Лабораторная работа 8 по С#

Интерфейсы.

студент, заочное, 40331, Василевский Евгений Валерьевич

**Задание 1**

Задание 1. Коллизия имен

На основании предметной области одной из предыдущих лабораторных работ (по выбору) спроектировать указанное наследование классов и интерфейсов. Каждый интерфейс должны содержать несколько методов или свойств. Должна присутствовать коллизия имен. Продемонстрировать все способы решения коллизии имен в классе, наследующем оба интерфейса • склеивание (комментируйте) • явное указание интерфейса (кастинг) (комментируйте) • переименование (кастинг + обертывание) (комментируйте)

using lab8\_задание\_1;

Console.WriteLine("Лабораторная 8. Задание 1");

Console.WriteLine();

Plane plane = new Plane("Boing");

Console.WriteLine(plane);

plane.Fly();

plane.Sound();

Console.WriteLine();

Console.WriteLine();

Auto auto = new Auto("Audi", 100);

Console.WriteLine(auto);

auto.Move();

auto.Sound();

Console.WriteLine();

Console.WriteLine();

Batmobile batmobile=new Batmobile("Dodge", 200, "Batman");

Console.WriteLine(batmobile);

batmobile.Move();

batmobile.Fly();

batmobile.Sound();

((IFlying)batmobile).Sound();

((IMoving)batmobile).Sound();

batmobile.FlyingSound();

batmobile.MovingSound();

Console.ReadKey();

namespace lab8\_задание\_1

{

internal class Auto:IMoving, IComparable<Auto>

{

private string name;

private int skorost;

public Auto():this("Noname", 0)

{

}

public Auto(string name, int skorost)

{

Name = name;

Skorost = skorost;

}

public string Name { get => name; set => name = value; }

public int Skorost

{

get => skorost;

set

{

if(value>0 && value <= 300) skorost = value;

else skorost = -1;

}

}

public int CompareTo(Auto? other)

{

return name.CompareTo(other.name);

}

public void Move()

{

Console.WriteLine("Машина едет по дороге");

}

public virtual void Sound()

{

Console.WriteLine("\*Звук обыкновенной машины\*");

}

public override string? ToString()

{

return $"Я машина марки {name} со скоростью {skorost}";

}

}

}

namespace lab8\_задание\_1

{

internal class Plane:IFlying

{

private string name;

public Plane():this("Noname")

{

}

public Plane(string name)

{

Name = name;

}

public string Name { get => name; set => name = value; }

public void Fly()

{

Console.WriteLine("Самолет летит в небе");

}

public void Sound()

{

Console.WriteLine("\*Звуки летящего самолета\*");

}

public override string? ToString()

{

return $"Я самолет по имени {name}";

}

}

}

namespace lab8\_задание\_1

{

internal class Batmobile:Auto,IFlying,IMoving

{

private string driver;

public Batmobile():this("Noname", 200, "Бэтмен")

{

}

public Batmobile(string name,int skorost,string driver):base(name, skorost)

{

Driver = driver;

}

public string Driver { get => driver; set => driver = value; }

public void Fly()

{

Console.WriteLine("Я бетмобиль - так что могу превращаться в самолет и летать");

}

public override void Sound() //склеивание

{

Console.WriteLine("\*Звуки автомобиля, заставляющие дрожать от страха всю преступность Готэма\*");

}

void IFlying.Sound() //кастинг

{

Console.WriteLine("\*Звуки летящего бетмобиля\*");

}

void IMoving.Sound() //кастинг

{

Console.WriteLine("\*Звуки мчащегося бетмобиля\*");

}

public void FlyingSound() //обертывание

{

((IFlying)this).Sound();

}

public void MovingSound() //обертывание

{

((IMoving)this).Sound();

}

public override string? ToString()

{

return $"Я бетмобиль, за рулем сидит {driver}";

}

}

}

namespace lab8\_задание\_1

{

internal interface IFlying

{

void Fly();

void Sound();

