 6, 52, "purple");

list.Add(rectangleFirst);

list.Add(rectangleSecond);

list.Add(rectangleThird);

list.Add(rectangleFourth);

list.Add(rectangleFifth);

list.Add(rectangleSixth);

//foreach (var l in list)

// Console.WriteLine(l.ToString());

((Figure)rectangleFifth).Print();

int a = 6;

Console.WriteLine(rectangleFifth + a);

var firstLetterOfsortedList = (from f in list

orderby f.x orderby f.y orderby f.x\*f.y

select f).First();

var lastLetterOfsortedList = (from f in list

orderby f.x

orderby f.y

orderby f.x \* f.y

select f).Last();

Console.WriteLine("Первый объект: " + firstLetterOfsortedList.Square() + ", Последний: " + lastLetterOfsortedList.Square());

var jsonSerializer = new DataContractJsonSerializer(typeof(List<Rectangle>));

using (var file = new FileStream("log.json", FileMode.OpenOrCreate))

{

jsonSerializer.WriteObject(file, list);

}

}

}

}

interface Figure

{

public virtual void Print()

{

}

}

[DataContract]

class Rectangle : Figure

{

[NonSerialized]

public int x;

[NonSerialized]

public int y;

public int width;

public int height;

public string? color;

void Figure.Print()

{

Console.WriteLine($"Данный прямоугольник имеет длину: {height} и ширину {width}");

}

public Rectangle()

{

Console.WriteLine();

}

public Rectangle(int X, int Y, string Color)

{

x = X;

y = Y;

color = Color;

}

public Rectangle(int X, int Y, int Width, int Height, string Color)

{

x = X;

y = Y;

width = Width;

height = Height;

color = Color;

}

public override string ToString()

{

return $"Данный прямоугольник имеет длину: {height} и ширину {width}";

}

public static Rectangle operator +(Rectangle rect, int a)

{

rect.width += a;

rect.height += a;

return rect;

}

public float Square()

{

return width \* height;

}

}

* //Создать класс **Item** со свойствами Name, ID,Price. Создать класс Manager с событием sale(распродажа). Созд вещи и добавить их в обобщ коллекцию типа очередь. Подпишите некоторые вещи на событие sale   
  //Реакция на событие следующая, цена Item уменьшается на 70%. Продемонстр ситуацию события и вывести содерж очереди на консоль

using System;

using System.Collections.Generic;

using System.Linq;

using System.IO;

namespace Exam

{

class MainClass

{

public static void Main(string[] args)

{

Queue<Item> a = new Queue<Item>();

Item first = new Item("Мыло", 12, 2000);

Item second = new Item("Maslo", 122, 9000);

Item third = new Item("Vika", 178, 90);

Manager b = new Manager();

b.sale += (sender, e) =>

{

first.price = (int)(first.price \* 0.3);

second.price = (int)(second.price \* 0.3);

third.price = (int)(third.price \* 0.3);

};

b.Sale();

a.Enqueue(first);

a.Enqueue(second);

a.Enqueue(third);

}

}

public class Item

{

public string name { get; set; }

public int id { get; set; }

public int price { get; set; }

public Item(string name, int ID, int price)

{

this.name = name;

this.id = ID;

this.price = price;

}

}

public class Manager

{

public event EventHandler sale;

public void Sale()

{

if (sale != null)

sale(this, new EventArgs());

}

}

}

Реализовать обобщённый класс **вектор**. Вложить в него обобщённую коллекцию .NET. Наследовать интерфейс IAction с методами добавления, удаления, вывода и очистки. Методы реализовать в классе.  
Добавить обработку исключений с finally.  
Провести проверку с целочисленным типов и с типом студент (параметры определите сами).  
interface IAction<T>

{

void Add(Queue<T> type);

void Del();

void Show();

void Clear();

}

public class Student

{

string name;

int kurs;

public Student()

{

}

public Student(string nm, int k)

{

this.name = nm;

this.kurs = k;

