Лабораторная работа №11

**Тема:** Разработка программ с использованием интерфейсов.

**Цель:** Научиться разрабатывать интерфейсы, и использовать их в программах, применять механизм множественного наследования интерфейсов.

**Техническое оснащение:** MS Windows 10, MS Word 2019, MS VS 2019, MS Visio 2019.

Выполнение работы

Вариант 18

**Пример 1.** Пусть имеется два вида объектов: игроки в мяч и те, кто умеет играть на гитаре. И те, и другие играют, но, естественно, делают это по-разному. Создадим интерфейс, в котором дадим описание метода, характеризующее это действие, т.е. игру, а так же классы, базирующиеся на этом интерфейсе. И именно в этих классах дадим необходимую реализацию.

Проверка работы примера (рисунок 1).

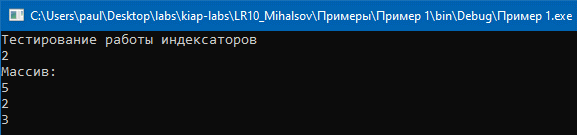


Рисунок 1 – Выполнение примера 1

**Пример 2.** Разовьём пример 1. Создадим для классов Ball и Gitara общего наследника — класс Man, при этом наши классы по-прежнему будут наследовать и от интерфейса IPlayer.

Проверка работы примера (рисунок 2).

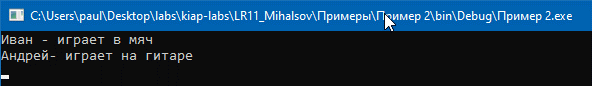


Рисунок 2 – Выполнение примера 2

**Пример 3.** Создадим класс Student, который будет наследовать от класса Man и интерфейсов IBall и IGitara (классы Ball и Gitara переделаем в интерфейсы). В этих интерфейсах определим одинаковый метод play(). В общем, наделим объекты типа Student способностью играть в мяч и играть на гитаре (при этом наш «студент» ещё и учится(!!) — у класса Student есть собственный метод study(), отвечающий за этот процесс).

Проверка работы примера (рисунок 3).

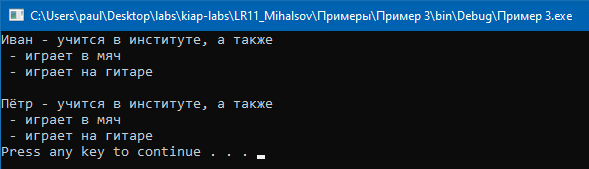


Рисунок 3 – Выполнение примера 3

**Пример 4**. Пусть теперь наш «студент» перевёлся на вечернее отделение. Класс Student преобразуем в интерфейс IStudent, который будет по-прежнему наследовать от интерфейсов IBall и IGitara. Основным классом станет класс Worker, который будет наследовать от класса Man и интерфейса IStudent.

Проверка работы примера (рисунок 4).

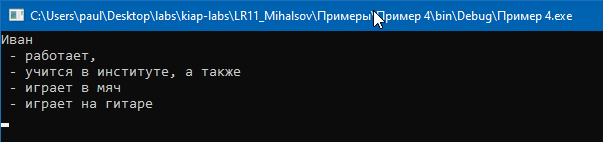


Рисунок 4 – Выполнение примера 4

**Пример 5.** Пусть наш «студент-вечерник» из примера 4 стал играть в КВН. Для реализации такой возможности добавим в интерфейс IStudent метод play(), означающий игру в КВН.

Проверка работы примера (рисунок 5).

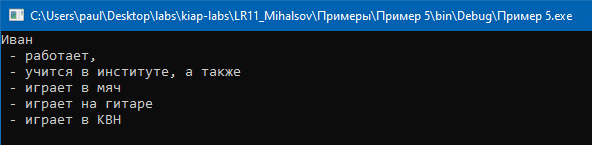


Рисунок 5 – Выполнение примера 5

**Задание.** Построить иерархию классов: рабочий, кадры, инженер, администрация.

Листинг программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Задание

{

class Program

{

interface IPersona

{

string getName();

string getId();

}

interface IWorker

{

string getWorkRang();

string getActivity();

}

interface IEngineer

{

string getActivityProject();

string getSecrecyLevel();

