# Лабораторная работа 5. Интерфейсы.

## Задание.

Внимание! Комментируйте все выполненные задания для демонстрации преподавателю.

1. **Проектирование**

На основании предметной области одной из предыдущих лабораторных работ (по выбору) спроектировать указанное наследование классов и интерфейсов.

Interface1

Class1

Interface2

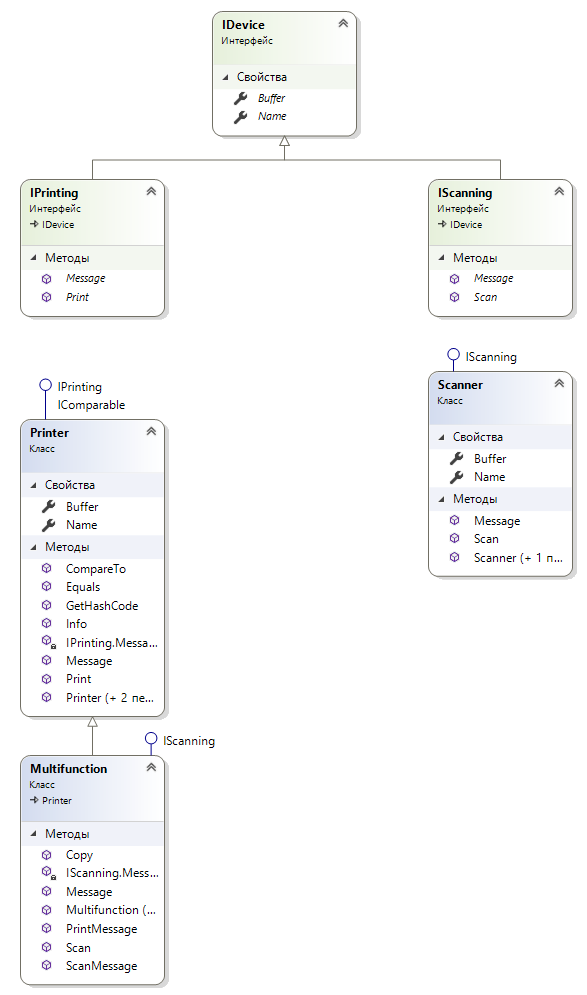
Class2

Class3

Каждый интерфейс должны содержать несколько методов или свойств.

Примеры методов:

* метод, который выводит на экран информацию об объекте класса (Print),
* метод, который каким-либо образом изменяет числовые поля класса (например, умножает на 2.7) и выводит поле (Mult),
* метод, который каким-либо образом изменяет текстовые поля класса (например, поднимает регистр) и выводит поле (Up).



**2. Коллизия имен**

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)

**3. Работа с массивом**

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

**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.

class ComparerByPrinterName:IComparer<Printer>

{

public int Compare(Printer p1, Printer p2)

{

return p1.Name.CompareTo(p2.Name);

}

}

class ComparerByPrinterBuffer:IComparer<Printer>

{

public int Compare(Printer p1, Printer p2)

{

return p1.Buffer.CompareTo(p2.Buffer);

}

}

**5. создать класс с именованным итератором, который принимает 2 аргумента – start и end.**

1. вывод названий месяцев от start до end
2. вывод чисел Фибоначчи от start до end
3. вывод случайных букв в диапазоне от start до end без повтора
4. вывод арифметической прогрессии от start до end с шагом 13
5. вывод геометрической прогрессии от start до end с шагом 2
6. вывод названий дней недели от start до end
7. вывод квадратов чисел от start до end
8. вывод чисел, кратных 7 или 13 от start до end
9. значения факториала от start не больше end
10. вывод високосных годов от start до end

// именованный итератор (5 задание). Арифмитическая прогрессия с шагом 13

static public IEnumerable MyIterator(int start,int end)

{

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

{

yield return (i-1) \* 13;

}

}

/////////////////////////////////////////////////////////

public interface IDevice

{

string Name { get; set; }

string Buffer { get; set; }

}

public interface IScanning : IDevice

{

void Scan(string str);

void Message();

}

public class Scanner : IScanning

{

public string Name { get; set; }

public string Buffer { get; set; }

public Scanner() : this("noname") { }

public Scanner(string name)

{

Name = name;

}

public void Scan(string str)

{

Console.WriteLine($"сканер {Name} получил команду на сканирование.");

Buffer = str;

}

public void Message()

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("Message:страница успешно отсканирована!\n");

Console.ForegroundColor = ConsoleColor.White;

}

}

public interface IPrinting : IDevice

{

void Print();

void Message();

}

public class Printer: IPrinting, IComparable

{

public string Name { get; set; }

public string Buffer { get; set; }

public Printer() : this("noname", "") { }

public Printer(string name) : this(name, "") { }

public Printer(string name, string buffer)

{

Name = name;

Buffer = buffer;

}

public void Print()

{

Console.WriteLine($"На принтер {Name} отправлена страница:\n\t\"{Buffer}\"");

}

public virtual void Message()

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine("Message: страница успешно напечатана!\n");