}

}

namespace lab8\_задание\_1

{

internal interface IMoving

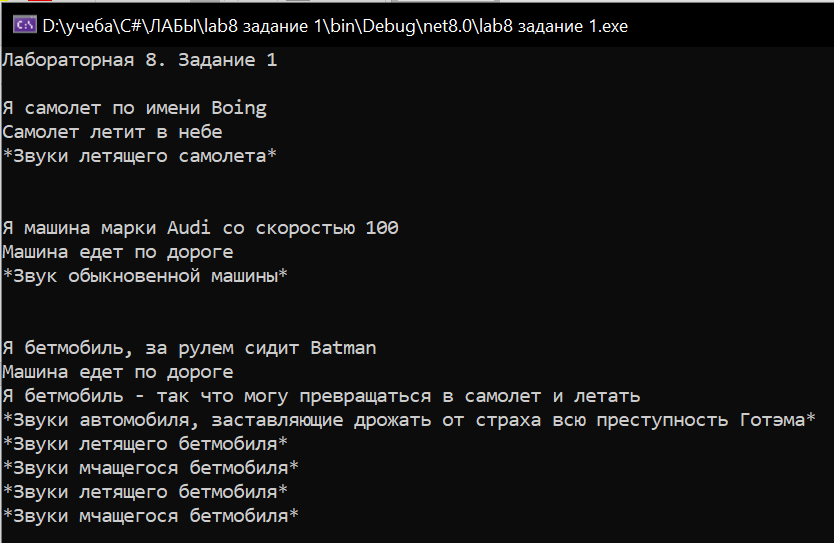
{

void Move();

void Sound();

}

}



**Задание 2**

Задание 2. Обобщенные интерфейсы. Задание выполняется с использованием двух ранее разработанных классов, связанных отношением наследования. Классы должны содержать перегруженный метод ToString(), дающий полную информацию об объекте класса. Также наследник должен содержать метод, которого нет у родителя. Объявите обобщенный ковариантный либо контрвариантный интерфейс (с использованием ключевых слов out или in). Интерфейс должен содержать 1 метод, который принимает либо возвращает тип-параметр. Создайте класс, реализующий этот интерфейс и параметризуйте его одним из ваших ранее созданных классов.

using lab8\_задание\_2;

using lab8\_задание\_1;

Console.WriteLine("Лабораторная работа 8. Задание 2");

Console.WriteLine();

ICovariant<Auto> garage1 = new Garage<Auto>();

Console.WriteLine(garage1.GetCar());

Console.WriteLine();

ICovariant<Auto> garage2= new Garage<Batmobile>();

Console.WriteLine(garage2.GetCar());

Console.WriteLine();

Console.WriteLine();

IContravariant<Batmobile> street1=new Street<Auto>();

street1.DrivesThroughTheStreets(new Batmobile());

Console.WriteLine();

IContravariant<Auto> street2 = new Street<Auto>();

street2.DrivesThroughTheStreets(new Batmobile("Dodge",250,"Batman"));

street2.DrivesThroughTheStreets(new Auto("BMW",150));

Console.ReadKey();

namespace lab8\_задание\_1

{

public class Auto:IMoving, IComparable<Auto>

{

private string name;

private int skorost;

public Auto():this("Noname", 0)

{

}

public Auto(string name, int skorost)

{

Name = name;

Skorost = skorost;

}

public string Name { get => name; set => name = value; }

public int Skorost

{

get => skorost;

set

{

if(value>0 && value <= 300) skorost = value;

else skorost = -1;

}

}

public int CompareTo(Auto? other)

{

return name.CompareTo(other.name);

}

public void Move()

{

Console.WriteLine("Машина едет по дороге");

}

public virtual void Sound()

{

Console.WriteLine("\*Звук обыкновенной машины\*");

}

public override string? ToString()

{

return $"Я машина марки {name} со скоростью {skorost}";

}

}

}

namespace lab8\_задание\_1

{

public class Batmobile:Auto,IFlying,IMoving

{

private string driver;

public Batmobile():this("Noname", 200, "Бэтмен")

{

}

public Batmobile(string name,int skorost,string driver):base(name, skorost)

{

Driver = driver;

}

public string Driver { get => driver; set => driver = value; }

public void Fly()

{

Console.WriteLine("Я бетмобиль - так что могу превращаться в самолет и летать");

}

public override void Sound() //склеивание

{

Console.WriteLine("\*Звуки автомобиля, заставляющие дрожать от страха всю преступность Готэма\*");

}

void IFlying.Sound() //кастинг

{

Console.WriteLine("\*Звуки летящего бетмобиля\*");

}

void IMoving.Sound() //кастинг

{

Console.WriteLine("\*Звуки мчащегося бетмобиля\*");

}

public void FlyingSound() //обертывание

{

((IFlying)this).Sound();

