1. [Статический](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/static) класс в основном такой же, как и нестатический класс, но имеется одно отличие: нельзя создавать экземпляры статического класса. Другими словами, нельзя использовать оператор [new](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/new-operator) для создания переменной типа класса. Поскольку нет переменной экземпляра, доступ к членам статического класса осуществляется с использованием самого имени класса.

Статические классы в C# используются для организации группы методов и свойств, которые не требуют создания экземпляра класса для их вызова. Они предоставляют удобный способ группировки функциональности и обеспечивают простой доступ к этим методам и свойствам без необходимости создавать объект класса.

* Содержит только статические члены.
* Создавать его экземпляры нельзя.
* Является запечатанным.
* Не может содержать [конструкторы экземпляров](https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/instance-constructors).

1. **Статический** **класс** **может** **содержать** только  **статические** поля, **статические** методы и **статический** конструктор. Не допускается создание объектов **статического** **класса**
2. Производный класс (derived class) – это класс, который использует программный код базового класса и изменяет (расширяет) его под свои потребности.

Базовый класс (англ. base class) — это класс, находящийся на вершине иерархии наследования классов и в основании дерева подклассов, т. е. не являющийся подклассом и не имеющий наследований от других суперклассов или интерфейсов. Базовым классом может быть абстрактный класс и интерфейс.

1. Ключевое base слово используется для доступа к членам базового класса из производного класса. Используйте его, если вы хотите: Вызов метода базового класса, который был переопределен другим методом. Определение конструктора базового класса, который должен вызываться при создании экземпляров производного класса.
2. 
3. У объекта производного класса есть доступ к унаследованным (public или protected) членам базового класса.

Метод **basefunc()** является методом, объявленным как **public** или **protected** в базовом классе.

8.Вот основные назначения виртуальных функций:

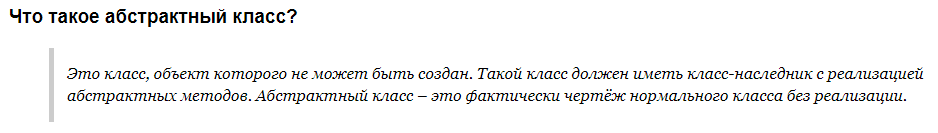
Полиморфизм

Замещение методов: Виртуальные функции позволяют производным классам замещать (переопределять) методы базовых классов с собственной реализацией. Расширение функциональности:

9 Виртуальные функции позволяют добавлять новую функциональность в производные классы, не изменяя интерфейс базового класса. Динамическое связывание: Виртуальные функции поддерживают динамическое связывание.

1. Внутри базового класса: Переменные с модификатором **protected** могут быть непосредственно доступными внутри самого базового класса, в котором они определены. То есть, методы и свойства базового класса могут обращаться к этим переменным напрямую.

Внутри производного класса: Переменные с модификатором **protected** также доступны внутри производных классов, которые наследуют базовый класс.

1. Переменные с модификатором **private** не наследуются производными классами. Модификатор **private** делает переменные и члены класса видимыми только внутри самого класса, в котором они определены, и не доступны в производных классах.
2. **as**: **as** используется для явного приведения типов с проверкой безопасности типов. Если приведение типа успешно, оно возвращает объект с приведенным типом, в противном случае возвращает **null. is: is используется для проверки типа объекта. Он возвращает true, если объект является экземпляром указанного типа, и false, если объект не является экземпляром указанного типа. Основное назначение is - это определение типа объекта перед выполнением операций, связанных с этим типом.**
3. C# не поддерживает множественное наследование классов (т.е., класс не может наследовать более одного базового класса), но поддерживает множественное наследование интерфейсов.
4. Да, в C# можно запретить наследование от класса с использованием ключевого слова **sealed**
5. Для запрета перекрытия метода в производных классах, вам следует объявить метод как **sealed** в public классе
6. 
7. **Когда класс содержит абстрактные методы**: Если ваш класс содержит хотя бы один абстрактный метод (метод без реализации), то этот класс должен быть объявлен как абстрактный

**Когда класс предназначен только для использования как базовый класс**

1. Разница между абстрактными и виртуальными классами. **Создание экземпляров**: Абстрактные классы не могут иметь экземпляров. Это означает, что нельзя создать объект абстрактного класса напрямую. Виртуальные классы могут иметь экземпляры, и их можно создавать.

Абстрактные методы:

Абстрактный метод (abstract method) не имеет реализации в абстрактном классе, который его объявляет.