}

public override string ToString()

{

return this.name + this.kurs;

}

}

public class Vector<T> : IAction<T> where T : new()

{

Queue<T>[] queue = new Queue<T>[4];

int size = 0;

public void Add(Queue<T> q)

{

queue[size++] = q;

}

public void Del()

{

if (size == 0)

{

throw new Exception("Del from empty");

}

queue[--size] = null;

}

public void Show()

{

int count = 0;

foreach (Queue<T> q in queue)

{

if (q != null && count <= size)

{

count++;

foreach (T t in q)

{

Console.WriteLine(t.ToString());

}

}

else

{

Console.WriteLine("Empty");

}

}

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

}

public void Clear()

{

while (size != 0)

{

this.Del();

}

}

}

try

{

Queue<int> queue1 = new Queue<int>();

queue1.Enqueue(1);

queue1.Enqueue(3);

queue1.Enqueue(42);

queue1.Enqueue(2);

queue1.Enqueue(5);

Queue<int> queue2 = new Queue<int>();

queue2.Enqueue(-1);

queue2.Enqueue(0);

queue2.Enqueue(8);

queue2.Enqueue(1);

queue2.Enqueue(4);

Vector<int> vector = new Vector<int>();

vector.Add(queue1);

vector.Add(queue2);

vector.Show();

vector.Del();

vector.Show();

vector.Add(queue1);

vector.Clear();

vector.Del();

vector.Show();

Queue<Student> q1 = new Queue<Student>();

Student stud1 = new Student("Andrei", 2);

Student stud2 = new Student("Alena", 2);

q1.Enqueue(stud1);

q1.Enqueue(stud2);

Queue<Student> q2 = new Queue<Student>();

Student st1 = new Student("Vadim", 1);

Student st2 = new Student("Egor", 1);

q1.Enqueue(st1);

q1.Enqueue(st2);

Vector<Student> vector2 = new Vector<Student>();

vector2.Add(q1);

vector2.Add(q2);

vector2.Show();

}

catch (Exception ex)

{

Console.WriteLine("Error: " + ex.Message);

}

finally

{

Console.WriteLine("done");

}

Console.ReadKey();

создать абстрактный класс **Transport** (состав поизвольно). Создайте интерфейс IAir c методами Fly и Check.Наследуйте их в классе Air, который содержит свойства Speed(скорость)   
CountOfPass(число пассажиров)и Status. Status принимает одно из значений переичсления fly,ready,error. Прив вызове метода Fly проверяется скорость,если она <220,гененрируйте исключение   
и выставляете статус error самолету. Check или ready(в пределах допустимого) или error(если их слишком мало или много). Создайте самолет и протестируйте его. Нет перечисления в этом варике

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Экзамен

{

class Program

{

static void Main(string[] args)

{

try

{

Air a = new Air();

a.Car = "Самолёт";

a.Speed = 130;

a.CountOfPass = 20;

a.Fly(a.Speed);

a.Check(a.CountOfPass);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

interface IAir

{

void Fly(int speed);

void Check(int pass);

}

class Air : Transport, IAir

{

public string Car { get { return car; } set { car = value; } }

private int speed;

public int Speed { get { return speed; } set { speed = value; } }

private int countofpass;

public int CountOfPass { get { return countofpass; } set { countofpass = value; } }

private string status;

public string Status { get { return status; } set { status = value; } }

public void Fly(int speed)

{

if (speed >= 220)

{

Console.WriteLine("Летит");

}

else

{

throw new Exception("Speed < 220");

Status = "error";

}

}

Per p;

public enum Per : int { fly, ready, error }

public void Check(int pass)

{

if (pass <= 30)

{

Console.WriteLine("Норм пассажиров");

Status = "ready";

}

else

{

throw new Exception("Pass > 30");

Status = "error";

}

}

}

abstract class Transport

{

public string car;

