**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);

**User**

1.Создать абстрактный класс AbstractUser с полем Data.Создать клас User с полями password и login, который является потомком обстрактного класа.

2-3. Создать исключения : 1)Если длинна пароля меньше 6 символов 2)если длинна пароля больше 12 символов 3) если пароль состоит из одних цифр.

4. Создать 4 объекта типа user и поместить их в лист. Продемонстрировать работу исключений.

5. На основе linq написать запрос : найти пользователя, добавленного раньше всех(по полю Data). Вроде как-то так.

**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);

}**Taxi тут есть where :**

1. Определите класс Location (позиция). Он содержит значения широты и долготы – lat, long, speed (скорость). Создайте класс Taxi, который содержит number (string) и экземпляр Location, status, который задан перечислением и принимает значения busy, free.
2. Создайте обобщенный класс Park, который содержит коллекцию параметризованного типа. Добавьте методы управления коллекцией: добавление, удаление, очистка. Добавьте метод поиска (Find), который принимает в качестве параметра функцию-предикат.
3. В Main создайте объект uber типа Park с параметром Taxi. В коллекцию добавьте 4 объекта Taxi с разным Location.
4. Отсортируйте такси (Taxi) по расстоянию в порядке возрастания к заданной пользователем координате.
5. Найдите ближайшее и запишите информацию о нем в файл.

Справочное: расстояние между точками считается как sqrt(((x2-x1)^2) + ((y2-y1)^2))

**class Location**

{

public int Lat { get; set; }

public int Long { get; set; }

public int Speed { get; set; }

public Location (int lat, int longe, int speed)

{

Lat = lat;

Long = longe;

Speed = speed;

}

**class Park<T> where T : Taxi**

{

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

public void tAdd(T taxi)

{

Taxis.Add(taxi);

}

public void tRemove(T taxi)

{

Taxis.Remove(taxi);

}

public void tRemoveAll(T taxi)

{

Taxis.Clear();

}

public object Find(Predicate<T> predicate)

{

foreach (T item in Taxis)

{

if (predicate(item))

return item.Number;

return null;

**enum Status**

{

busy,

free

}

**class Taxi**

{

public string Number { get; set; }

public Location Location { get; set; }

public Status Status { get; set; }

}

**static void Main(string[] args)**

{

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

Console.WriteLine("Enter the location");

double lat\_b = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter the location");

double long\_b = Convert.ToDouble(Console.ReadLine());

double[] range = new double[3];

Taxi car1 = new Taxi();

car1.Number = "8805553535";

Location loc1 = new Location(20, 25, 100);

car1.Location = loc1;

car1.Status = Status.free;

range[0] = Math.Sqrt(Math.Pow(car1.Location.Lat - lat\_b,2)+ Math.Pow(car1.Location.Long - long\_b, 2));

Taxi car2 = new Taxi();

car2.Number = "332281447";

Location loc2 = new Location(25, 30, 120);

car2.Location = loc2;

car2.Status = Status.busy;

range[1] = Math.Sqrt(Math.Pow(car2.Location.Lat - lat\_b, 2) + Math.Pow(car2.Location.Long - long\_b, 2));

Taxi car3 = new Taxi();

car3.Number = "291654914";

Location loc3 = new Location(0, 5, 100);

car3.Location = loc3;

car3.Status = Status.busy;

range[2] = Math.Sqrt(Math.Pow(car3.Location.Lat - lat\_b, 2) + Math.Pow(car3.Location.Long - long\_b, 2));

Array.Sort(range);

foreach (double i in range)

{

Console.WriteLine(i);

}

uber.tAdd(car1);

uber.tAdd(car2);

uber.tAdd(car3);

Console.WriteLine();

Predicate<Taxi> find = (Taxi tax) => { return tax.Status == Status.free; };

Console.WriteLine(uber.Find(find));