}

public void MovingSound() //обертывание

{

((IMoving)this).Sound();

}

public override string? ToString()

{

return $"Я бетмобиль, за рулем сидит {driver}";

}

}

}

namespace lab8\_задание\_2

{

internal interface IContravariant<in T>

{

void DrivesThroughTheStreets(T car);

}

}

namespace lab8\_задание\_2

{

internal interface ICovariant<out T>

{

T GetCar();

}

}

using lab8\_задание\_1;

namespace lab8\_задание\_2

{

internal class Street<T> : IContravariant<T> where T : Auto

{

public void DrivesThroughTheStreets(T car)

{

Console.WriteLine($"{car.Name} ездит по улицам");

}

}

}

using lab8\_задание\_1;

namespace lab8\_задание\_2

{

internal class Garage<T> : ICovariant<T> where T : Auto, new()

{

public T GetCar()

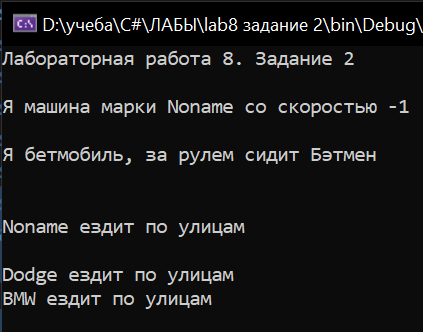
{

return new T();

}

}

}



**Задание 3**

Задание 3. Работа с массивом.

Создать массив типа Interface1. Заполнить массив объектами различных классов (Class1 и Class3). Организовать цикл по всем элементам массива. Для всех элементов вызвать метод ToString(). Для всех элементов, поддерживающих Interface2, вызвать метод из этого интерфейса.

using lab8\_задание\_1;

Console.WriteLine("Лабораторная работа 8. Задание 3");

Console.WriteLine();

Auto auto = new Auto("BMW",130);

Auto auto1 = new Auto("Audi", 100);

Batmobile batmobile = new Batmobile("Dodge", 200, "Batman");

Batmobile batmobile2 = new Batmobile("Audi", 180, "Robin");

IMoving[] move=new IMoving[4] { auto,auto1,batmobile,batmobile2};

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

{

Console.WriteLine(move[i]);

if (move[i] is IFlying)

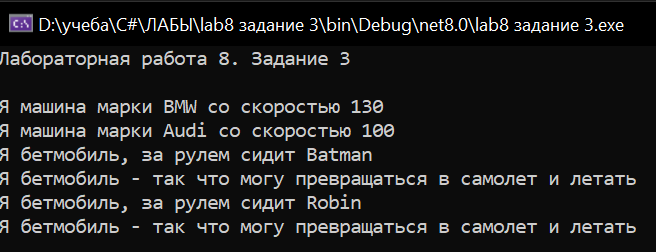
{

((IFlying)move[i]).Fly();

}

}

Console.ReadKey();



**Задание 4**

Задание 4. Использование стандартных интерфейсов.

Создать массив типа Class1. Организовать сортировку массива, используя стандартный метод Sort.Array и интерфейс IComparable. Ключ сортировки – поле name из родительского класса (Class1). Организовать сортировку, используя интерфейс IComparer. Создать 2 различных компаратора (по другим полям Class1), продемонстрировать работу.

using lab8\_задание\_1;

using lab8\_задание\_4;

Console.WriteLine("Лабораторная 8. Задание 4");

Auto auto = new Auto("BMW", 130);

Auto auto1 = new Auto("Audi", 100);

Batmobile batmobile = new Batmobile("Dodge", 200, "Batman");

Batmobile batmobile2 = new Batmobile("Audi", 180, "Robin");

Auto[] array = {auto,auto1,batmobile,batmobile2 };

foreach (Auto item in array)

{

Console.WriteLine(item);

}

Console.WriteLine();

Array.Sort(array);

foreach (Auto item in array)

{

Console.WriteLine(item);

}

Console.WriteLine();

Array.Sort(array, new ComparerAutoBySkorost());

foreach (Auto item in array)

{

Console.WriteLine(item);

}

Console.WriteLine();

Array.Sort(array, new ComparerAutoByName());

foreach (Auto item in array)

{

Console.WriteLine(item);

}

Console.WriteLine();