}

}

}

есть класс **Card**, с приватными полями balance, класс ExDate, содержащий свойства Monght, Yehr(два последние числа года), number. Реализован интерфейс IPay с мотодом Pay(int, ExDate). Привести явное приобразование интерфейса в классе Card. При условии, если баланс меньше - 100 вызвать исключение. Реализовать работу с Card. Создать массив значений баланса. Используя LINQ найти карту с максимальным балансом, вывести ее номер карты

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp3

{

public class Card: IPay

{

private int balance;

public class ExDate

{

public int balance1;

public int monght;

public int Monght

{

get

{

return monght;

}

set

{

monght = value;

}

}

public int yers;

public int Yehrs

{

get

{

return yers;

}

set

{

yers = value;

}

}

public int number;

//public void Input(int a)

//{

// if (a == balance1)

// {

// Console.WriteLine(number);

// }

//}

}

void IPay.Pay(int a, ExDate exDate)

{

if (a <= -100)

{

throw new Exception("Error");

}

balance = a;

exDate.balance1 = balance;

exDate.number = balance + 1000;

}

}

public interface IPay

{

void Pay(int a, Card.ExDate exDate);

}

class Program

{

static void Main(string[] args)

{

Card.ExDate card = new Card.ExDate();

Card.ExDate card1 = new Card.ExDate();

Card.ExDate card2 = new Card.ExDate();

var rdf = new Card();

((IPay)rdf).Pay(10, card);

((IPay)rdf).Pay(-19, card1);

((IPay)rdf).Pay(90, card2);

Card.ExDate[] arr = { card, card1, card2 };

var lin = (from i in arr

select i.balance1).Max();

Console.WriteLine(lin);

foreach(var i in arr)

{

if(i.balance1 == lin)

{

Console.WriteLine("Номер карты с балансом {0}: {1} ",lin,i.number);

}

}

}

}

}

**КНОПКА**

namespace \_7\_3

{

public class Button: CheckButton

{

public string caption;

(int x, int y) startpoint;

public int X

{

get

{

return startpoint.x;

}

set

{

value = startpoint.x;

}

}

public int Y

{

get

{

return startpoint.y;

}

set

{

value = startpoint.y;

}

}

public double w;

public double h;

public Button(string caption, int x, int y, double w, double h, State state)

{

this.caption = caption;

this.startpoint.x = x;

this.startpoint.y = y;

this.w = w;

this.h = h;

this.state = state;

}

public override string ToString()

{

return $"Caption: {caption} Startpoint: x = {startpoint.x} y = {startpoint.y} Width: {w} Height: {h}";

}

public override bool Equals(object obj)

{

if (obj == null) return false;

if (obj.GetType() != this.GetType()) return false;

Button button = (Button)obj;

return this.caption == button.caption && this.w == button.w && this.h == button.h;

}

public override int GetHashCode()

{

return base.GetHashCode();

}

public void Check()

{

if (state == State.check)

state = State.uncheck;

else state = State.check;

}

public void Zoom(double q)

{

this.w = this.w \* q;

this.h = this.h \* q;

}

double square;

public double Square()

{

square = w \* h;

return square;

}

}

public class CheckButton

{

public State state;

public enum State

{

check = 1,

uncheck

}

}

public class User

{

public int Click { get; set; }

public int Resize { get; set; }

}

class Program

{

static void Main(string[] args)

{

Button button1 = new Button("try", 12, 45, 12.2, 12.7, CheckButton.State.check);

Button button2 = new Button("catch", 34, 5, 11.6, 7.8, CheckButton.State.uncheck);

Button button3 = new Button("finally", 6, 13, 5.6, 7.9, CheckButton.State.uncheck);

User user = new User();

Console.WriteLine(button1.ToString());

Console.WriteLine(button2.ToString());

Console.WriteLine(button3.ToString());

button1.Check();

button2.Check();

button3.Zoom(0.4);

Console.WriteLine(button1.Equals(button2));

Console.WriteLine(button1.ToString());

Console.WriteLine(button2.ToString());

