Лабораторная работа №11

Тема: Разработка программ с использованием интерфейсов.

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

Теоретические сведения:

Интерфейс — это особый вид класса. Если класс служит для создания объектов каких-либо типов, наделённых неким набором возможностей, то интерфейс предназначен для других целей.

Описание интерфейса во многом напоминает описание класса, но вместо ключевого слова **class** используется слово **interface**. По-умолчанию всякий интерфейс являются доступным в сборке (можно использовать слово **internal**), а можно задать общий доступ (**public**).

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.

Возможная реализация программы:

```
using System;
namespace Prim_Interface1
{
interface IPlayer
{
void play();
}
public class Ball: IPlayer
{
public void play()
{
Console.WriteLine("Игра в мяч");
}
}
public class Gitara: IPlayer
{
public void play()
```

```
{
Console.WriteLine("Игра на гитаре");
}
}
class Program
{
public static void Main(string[] args)
{
Console.WriteLine("Тест одного интерфейса для двух классов");
Ball x = new Ball();
x.play();
Gitara y = new Gitara();
y.play();
Console.Write("Press any key to continue . . . ");
Console.ReadKey(true);
}
}
}
```

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**.

Возможная реализация программы:

```
using System;
namespace Prim_Interface2
{
interface IPlayer
{
void play();
}
public class Man
{
string name;
public Man()
        {
name = "NoName";
}
public Man(string n)
```

```
{
name = n;
}
public void print()
{
Console.Write(name);
}
}
public class Ball: Man, IPlayer
{
public Ball(): base()
{ }
public Ball(string n): base(n)
{ }
public void play()
{
Console.WriteLine(" - играет в мяч");
}
}
public class Gitara: Man, IPlayer
{
public Gitara(): base()
{ }
public Gitara(string n): base(n)
{ }
public void play()
{
Console.WriteLine("- играет на гитаре");
}
}
class Program
{
public static void Main(string[] args)
{
//Наследование от одного класса и одного интерфейса для двух классов
Ball x = new Ball("Иван");
x.print();
```

```
x.play();
Gitara y = new Gitara("Андрей");
y.print();
y.play();
Console.Write("Press any key to continue . . . ");
Console.ReadKey(true);
}
}
}
```

Результат работы программы:

```
Иван - играет в мяч
Андрей- играет на гитаре
Press any key to continue . . .
```

Рисунок 1 – Выполнение программы

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).

Возможная реализация программы:

```
using System;
namespace Prim_Interface3
{
public class Man
{
string name;
public Man()
{
name = "NoName";
}
public Man(string n)
{
name = n;
}
```

```
public void print()
{
Console.Write(name);
}
}
interface IBall
{
void play();
}
interface IGitara
{
void play();
}
        public class Student: Man, IBall, IGitara
{
public Student(): base()
{ }
public Student(string n): base(n)
{ }
void IBall.play()
{
Console.WriteLine(" - играет в мяч");
}
void IGitara.play()
{
Console.WriteLine(" - играет на гитаре");
}
public void study()
{
Console.WriteLine(" - учится в институте, а
также");
}
}
class Program
{
public static void Main(string[] args)
{
```

```
// Наследование от класса и двух интерфейсом, имеющих метод с одинаковым именем
Student x = new Student("Иван");
x.print();
x.study();
((IBall)x).play();
((IGitara)x).play();
Console.WriteLine();
Student y = new Student("Пётр");
y.print();
y.study();
IBall i1 = y;
i1.play();
IGitara i2 = y;
i2.play();
Console.Write("Press any key to continue . . . ");
Console.ReadKey(true);
}
}
}
```

Результат работы программы:

```
Иван - учится в институте, а также
- играет в мяч
- играет на гитаре

Пётр - учится в институте, а также
- играет в мяч

 - играет на гитаре
 Press any key to continue . . .
```

Рисунок 2 – Выполнение программы

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**.

