* //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=" \t "_blank);   
  }   
    
  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);

}

}

}

}

}