Console.WriteLine(button3.ToString());

LinkedList<Button> list = new LinkedList<Button>();

list.AddFirst(button1);

list.AddFirst(button2);

list.AddFirst(button3);

foreach (var i in list)

{

Console.WriteLine(i);

}

button1.w = button1.Square();

button2.w = button2.Square();

button3.w = button3.Square();

Console.WriteLine(button1.w);

Console.WriteLine(button2.w);

Console.WriteLine(button3.w);

double z = Convert.ToDouble(Console.ReadLine());

var select = from i in list

where i.w == z

select i;

foreach (var i in select)

{

Console.WriteLine(i.w);

}

}

}

}

**SOMESTRING**

class Program

{

static void Main(string[] args)

{

SomeString fStr = new SomeString("ШИЗА");

SomeString sStr = new SomeString("КРУТО");

if(fStr.Equals("ЗАМЕЧАТЕЛЬНО"))

SomeString.PrintToFile("ПРАВИЛЬНО, сестра");

else

SomeString.PrintToFile("ОШИБКА, брат");

SomeString.PrintToFile(fStr.Compare(2,4));

SomeString rez = fStr + sStr;

SomeString.PrintToFile(rez.MyString);

SomeString rez1 = fStr-sStr;

SomeString.PrintToFile(rez1.MyString);

SomeString.PrintToFile("Static Class: ");

SomeString str = new SomeString

("Hello my bro. How are you ? Now, we speak english, it is so good . Goodby my bro... ");

SomeString.PrintToFile(str.CountSpace());

SomeString cho = str.DellOther();

SomeString.PrintToFile(cho.MyString);

SomeString[] myArr = { fStr, sStr, str };

var z = (from t in myArr select t.CountSpace()).Sum();

SomeString.PrintToFile(z);

}

}

class ExClass:Exception

{

public override string Message => "Empty string";

}

public class SomeString : IComparer

{

private string myString;

public SomeString(){}

public SomeString(string myString)

{

this.myString = myString;

}

public string MyString { get => myString; set => myString = value; }

public int Compare(object x, object y)

{

int x1 = (int)x;

int y1 = (int)y;

if (x1 > y1)

return 1;

else if (x1 < y1)

return -1;

else

return 0;

}

public override bool Equals(object myObj)

{

string obj = (string)myObj;

if (obj.Length == MyString.Length)

{

if (obj[obj.Length - 1] == MyString[MyString.Length - 1])

return true;

else

return false;

}

else

{

return false;

}

}

public static SomeString operator +(SomeString x, SomeString y)

{

try

{

if (x.MyString == "" || y.MyString == "" || x.MyString == null || y.MyString == null)

throw new ExClass();

else

{

SomeString rez = new SomeString();

rez.myString = x.myString + y.myString;

return rez;

}

}

catch (ExClass ex)

{

Console.WriteLine(ex.Message);

SomeString rez = new SomeString();

return rez;

}

}

public static SomeString operator -(SomeString x, SomeString y)

{

try

{

if (x.MyString == "" || y.MyString == "" || x.MyString == null || y.MyString == null)

throw new ExClass();

else

{

SomeString rez = new SomeString();

for (int i = 1; i < x.MyString.Length; i++)

{

rez.MyString += x.MyString[i];

}

return rez;

}

}

catch (ExClass ex)

{

Console.WriteLine(ex.Message);

SomeString rez = new SomeString();

return rez;

}

}

public static void PrintToFile(string str)

{

using(StreamWriter stream = new StreamWriter(@"my.txt",true))

{

stream.WriteLine(str);

}

}

public static void PrintToFile(int num)

{

using (StreamWriter stream = new StreamWriter(@"my.txt",true))

{

stream.WriteLine(num);

}

}

}

public static class WorkClass

{

public static int CountSpace(this SomeString str)

{

int counter=0;

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

if (str.MyString[i] == ' ')

counter++;

return counter;

}

public static SomeString DellOther(this SomeString str)

{

SomeString rez = new SomeString();

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

{

if (str.MyString[i] == ' ' || str.MyString[i] == '.' || str.MyString[i] == ','

|| str.MyString[i] == '-' || str.MyString[i] == '!' || str.MyString[i] == '?'