}

interface IAdmin

{

string getDeportament();

string getadminLevel();

}

class Persona : IPersona

{

string id;

string name;

public string getId()

{

return id;

}

public string getName()

{

return name;

}

public Persona(string name, string id)

{

this.id = id;

this.name = name;

}

public Persona() { }

}

class Worker : Persona, IWorker

{

public string getActivity()

{

return activity;

}

public string getWorkRang()

{

return workrang;

}

string activity;

string workrang;

public Worker(string name, string id) : base(name, id)

{

activity = "nowork";

workrang = "C9";

}

public void setRang(string rang)

{

workrang = rang;

}

public void setActivity(string activity)

{

this.activity = activity;

}

}

class Engener : Worker, IEngineer

{

public string getActivityProject()

{

return activityproject;

}

public string getSecrecyLevel()

{

return secrecy;

}

string secrecy;

string activityproject;

public Engener(string name, string id, string secrecy, string activityproject) : base(name, id)

{

this.secrecy = secrecy;

this.activityproject = activityproject;

}

}

class Admin : Persona

{

string deportament;

string adminlevel;

public Admin(string name, string id, string adminlevel) : base(name, id)

{

this.adminlevel = adminlevel;

}

public void setDeportament(string deportament)

{

this.deportament = deportament;

}

}

static int lastid = -1;

public static string getNewId()

{

lastid++;

return lastid.ToString();

}

static void Main(string[] args)

{

List<Admin> Admins = new List<Admin>

{

new Admin("Артёмов К.А",getNewId(),"S0"),

new Admin("Андреев Г.А",getNewId(),"S1"),

new Admin("Кучинский Д.С",getNewId(),"S1"),

};

List<Engener> Engeners = new List<Engener>

{

new Engener("Кукуш З.С",getNewId(),"S3","KOLIMA3.0"),

new Engener("Генадьев Т.З",getNewId(),"S3","KOLIMA4.0"),

};

List<Worker> workers = new List<Worker>

{

new Worker("Мякишь П.А.",getNewId()),

new Worker("Масляков Н.А.",getNewId()),

new Worker("Цаль В.И.",getNewId()),

new Worker("Цой В.Р.",getNewId()),

};

List<Persona> deportam = new List<Persona>();

foreach (var item in Admins)

{

item.setDeportament("депортамент 1");

deportam.Add(item);

}

foreach (var item in Engeners)

{

item.setActivity("наботает над проектом " + item.getActivityProject());

deportam.Add(item);

}

string[] works = new string[] { "слесарь", "сварщик", "бригадир", "станочник"};

Random rnd = new Random();

foreach (var item in workers)

{

item.setRang("B1");

item.setActivity("работает по спецальности " + works[rnd.Next(0,works.Length)]);

deportam.Add(item);

}

List<Worker> workersandengeners = new List<Worker>();

foreach (var item in workers)

workersandengeners.Add(item);

foreach (var item in Engeners)

workersandengeners.Add(item);

Console.WriteLine("Отдел:");

foreach (var item in deportam)

Console.WriteLine("Списочный номер {0}, имя {1}", item.getId(), item.getName());

Console.WriteLine("\nДеятельность рабочих и инженеров:");

foreach (var item in workersandengeners)

{

Console.WriteLine("Номер {0}, {1}", item.getId(), item.getActivity());

}

Console.ReadKey(true);

}

}

}

Проверка работы программы (рисунок 6).

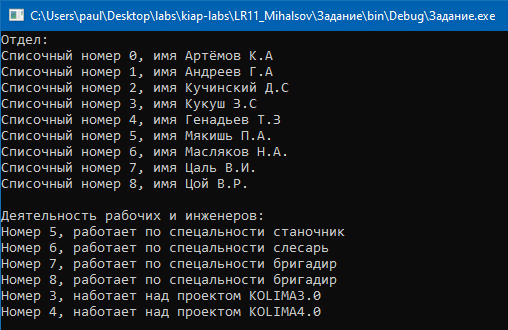


Рисунок 6 – Проверка работы программы

**Вывод:** в результате выполнения лабораторной работы были получены практические навыки по реализации собственных интерфейсов, а также укреплены навыки в использовании механизма наследования.