Лабораторная работа №3

Цель работы:

Познакомиться с использованием полиморфных методов при объектно-ориентированном подходе, научиться создавать и реализовывать интерфейсы в языке программирования C#.

1) Для иерархии классов, построенной в лабораторной работе №2, выполнить замену абстрактного базового класса на класс, содержащий виртуальные функции.

2) Выполнить переопределение методов в классах-потомках. Показать пример использования полиморфного вызова методов (можно использовать любые типы коллекций для объектов заданных классов).

3) Создать как минимум два интерфейса и выполнить их реализацию в заданной иерархии.

Необходимые теоретические сведения

Полиморфизм

Полиморфизм — одна из ключевых парадигм ООП, позволяющая определять в наследуемом классе методы, которые будут общими для всех наследующих классов, при этом наследующий класс может определять специфическую реализацию некоторых или всех этих методов. Главный принцип полиморфизма: «один интерфейс, несколько методов». Благодаря ему, можно пользоваться методами, не обладая точными знаниями о типе объектов.

Основным инструментом для реализации принципа полиморфизма является использование виртуальных методы и абстрактных классов.

Виртуальные методы

Метод, при определении которого в наследуемом классе было указано ключевое слово **virtual**, и который был переопределен в одном или более наследующих классах, называется виртуальным методом. Следовательно, каждый наследующий класс может иметь собственную версию виртуального метода.

Выбор версии виртуального метода, которую требуется вызвать, осуществляется в соответствии с типом объекта, на который ссылается ссылочная переменная, во время выполнения программы. Другими словами, именно тип объекта, на который указывает ссылка (а не тип ссылочной переменной), определяет вызываемую версию виртуального метода. Таким образом, если класс содержит виртуальный метод и от этого класса были наследованы другие классы, в которых определены свои версии метода, при ссылке переменной типа наследуемого класса на различные типы объектов вызываются различные версии виртуального метода.

При определении виртуального метода в составе наследуемого класса перед типом возвращаемого значения указывается ключевое слово **virtual**, а при переопределении виртуального метода в наследующем классе используется модификатор **override**. Виртуальный метод не может быть определен с модификатором **static** или **abstract**.

Переопределять виртуальный метод не обязательно. Если наследующий класс не предоставляет собственную версию виртуального метода, то используется метод наследуемого класса.

Переопределение метода положено в основу концепции динамического выбора вызываемого метода — выбора вызываемого переопределенного метода осуществляется во время выполнения программы, а не во время компиляции.

Синтаксис:

```
virtual тип имя(список_параметров)
{
        тело_метода
};
```

Интерфейсы

В C# для полного отделения структуры класса от его реализации используется механизм интерфейсов.

Интерфейс является расширением идеи абстрактных классов и методов. Синтаксис интерфейсов подобен синтаксису абстрактных классов.

Объявление интерфейсов выполняется с помощью ключевого слова **interface**. При этом методы в интерфейсе не поддерживают реализацию.

Членами интерфейса могут быть методы, свойства, индексаторы и события.

Интерфейс может реализовываться произвольным количеством классов. Один класс, в свою очередь, может реализовывать любое число интерфейсов. Каждый класс, включающий интерфейс, должен реализовывать его методы. В интерфейсе для методов неявным образом задается тип **public**. В этом случае также не допускается явный спецификатор доступа.

Синтаксис:

```
[атрибуты] [модификаторы] interface Имя_интерфейса
[:список_родительских_интерфейсов]
{
        обьявление_свойств_и_методов
}
```

Пример:

```
interface Species
{
        string getSpecies();
        void Feed();
}

class Cheetah : Animal, Species
{
        private string scientificName;

        //реализация методов интерфейса
        public string getSpecies()
        {
                return scientificName;
        }

        public void Feed()
        {
                weight++;
        }
}
```

Можно объявлять ссылочную переменную, имеющую интерфейсный тип. Подобная переменная может ссылаться на любой объект, который реализует ее интерфейс. При вызове метода объекта с помощью интерфейсной ссылки вызывается версия метода, реализуемого данным объектом.

Возможно наследование интерфейсов. В этом случае используется синтаксис, аналогичный наследованию классов. Если класс реализует интерфейс, который наследует другой интерфейс, должна обеспечиваться реализация для всех членов, определенных в составе цепи наследования интерфейсов.