)

continue;

else

rez.MyString += str.MyString[i];

}

return rez;

}

}

**ISocer**

namespace IsScore

{

interface IScore

{

public int Amount { get; set; }

public void AddBalance(int amount);

public void RemoveBalance(int amount);

}

abstract class Human

{

public string? birthday;

}

class Person : Human, IScore

{

public string? birthday;

public string? name;

public int amount;

public static int humansCount = 0;

public int Amount

{

get { return amount; }

set { amount = value; }

}

public Person(string userName, int userAmount, string userBirthday)

{

name = userName;

amount = userAmount;

birthday = userBirthday;

}

static Person()

{

Console.WriteLine("The Human was Created");

humansCount++;

}

public static void Show()

{

Console.WriteLine(humansCount);

}

public void AddBalance(int userAmount)

{

amount += userAmount;

}

public void RemoveBalance(int userAmount)

{

amount -= userAmount;

}

public override bool Equals(object? obj)

{

Person? somePerson = obj as Person;

if (somePerson.birthday == birthday)

return true;

else

return false;

}

}

class Bank<T> : List<T>

{

public List<T> list = new List<T>();

public void Add(T elem)

{

list.Add(elem);

}

public void Remove(T elem)

{

list.Remove(elem);

}

}

class Program

{

static void Main()

{

Person person = new("vlad", 15, "2003");

Person person2 = new("ars", 17, "2004");

person.AddBalance(15);

person2.AddBalance(15);

Console.WriteLine(person.amount);

Person.Show();

Console.WriteLine(person.Equals(person2));

Bank<Person> belarus = new Bank<Person>();

Bank<Person> alfa = new Bank<Person>();

Bank<Person> vtb = new Bank<Person>();

belarus.Add(person);

belarus.Add(person2);

alfa.Add(person);

alfa.Add(person2);

vtb.Add(person);

vtb.Add(person2);

Console.WriteLine("person's date in belarus bank = ");

string userData = Console.ReadLine();

Task task1 = Task.Run(() =>

{

for (int i = 0; i < belarus.list.Count; i++)

{

if (belarus.list[i].birthday == userData)

Console.WriteLine("Такой клиент есть");

}

});

Console.WriteLine("person's date in alfa bank = ");

userData = Console.ReadLine();

Task task2 = Task.Run(() =>

{

for (int i = 0; i < belarus.list.Count; i++)

{

if (alfa.list[i].birthday == userData)

Console.WriteLine("Такой элемент есть");

}

});

Console.WriteLine("person's date in vtb bank = ");

userData = Console.ReadLine();

Task task3 = Task.Run(() =>

{

for (int i = 0; i < belarus.list.Count; i++)

{

if (vtb.list[i].birthday == userData)

Console.WriteLine("Такой элемент есть");

}

});

}

}

}

**EXAMCARD**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Students

{

public int mark;

public string name;

public string subject;

public Students(string Name, int Mark, string Subject)

{

this.name = Name;

this.mark = Mark;

this.subject = Subject;

}

public Students()

{

}

public override string ToString()

{

return mark + " " + name + " " + subject;

}

}

interface IAction<T>

{

void Add(T a);

void Del(T a);

void Clean();

void Show();

}

class ExamCard<T> : IAction<T> where T : new()

{

public static List<T> list = new List<T>();

public void Add(T a)

{

list.Add(a);

}

public void Del(T a)

{

try

{

if (list.Count == 0)

{

throw new NullSizeCollection("Коллекция пустая");

}

else list.Remove(a);

}

catch (NullSizeCollection ex)

{

Console.WriteLine(ex.Message);

}

}

public void Clean()

{

try

{

if (list.Count == 0)