**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);

**The first**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Xml;

using System.Xml.Linq;

using System.Xml.Serialization;

using System.Text.Json;

using System.Threading.Tasks;

**interface Figure**

{

string Print(Rectangle t);

}

[Serializable]

**class Rectangle : Figure**

{

public int x { get; set; }

public int y { get; set; }

public int l { get; set; }

public int h { get; set; }

public string color { get; set; }

public Rectangle() { }

public Rectangle(int x, int y, string color)

{

this.x = x;

this.y = y;

this.color = color;

}

public Rectangle(int x, int y, int l, int h, string color) : this(x,y,color)

{

this.h = h;

this.l = l;

}

public static Rectangle operator +(Rectangle H, int i)

{

H.h = H.h + i;

H.h = H.l + i;

return H;

}

public int Sqr(Rectangle pl)

{

return pl.h \* pl.l;

}

public string Print(Rectangle t)

{

return t.x + " " + t.y + " " + t.l + " " + t.h + " " + t.color;

}

public string ToString()

{

return base.ToString() + " " + l + " " + y + " " + l;

}

}

**static async Task Main(string[] args)**

{

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

Rectangle rec1 = new Rectangle (22, 24, 22, 25, "синий" );

Rectangle rec2 = new Rectangle(12, 14, 12, 15, "ыыы");

Rectangle rec3 = new Rectangle(12, 14, 12, 15, "ааа");

Rectangle rec4 = new Rectangle(12, 14, 1, 1, "сий");

Rectangle rec5 = new Rectangle(12, 14, 12, 15, "сиий");

Rectangle rec6 = new Rectangle(12, 14, 12, 15, "ний");

listochek.Add(rec1);

listochek.Add(rec2);

listochek.Add(rec3);

listochek.Add(rec4);

listochek.Add(rec5);

listochek.Add(rec6);

rec2 = rec2 + 20;

Console.WriteLine(rec2.h + rec2.l);

/\*from i in listochek orderby i.h orderby i.l select i;\*/

var linq = listochek.OrderBy(t => t.h).OrderBy(t => t.l).Last();

//foreach (var item in linq)

//{

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

//}

Console.WriteLine(linq.ToString());

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

{

await JsonSerializer.SerializeAsync<Rectangle>(fs, rec1);

Console.WriteLine("Data has been saved to file");

}

* //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](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);  
    
  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](https://vk.com/away.php?to=http%3A%2F%2Fb.sale&cc_key=)+= (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](https://vk.com/away.php?to=http%3A%2F%2Fb.Sale&cc_key=)();  
  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](https://vk.com/away.php?to=http%3A%2F%2Fthis.name&cc_key=) = name;  
  [this.id](https://vk.com/away.php?to=http%3A%2F%2Fthis.id&cc_key=) = 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](https://vk.com/away.php?to=http%3A%2F%2Fthis.name&cc_key=) = nm;   
this.kurs = k;   
}   
public override string ToString()   
{   
return [this.name](https://vk.com/away.php?to=http%3A%2F%2Fthis.name&cc_key=) + 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](https://vk.com/away.php?to=http%3A%2F%2Fvector.Show&cc_key=)();   
vector.Del();   
[vector.Show](https://vk.com/away.php?to=http%3A%2F%2Fvector.Show&cc_key=)();   
vector.Add(queue1);   
vector.Clear();   
vector.Del();   
[vector.Show](https://vk.com/away.php?to=http%3A%2F%2Fvector.Show&cc_key=)();   
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](https://vk.com/away.php?to=http%3A%2F%2Fvector2.Show&cc_key=)(); }   
catch (Exception ex) {   
Console.WriteLine("Error: " + ex.Message); }   
finally {   
Console.WriteLine("done"); }   
Console.ReadKey();

создать класс **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](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);   
  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](https://vk.com/away.php?to=http%3A%2F%2FlistNode.Next&cc_key=);   
  }   
    
  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)}");   
  }   
  }   
  }

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);

}

}

}

}