Основы программирования

Иерархия классов

Наследование

Для чего?

При решении сложных задач часто возникает необходимость в описании большого количества классов, многие из которых имеют некоторую общую функциональность.

Такой набор классов нуждается в упорядочении и структурировании.

В этом помогает реализация принципа наследования.

Пример

Объекты: человек и студент.

«Человек»: имя, рост, вес, другие характеристики.

«Студент»: имя, рост, вес, другие характеристики.
(то есть то же что и у объекта «Человек») +
название ВУЗа, специальность, курс, средний балл…

Объект «Студент» это наследник объекта «Человек».

Наследование это …

- механизм создания нового класса на основе уже существующего старого;
- старый класс называется «родительским», «предком» («super class»);
- новый класс называется «дочерним», «наследником» («sub class»);
- наследование нужно для повторного использования кода, которое облегчает следование принципу DRY (Don’t Repeat Yourself — не повторяйся);
- дочерний класс содержит методы и переменные родительского.

Наследование. Синтаксис

Класс может иметь сколько угодно потомков и только одного предка.

Синтаксис:

```
[атрибуты] [спецификаторы] class имя_класса :
                                  [предки]
{
      // тело класса
}
```

Пример. Класс «Студент»

```
class Student
{
string name; //фамилия
string address; //адрес
DateTime birthday; //дата рождения
int id; //номер зачетной книжки
int group; //номер группы
}
```

Пример. Класс «Преподаватель»

```
class Teacher
{
string name; //фамилия
string address; //адрес
DateTime birthday; //дата рождения
string department; //кафедра
string degree; //ученая степень
}
```

Пример. Базовый класс «Человек»

```
class Pearson
{
string name; //фамилия
string address; //адрес
DateTime birthday; //дата рождения
}
```

Пример. Производные классы

```
class Student : Person
{
int id; //номер зачетной книжки
int group; //номер группы
}

class Teacher : Person
{
string department; //кафедра
string degree; //ученая степень
}
```

Достоинства наследования

- позволяет исключить повторяющиеся блоки кода;
- упростить модификацию классов;
- упростить расширение иерархии классов.

Пример. Изменения

```
class Pearson
{
…
int INN; //номер ИНН
}
class Engineer: Person
{
string department; //кафедра
string qualification; //квалификация
}
```

Особенности наследования

- не поддерживается множественное наследование, класс может наследоваться только от одного класса
- при создании производного класса надо учитывать тип доступа к базовому классу - тип доступа к производному классу должен быть таким же, как и у базового класса, или более строгим. То есть, если базовый класс у нас имеет тип доступа `internal`, то производный класс может иметь тип доступа `internal` или `private`, но не `public`
- класс может быть объявлен с модификатором `sealed`, Это означает, что от этого класса нельзя наследовать и создавать производные классы.

Одиночное наследования

В языке C# для классов реализовано одиночное наследование, то есть

- класс может иметь только одного предка,
- и неограниченное количество потомков.

Количество потомков определяется необходимостью и здравым смыслом.

Родительский класс указывается в заголовке при описании класса, если предок явно не указан, то класс считается наследующим от базового класса `Object`.