**Практическая работа №8**

**Создание наследованных классов.**

**Цель:** Формирование навыков программирования C# и работы с системой MS Visual Studio,реализации наследования.

### Теоретические сведения:

**Наследование** (inheritance) – это процесс приобретения состояния и поведения одного класса (называемого **базовым** или **предком**) другим классом (называемым **производным, наследником** или **потомком**). Для любого класса, кроме бесплодного (модификатор **sealed),** можно задать классы-наследники, в которых повторяется и дополняется состояние и поведение предка.

Синтаксис объявления производного класса (потомка):

[ модификаторы ] **class Имя : класс-предок, интерфейсы…**

**{ тело класса }**

Класс в C# может иметь произвольное количество потомков и только один класс-предок. В то же время, класс может наследоваться от произвольного количества интерфейсов. Наследование позволяет многократно использовать программный код, исключать из программы повторяющиеся фрагменты; упрощает модификацию программ и создание новых классов на основе существующих. Благодаря наследованию можно, например, использовать объекты, исходный код которых недоступен, но в поведение которых требуется внести изменения.

Наследование позволяет строить иерархии объектов. Они представляется в виде деревьев, в которых более общие объекты (предки) располагаются ближе к корню, а более специализированные (потомки) – на ветвях и листьях (рис. 2.6).

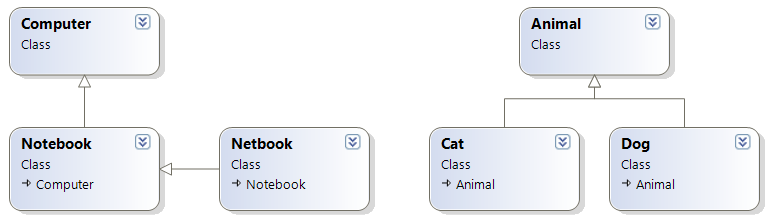


Рис. 2.6. Иерархии классов в окне Схема классов

Фрагменты программного кода для этих иерархий:

class **Computer { ... }** // базовый класс class **Animal { ... }**

class **Notebook : Computer** // потомки class **Cat : Animal**

**{ ... } { ... }**

class **Netbook : Notebook** class **Dog : Animal**

**{ ... } { ... }**

Возможны различные стратегии классического наследования:

* функционал потомка остается неизменным;
* функционал методов базового класса скрывается и замещается в производном класса (модификатор **new**);
* функционал методов базового класса, называемых **виртуальными** (модификатор **virtual**), переопределяется в производном классе (модификатор **override**).

Отметим важные особенности классического наследования:

* Наследуются поля, методы и свойства класса.
* Конструкторы не наследуются! Класс – потомок должен иметь собственные конструкторы.
* Объекту базового класса можно присвоить объект производного, например: public class **Notebook : Computer { … }**

**Computer comp = new Notebook( … );**

Таким образом, методы, которые у потомков должны реализовываться по- разному, при описании базовых классов следует определять виртуальными. Если во всех классах иерархии метод будет выполняться одинаково, его лучше определить как обычный метод. Виртуальные методы базового класса задают поведение всей иерархии, которое может изменяться и дополняться в потомках за счет добавления новых виртуальных методов. С помощью виртуальных методов реализуется один из основных принципов ООП – полиморфизм.

Отметим, что элементы базового класса с модификатором **private** в классе- наследнике недоступны, для них следует использовать модификатор **protected**, а для самого базового класса **public** или **internal.**

Спроектируем класс **Notebook**, который имеет два поля **model** (модель) и **ram** (оперативная память), а также методы **Start** (включение) и **End** (выключение). Имеет смысл считать класс **Notebook** потомком класса **Computer**, который наследует эти поля и метод **Start** без изменения (при включении они работают одинаково). Особенность ноутбука – наличие батареи, которая может заряжаться и после выключения ноутбука. Поэтому в класс **Notebook** надо добавить поле **time** (требуемое время зарядки батареи) и переопределить метод **End**, чтобы он показывал это время.

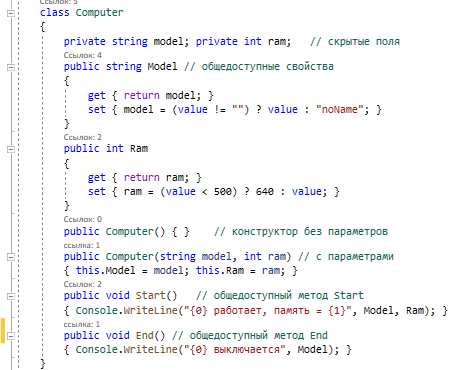
.

**Порядок выполения работы:**

**Пример 1**

*Реализация наследования.*

1. Создадим проект **con8\_1\_ФИО**. По умолчанию он имеет класс **Program**.
2. В этом же пространстве имен в отдельном файле создадим класс **Computer**
3. Для класса **Computer** определим поля **ram, mode**l. На первом этапе целесообразно все поля задавать общедоступными (public), и лишь убедившись в отсутствии ошибок, инкапсулировать поля, настроив ограничения доступа в методах set – get. Окончательный вид программы:

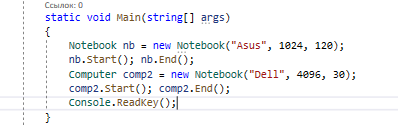


1. В том же пространстве имен в отдельном файле создадим класс **Notebook** с полем **time,** инкапсулируем его**.** Создадим новый метод **End** класса – наследника с модификатором **override**. Чтобы он смог переопределить одноименный метод базового класса, зададим тому в классе – предке Computer модификатор virtual.

Изображение выглядит как текст

Автоматически созданное описание

1. Протестируем программу, добавив в класс **Program** создание экземпляров класса **Notebook** и вызовы методов:



1. Обратим внимание на важное проявление принципа полиморфизма: объект **comp2** имеет тип **Computer**, но ведет себя как ноутбук, поскольку инициализируется с помощью конструктора **Notebook.**

**Задания для самостоятельной работы**

1. Модифицируйте пример **1** (**сon8):** добавив **public class Nеtbook** – потомок **Notebook**. В нем определите поле **mas** (масса). Переопределите метод **Start**, добавив информацию о массе. Протестируйте, изменяя параметры.
2. Используя инструменты и методы визуального проектирования, создайте классы с наследниками, содержащие указанные поля, конструкторы и методы. В классе **Program** создаются и инициализируются 2–3 объекта и указанными методами выводится информация. Продемонстрируйте результаты, изменяя параметры.

**Индивидуальные задания:**

1. Спроектируйте класс **Transport** с полями **mоdel** (модель)**, speed** (ско- рость), **mas** (масса) и методами **Start**, **Stop**, **ShowInfo.** Наследуйте от него классы **Avto** (автомобиль), **Moto** (мотоцикл), **Velo** (велосипед).
2. Спроектируйте класс **Animal** с полями **name** (имя)**, ves** (вес)**, col** (цвет) и методами **Run** (бегать), **Sleep**(спать), **Golos** (голос). Наследуйте от него классы **Cat** (Кот) и **Dog** (Собака). Переопределите метод **Golos** для каждого животного (**мяу-мяу** и **гав-гав**).
3. Спроектируйте класс **Tovar** с полями **name** (название), **price** (цена) и ме- тодом **Calc** (расчет и печать стоимости). Наследуйте от него классы **Book** (книги) c полем **kvo** (количество), **Pen** (ручки) c полем **kvo** и **Сandy** (конфеты) c полем **ves** (вес). Переопределите метод **Calc** для расчета стоимости каждого товара.