Абстрактные методы обязательно должны быть реализованы в производных классах, иначе эти классы также должны быть объявлены как абстрактные.

Виртуальные методы:

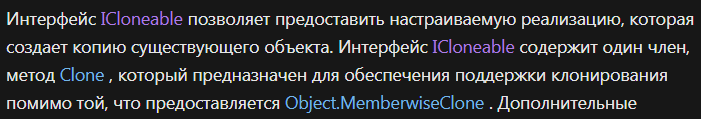
Виртуальный метод (virtual method) имеет реализацию в базовом классе, но может быть переопределен в производных классах.

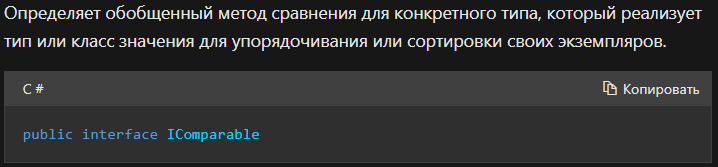
Переопределение виртуального метода в производных классах является опциональным. Если метод не переопределен, будет использоваться его реализация из базового класса.

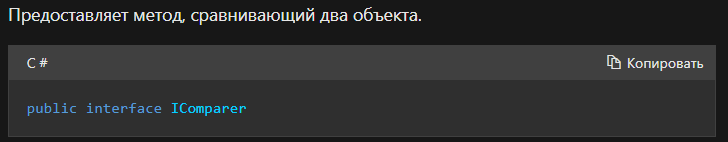
Виртуальные методы обычно используются, чтобы предоставить базовую реализацию, которую можно изменить в производных классах.

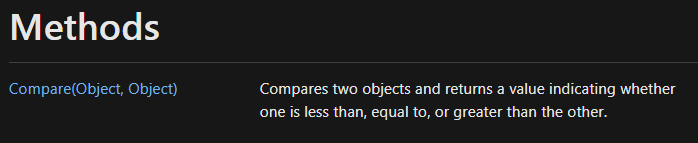
1. Методы, свойства, индексаторы, события.
2. Интерфейс (interface) в C# - это абстрактный тип данных, который определяет набор абстрактных членов (методов, свойств, событий и индексаторов), которые класс должен реализовать. Интерфейс задает контракт, который гарантирует, что класс, реализующий интерфейс, предоставит определенный набор функциональности.
3. .
4. Ю
5. .
6. Причина заключается в том, что интерфейс определяет контракт, который классы, реализующие интерфейс, обязаны выполнить. Методы интерфейса по своей природе всегда являются абстрактными и предназначены для реализации в классах, которые этот интерфейс реализуют. Модификаторы видимости указываются именно в реализации методов класса, а не в интерфейсе.
7. Можно
8. Интерфейсы позволяют множественное наследование. Класс может реализовывать несколько интерфейсов одновременно.

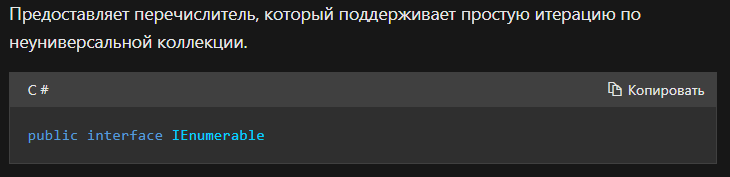
Абстрактные классы поддерживают только одиночное наследование. Класс может наследовать только один абстрактный класс

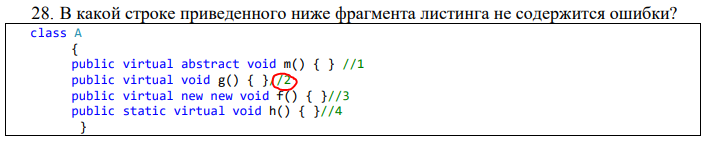
1. 









28. 

29. Итак, результат выполнения метода **m** будет следующим:

* **x** из класса **B** равно 3.
* **base.x** из базового класса **A** равно 4.

Таким образом, на консоль будет выведено "3 4".

30. Итак, результат выполнения программы будет следующим:

A B

На консоль будет выведено "A" с новой строки, а затем "B" с новой строкой.

31. Интерфейс или класс

32. Верное присваивание для объектов, определенных в листинге, будет следующим:

1. b = a;

Поскольку **B** наследует от **A**, объект класса **B** может быть присвоен переменной типа **A**.

Начало формы