**Объект. Теоретические сведения** ключевое слово new отвечает за вычисление правильного количества байт для указанного объекта и выделение досточного объема памяти из управляемой кучи.  
Объектные переменные представляют собой ссылки на объекты в памяти, а не сами объекты.  
Принцип разделения ответственности: один тип работает как объект приложения (определяет метод Main()) и множество других типов, составляющих все приложение.  
При определении ЛОКАЛЬНЫХ переменных присвоение начальных значений перед использованием ОБЯЗАТЕЛЬНО, т.к. они не получают значений по умолчанию.  
Static методы могут быть вызваны на уравне класса, а не на уравне экземпляра. Могут использовать только статические данные.  
Адреса объектов определяются во время выполнения, а значение const полей устанавливается на этапе компиляции.

Интерфейсы как элемент ООП.

Для объявления интерфейса используется ключевое слово interface. Интерфейс содержит только заголовки методов, свойств и событий. Для свойства указываются только ключевые слова get и (или) set. При объявлении элементов интерфейса не могут использоваться следующие модификаторы: abstract, public, protected, internal, private, virtual, override, static. Считается, что все элементы интерфейса имеют public-уровень доступа:

Если класс является производным от некоторого базового класса, то имя базового класса указывается перед именем реализуемого интерфейса.

Элементы интерфейса допускают явную и неявную реализацию. При *неявной реализации* тип должен содержать открытые экземплярные элементы, имена и сигнатура которых соответствуют элементам интерфейса. При *явной реализации* элемент типа называется по форме *<имя интерфейса>*.*<имя элемента>*, а указание любых модификаторов для элемента при этом запрещается. Все неявно реализуемые элементы интерфейса по умолчанию помечаются в классе как sealed. А значит, наследование классов не ведёт к прямому наследованию реализаций:

Чтобы осуществить наследование реализаций, требуется при реализации использовать модификаторы virtual и override[[1]](#footnote-2).

Подобно классам, интерфейсы могут наследоваться от других интерфейсов. При этом наследование интерфейсов может быть множественным. Один класс может реализовывать несколько интерфейсов ‑ имена интерфейсов перечисляются после имени класса через запятую.

Интерфейсы сходны с абстрактными классам, что иногда порождает проблему выбора «интерфейс или абстрактный класс?». Табл. 3 содержит сравнение возможностей этих пользовательских типов, для того, чтобы помочь сделать правильный выбор между ними.

Таблица 3

Сравнение абстрактных классов и интерфейсов

|  |  |
| --- | --- |
| **Абстрактные классы** | **Интерфейсы** |
| Не могут быть созданы напрямую, но могут содержать конструктор, который вызывается в классе-наследнике | Не могут содержать конструктор |
| Абстрактный класс может быть так дополнен элементами, что это не повлияет на его классы-наследники | Если в интерфейс помещаются дополнительные элементы, все классы, которые его реализуют, должны быть дополнены |
| Может хранить данные в полях | Не может хранить данных |
| Виртуальные элементы могут содержать базовую реализацию. Допустимы невиртуальные элементы | Все элементы являются виртуальными и не включают реализацию |
| Класс может наследоваться от единственного абстрактного класса | Класс может реализовывать несколько интерфейсов |
| Класс-наследник может переопределить только некоторые элементы абстрактного класса | Класс, который реализует интерфейс, должен реализовать все элементы интерфейса |
| Наследование поддерживается только для классов | Интерфейс может быть реализован структурой |

1. При явной реализации использование модификаторов невозможно. [↑](#footnote-ref-2)