Console.ReadKey();

using lab8\_задание\_1;

namespace lab8\_задание\_4

{

internal class ComparerAutoByName : IComparer<Auto>

{

public int Compare(Auto? x, Auto? y)

{

return x.Name.CompareTo(y.Name);

}

}

}

namespace lab8\_задание\_4

{

internal class ComparerAutoBySkorost : IComparer<Auto>

{

public int Compare(Auto? x, Auto? y)

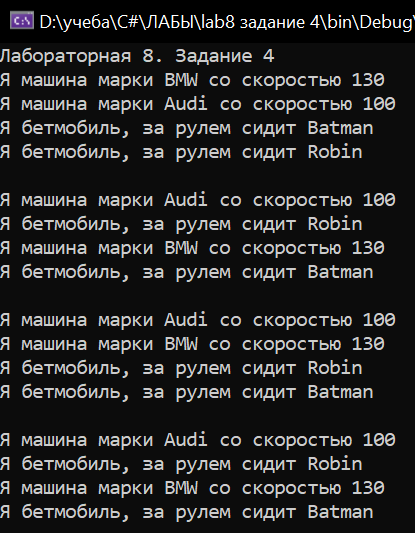
{

return x.Skorost.CompareTo(y.Skorost);

}

}

}



**Задание 5**

Задание 5. Именованные итераторы

Создать класс с именованным итератором, который принимает 2 аргумента – start и end. Согласно варианту задания использовать внутренний массив или алгоритм. Продемонстрировать работу итератора, используя цикл foreach.

//1) вывод названий месяцев от start до end

Iter iter=new Iter();

foreach(var item in iter.MyMonths(1,12))

{

Console.WriteLine(item) ;

}

namespace lab8\_задание\_5

{

internal class Iter

{

string[] months = {"янв","фев","мар", "апр", "май", "июн", "июл", "авг", "сен", "окт", "ноя", "дек" };

public IEnumerable MyMonths(int beg,int end)

{

for(int i = beg; i <= end; i++)

{

if (i < 1 || i > months.Length) break;

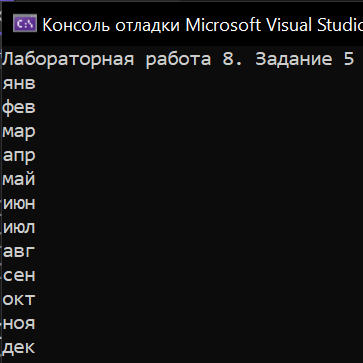
yield return months[i-1];

}

}

}

}



//2) вывод чисел Фибоначчи от start до end

Iter iter = new Iter(35);

foreach (var item in iter.MyFib(1, 29))

{

Console.WriteLine(item);

}

int end;

public Iter()

{

}

public Iter(int end)

{

this.end = end;

}

public IEnumerable MyFib(int beg,int end)

{

int x = 0; int y = 1;

while(y<= end)

{

if (y >= beg)

{

yield return y;

}

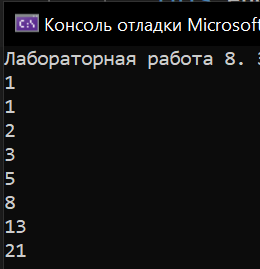
int next = x + y;

x = y;

y=next;

}

}



//3)вывод случайных букв в диапазоне от start до end без повтора

Iter iter=new Iter();

foreach(var item in iter.MyChars(5, 18))

{

Console.WriteLine(item);

}

int start;

int end;

Random random=new Random();

List<char> chars = new List<char>();

public Iter()

{

}

public Iter(int start, int end)

{

this.start = start;

this.end = end;

}

public IEnumerable MyChars(int start, int end)

{

for(int i = start; i < end; i++)

{

chars.Add((char)((int)'a'+i-1));

}

while(chars.Count > 0)

{

int index = random.Next(chars.Count);

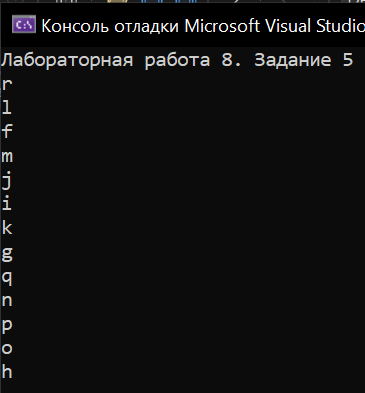
char char1 = chars[index];

chars.RemoveAt(index);

yield return char1;