Возможная реализация программы:

```
        using System;
namespace Prim_Interface4
{
public class Man
{
string name;
public Man()
{
name = "NoName";
}
public Man(string n)
{
name = n;
}
public void print()
{
Console.Write(name);
}
}
interface IBall
{
void play();
}
interface IGitara
{
void play();
}
interface IStudent: IBall, IGitara
{
void study();
}
public class Worker: Man, IStudent
{
public Worker(): base()
{ }
public Worker(string n): base(n)
```

```
{ }
void IBall.play()
        {
Console.WriteLine(" - играет в мяч");
}
void IGitara.play()
{
Console.WriteLine(" - играет на гитаре");
}
public void study()
{
Console.WriteLine(" - учится в институте, а
также");
}
public void work()
{
Console.WriteLine("\n - работает,");
}
}
class Program
{
public static void Main(string[] args)
{
Worker x = new Worker("Иван");
x.print();
x.work();
x.study();
((IBall)x).play();
((IGitara)x).play();
Console.Write("Press any key to continue . . . ");
Console.ReadKey(true);
}
}
}
```

Результат работы программы:

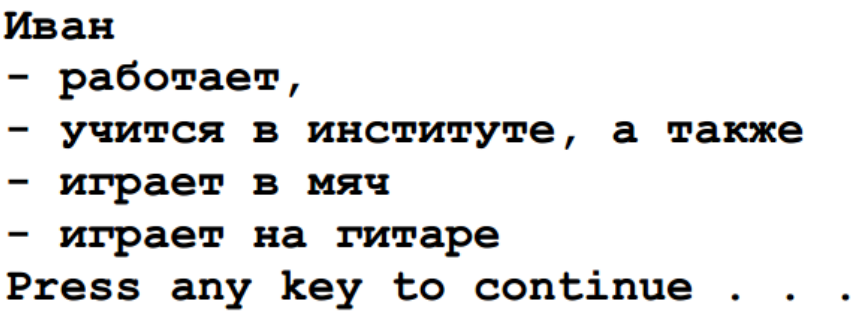

Рисунок 4 – Выполнение программы

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.

Возможная реализация программы:

```
using System;
namespace Prim_Interface5
{
public class Man
{
string name;
public Man()
{
name = "NoName";
}
public Man(string n)
{
name = n;
}
public void print()
{
Console.Write(name);
}
}
interface IBall
{
void play();
}
interface IGitara
{
```

```
void play();
}
interface IStudent: IBall, IGitara
{
void study();
new void play();
}
public class Worker: Man, IStudent
{
public Worker(): base()
{ }
public Worker(string n): base(n)
{ }
void IBall.play()
{
Console.WriteLine(" - играет в мяч");
}
void IGitara.play()
{
Console.WriteLine(" - играет на гитаре");
}
public void study()
{
Console.WriteLine(" - учится в институте, а
также");
}
void IStudent.play()
{
        Console.WriteLine(" - играет в КВН");
}
public void work()
{
Console.WriteLine("\n - работает,");
}
}
class Program
{
```

```
public static void Main(string[] args)
{
Worker x = new Worker("Иван");
x.print();
x.work();
x.study();
((IBall)x).play();
((IGitara)x).play();
((IStudent)x).play();
Console.Write("Press any key to continue . .
. ");
Console.ReadKey(true);
}
}
}
```

Результат работы программы:

```
Иван
- работает,
- учится в институте, а также
- играет в мяч
- играет на гитаре
- играет в КВН
Press any key to continue . . .
```

Рисунок 4 – Выполнение программы

Выполнение работы:

Вариант задания выбирается по порядковому номеру студента в списке подгруппы.

Построить иерархию классов в соответствии с вариантом задания:

1) Студент, преподаватель, персона, заведующий кафедрой
2) Служащий, персона, рабочий, инженер
3) Рабочий, кадры, инженер, администрация
4) Деталь, механизм, изделие, узел
5) Организация, страховая компания, нефтегазовая компания, завод
6) Журнал, книга, печатное издание, учебник
7) Тест, экзамен, выпускной экзамен, испытание
8) Место, область, город, мегаполис
9) Игрушка, продукт, товар, молочный продукт

10) Квитанция, накладная, документ, счет

11) Автомобиль, поезд, транспортное средство, экспресс

12) Двигатель, двигатель внутреннего сгорания, дизель, реактивный двигатель

13) Республика, монархия, королевство, государство

14) Млекопитающее, парнокопытное, птица, животное

15) Корабль, пароход, парусник, корвет

Шкала оценивания индивидуальных заданий

Примеры	1-5 балла
Примеры + Индивидуальное задание (наследование классов без использования интерфейсов)	1-8 баллов
Примеры + Индивидуальное задание (использование интерфейсов)	1-10 баллов

Содержание отчета:

1. Номер и тема лабораторной работы.
2. Цель лабораторной работы.
3. Техническое оснащение.
4. Скриншоты выполнения примеров
5. При выполнении индивидуальных заданий в отчет внести изображение кода программы и окно выполнения программы.
6. Вывод по лабораторной работе