{

throw new NullSizeCollection("Коллекция пустая");

}

else list.Clear();

}

catch (NullSizeCollection ex)

{

Console.WriteLine(ex.Message);

}

}

public void Show()

{

Console.WriteLine("Вся коллекция: ");

foreach (var l in list)

Console.WriteLine(l);

}

}

class NullSizeCollection : Exception

{

private string message;

public override string Message

{

get

{

return message;

}

}

public NullSizeCollection(string mess)

{

message = mess;

}

}

static class Met

{

public static void qwe(this Students st)

{

Random random = new Random();

st.mark += random.Next(1, 3);

}

}

class Program

{

static void Main(string[] args)

{

Students st1 = new Students("qwe", 6, "rer");

Students st2 = new Students("qwe", 7, "rer");

Students st3 = new Students("qwe", 5, "rer");

Students st4 = new Students("qwe", 3, "rer");

ExamCard<Students> st = new ExamCard<Students>();

((IAction<Students>)st).Add(st1);

((IAction<Students>)st).Add(st2);

((IAction<Students>)st).Add(st3);

((IAction<Students>)st).Add(st4);

((IAction<Students>)st).Show();

((IAction<Students>)st).Del(st3);

var linq1 = from s in ExamCard<Students>.list

where s.mark >= 4

select s;

Console.WriteLine(linq1.Count());

var linq2 = from s in ExamCard<Students>.list

select s.mark;

Console.WriteLine(linq2.Average());

st1.qwe();

Console.WriteLine(st1);

}

}

**Делегаты**

namespace laba8

{

class Director

{

public int salary;

public string position;

public delegate void Fine(Director obj, int salary);

public delegate void Increase(Director obj, int salary, string position);

public event Fine fine;

public event Increase increase;

public Director(int salary, string position)

{

this.salary = salary;

this.position = position;

}

public void Action(int salary, string position)

{

Console.WriteLine(ToString());

Console.Write("Изменения объекта: ");

if (fine != null)

{

fine(this, salary);

}

else

{

Console.Write("Зарплата не изменена. ");

}

if (increase != null)

{

increase(this, salary, position);

}

else

{

Console.Write("Должность не изменена. ");

}

Console.WriteLine();

Console.WriteLine(ToString());

}

public override string ToString()

{

return $"Текущие параметры объекта: зарплата = {this.salary}, должность = {this.position}";

}

}

}

**Progrma.cs**

namespace laba8

{

class Program

{

static void Main(string[] args)

{

Director d1 = new Director(3000, "director");

Director d2 = new Director(4500, "director");

Director d3 = new Director(5000, "director");

ClassEvent CE = new ClassEvent();

d1.fine += CE.DoFine;

d1.increase += CE.DoIncrease;

d2.fine += CE.DoFine;

d1.Action(7000, "finantial director");

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

d2.Action(4700, "director");

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

d3.Action(5000, "director");

string str = "P. O- i? T k-l,a ss";

Func<string, string> A;

Console.WriteLine("--------------Работа со строками--------------");

A = Str.RemoveS;

Console.WriteLine($"Без знаков препинания:\nСтрока до: {str}\nПосле: {A(str)}\n");

A = Str.RemoveSpase;

Console.WriteLine($"Убрать пробелы:\nСтрока до: {str}\nПосле: {A(str)}\n");

A = Str.Upper;

Console.WriteLine($"Заглавные буквы:\nСтрока до: {str}\nПосле: {A(str)}\n");

A = Str.Letter;

Console.WriteLine($"Прописные буквы:\nСтрока до: {str}\nПосле: {A(str)}\n");

A = Str.AddToString;

Console.WriteLine($"Добавление символов:\nСтрока до: {str}\nПосле: {A(str)}\n");

void Hello() => Console.WriteLine("Hello");

void HowAreYou() => Console.WriteLine("How are you?");

Message message = Hello;

message += HowAreYou;

message();

}

delegate void Message();

}}