}

}



4) вывод арифметической прогрессии от start до end с шагом 13

//4) вывод арифметической прогрессии от start до end с шагом 13

Iter iter=new Iter();

foreach(var item in iter.MyArif(5, 67))

{

Console.WriteLine(item);

}

int start;

int end;

public Iter()

{

}

public Iter(int start, int end)

{

this.start = start;

this.end = end;

}

public IEnumerable MyArif(int start, int end)

{

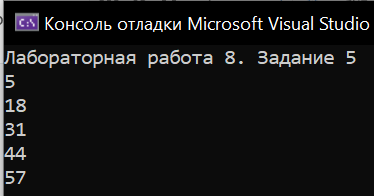
for(int i = start; i <= end; i+=13)

{

yield return i;

}

}



5) вывод геометрической прогрессии от start до end с шагом 2

Iter iter = new Iter();

foreach (var item in iter.MyGeom(3, 50))

{

Console.WriteLine(item);

}

int start;

int end;

public Iter()

{

}

public Iter(int start, int end)

{

this.start = start;

this.end = end;

}

public IEnumerable MyGeom(int start, int end)

{

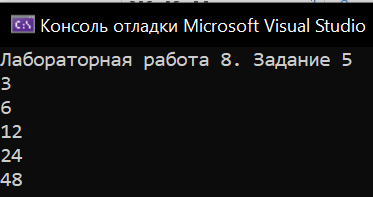
for (int i = start; i <= end; i \*= 2)

{

yield return i;

}

}



7) вывод квадратов чисел от start до end

Iter iter = new Iter();

foreach (var item in iter.MySqrt(2,7))

{

Console.WriteLine(item);

}

int start;

int end;

public Iter()

{

}

public Iter(int start, int end)

{

this.start = start;

this.end = end;

}

public IEnumerable MySqrt(int start, int end)

{

for (int i = start; i <= end; i++)

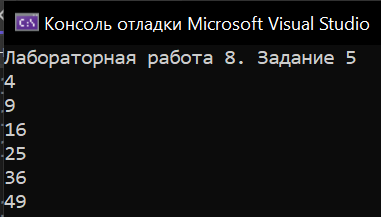
{

yield return i\*i;

}

}

}



8) вывод чисел, кратных 7 или 13 от start до end

Iter iter = new Iter();

foreach (var item in iter.MyKrat(2, 45))

{

Console.WriteLine(item);

}

int start;

int end;

public Iter()

{

}

public Iter(int start, int end)

{

this.start = start;

this.end = end;

}

public IEnumerable MyKrat(int start, int end)

{

for (int i = start; i <= end; i++)

{

if (i % 7 == 0 || i % 13 == 0)

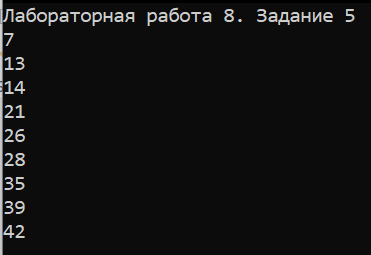
{

yield return i;

}

}

}



9) значения факториала от start не больше end

Iter iter = new Iter();

foreach (var item in iter.MyFact(2, 45))

{

Console.WriteLine(item);

}

int start;

int end;

public Iter()

{

}

public Iter(int start, int end)

{

this.start = start;

this.end = end;

}

public IEnumerable MyFact(int start, int end)

{

for(int i = 1; i < end; i++)

{

int p = 1;

for(int j = 1; j < i; j++)

{

p \*= j;

}

if (p >= start && p < end)

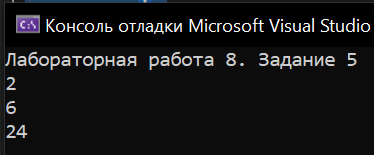
{

yield return p;

}

}

}



10) вывод високосных годов от start до end

Iter iter = new Iter();

foreach (var item in iter.MyVis(1890,2025))

{

Console.WriteLine(item);

}

int start;

int end;

public Iter()

{

}

public Iter(int start, int end)

{

this.start = start;

this.end = end;

}

public IEnumerable MyVis(int start, int end)

{

for (int i = start; i <= end; i++)

{

if (i % 4 == 0)

{

if (i % 100 == 0)

{

if (i % 400 != 0)

{

continue;

}

else yield return i;

}

else yield return i;

}

}

}