Console.ForegroundColor = ConsoleColor.White;

}

// явное указание интерфейса (кастинг)

void IPrinting.Message()

{

Console.WriteLine("(кастинг)Message: страница успешно напечатана!\n");

}

public int CompareTo(object o)

{

return this.Name.CompareTo(((Printer)o).Name);

}

public override bool Equals(object obj) // перегружаем для использования в лабе №6

{

return (this.Name == ((obj as Printer)?.Name ?? ""));

}

public override int GetHashCode()

{

return base.GetHashCode();

}

public void Info()

{

Console.WriteLine($"Название: {Name}; содержимое буфера:{Buffer}");

}

}

public class Multifunction : Printer, IScanning

{

public Multifunction() : this("noname") { }

public Multifunction(string name) : base(name) { }

public Multifunction(string name, string buffer) : base(name)

{

Buffer = buffer;

}

public void Scan(string str)

{

Console.WriteLine("Сканирование...");

Buffer = str;

}

public void Copy(string str)

{

Console.WriteLine("Получено задание на копирование...");

Scan(str);

((IScanning)this).Message();

Print();

((IPrinting)this).Message();

}

public override void Message() // Склеивание. перегружаем метод родительского класса Printer

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("(склеивание)Message: Копия готова!\n");

Console.ForegroundColor = ConsoleColor.White;

}

// явное указание интерфейса (кастинг)

void IScanning.Message() {

Console.WriteLine("(кастинг)Message: страница отсканирована!\n");

}

// применим обертывание

public void ScanMessage()

{

((IScanning)this).Message();

}

public void PrintMessage()

{

((IPrinting)this).Message();

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("2. Коллизия имен\n");

Scanner scanjet = new Scanner("HPScanJet\_2400");

scanjet.Scan("В целом, конечно, консультация с широким активом однозначно...");

scanjet.Message();

Console.WriteLine("Cодержимое буфера:\n" + scanjet.Buffer + "\n");

Console.WriteLine("\*\*\*");

Printer hplaserjet = new Printer("HPLaserJet\_1160");

hplaserjet.Buffer = "Hello world";

hplaserjet.Print();

hplaserjet.Message();

Console.WriteLine("вызовем явно реализованный метод интерфейса IPrinting");

((IPrinting)hplaserjet).Message();

Console.WriteLine("\*\*\*");

Multifunction canon410 = new Multifunction("CanonMF410");

canon410.Copy("Lorem ipsum dolor sit amet, consectetur adipiscing elit...");

canon410.Message();

Console.WriteLine("Вызовем переименованные методы (кастинг + обертывание):");

canon410.ScanMessage();

canon410.PrintMessage();

Console.WriteLine("\*\*\*");

Console.WriteLine("3. Работа с массивом");

//инициализируем массив элементов типа IPrinter

IPrinting[] devices = new IPrinting[]

{

new Printer("Kyocera","qwerty"),

new Multifunction("Xerox", "yesterday"),

new Printer("Samsung","good morning"),

new Printer("Epson","bye"),

new Multifunction("Canon", "togetho")

};

Console.WriteLine("для всех элементов, поддерживающих интерфейс IScanning вызовем метод Scan()\n");

foreach (IPrinting device in devices)

if (device is IScanning)

{

IScanning scanner = (IScanning)device;

scanner.Scan("Hello, I can scan");

Console.WriteLine($"Содержимое буффера устройства {scanner.Name}:" + scanner.Buffer);

scanner.Message();

}

Console.WriteLine("4. Использование стандартных интерфейсов");

Console.WriteLine("\nМассив до сортировки:");

foreach (IPrinting dev in devices)

((Printer)dev).Info();

Array.Sort(devices); // отсортируем наш массив по имени

Console.WriteLine("\nМассив после сортировки:");

foreach (IPrinting dev in devices)

((Printer)dev).Info();

Printer[] printers = new Printer[5];

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

printers[i] = (Printer)devices[i];

Console.WriteLine("\nОтсортируем используя компараторы.");

Array.Sort(printers, new ComparerByPrinterBuffer());

Console.WriteLine("\nМассив после сортировки по содержимому буфера:");

foreach (Printer printer in printers)

printer.Info();

Array.Sort(printers,new ComparerByPrinterName()); // отсортируем наш массив по имени

Console.WriteLine("\nМассив после сортировки по имени:");

foreach (Printer printer in printers)

printer.Info();

Console.WriteLine("\n 5. создать класс с именованным итератором, который принимает 2 аргумента – start и end.");

Console.WriteLine("Вариант 4. Вывод арифмитической прогрессии от start до end с шагом 13.");

Console.WriteLine("Введите пределы.");

Console.WriteLine("start = ");

int start = int.Parse(Console.ReadLine());

Console.WriteLine("end = ");

int end = int.Parse(Console.ReadLine());

Console.WriteLine("Прогрессия:");

foreach (int a in MyIterator(start, end))

{

Console.WriteLine("\n"+ a.ToString());

}

Console.ReadKey();

}

}

