Лабораторная работа №3

**Цель работы:**

Познакомиться с программированием полиморфных методов при объектно-ориентированном подходе при использовании языка C#.

**Необходимые теоретические сведения**

**Полиморфизм**

Полиморфизм – одна из основных составляющих объектно-ориентированного программирования, 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};

**Абстрактные классы**

В абстрактном классе определяются лишь общие предназначения методов, которые должны быть реализованы в наследующих классах, но сам по себе этот класс не реализует один, или несколько подобных методов, называемых абстрактными (для них определены только некоторые характеристики, такие как тип возвращаемого значения, имя и список параметров).

При объявлении абстрактного метода используется модификатор abstract. Абстрактный метод автоматически становится виртуальным, так что модификатор virtual при объявлении метода не используется. Абстрактный класс предназначен только для создания иерархии классов, нельзя создать объект абстрактного класса.

Пример:

```
abstract class Animal
{
    public string Name;
    protected int Weight;
    private int Type;
    abstract void Feed();
    public int Animal(int W, int T, string N)
    {
        Weight = W;
        Type = T;
        Name = N;
    }
    public int GetWeight() { return Weight; }
}
class Predator : Animal
{
    private int Speed;
    override void Feed(int Food)
    {
        Weight += Food;
    }
}
```

**Задание**

Написать C# программу на основе лабораторной №1,2 реализующую работу с классами по вариантам.

Каждый из классов должен удовлетворять следующим требованиям:

1. Создать корневой абстрактный класс, который будет содержать, общие поля для всех классов, а также виртуальные методы ShortDescription – «Краткое описание» и ItemType (текстовое название класса).
2. Реализовать прямое или косвенное наследование всех классов от абстрактного класса и класса Object.
3. В каждом классе переопределить методы ShortDescription, ToString, Equal.
4. Проверить возможность сравнивать объекты одинаковых классов на равенство.

Разработанная программа должна быть консольной, позволяющей на выбор добавлять объекты классов, при наличии точно такого же экземпляра сообщать об этом пользователю. Кроме того должна быть возможность просмотреть все

добавленные объекты заданного класса. Все экземпляры классов должны храниться в одном массиве.

В отчёт обязательно построить диаграмму классов, описывающую иерархию созданных классов.

**Варианты**:

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