Иерархия

**Наследование**:

* **Наследование** в C# обозначается ключевым словом :. Когда класс наследуется от другого класса, он может использовать свойства и методы базового класса, а также добавлять свои уникальные.

**Абстрактные классы**:

* **Абстрактные классы** — это классы, которые нельзя инстанцировать напрямую. Они определяют структуру для своих наследников.
* В абстрактных классах могут быть как обычные, так и **абстрактные методы**, которые должны быть реализованы в производных классах.

**Невозможность создать объект абстрактного класса**:

* Нельзя создать экземпляр абстрактного класса напрямую.

 Абстрактный метод объявляется без тела (реализации), и производные классы обязаны его переопределить и реализовать.

*  Абстрактные методы могут быть объявлены только в абстрактных классах.

**Интерфейсы**:

* **Интерфейсы** — это контракты, которые могут быть реализованы классами. Интерфейсы не содержат реализации методов, только их сигнатуры.
* Класс может реализовывать несколько интерфейсов, в отличие от наследования, где класс может наследоваться только от одного класса.

Интерфейс

**Виртуальные методы** в C# — это методы, которые могут быть переопределены в классах-наследниках. Они позволяют классам предоставлять базовую реализацию, которую производные классы могут изменить или расширить при необходимости.

**Основные характеристики виртуальных методов:**

**Ключевое слово virtual**:

* + Метод в базовом классе объявляется с ключевым словом virtual, что делает его "виртуальным" — то есть возможным для переопределения в производных классах(необязательное).

**Запрет на наследование**:

* Класс, помеченный как sealed, не может быть базовым для других классов.

**Одноименные методы** — это методы в классе или интерфейсе, которые имеют одинаковое имя, но могут отличаться по своему поведению, параметрам или возвращаемым значениям. В C# одноименные методы могут существовать в рамках одного класса (путем перегрузки методов) или в разных классах или интерфейсах (путем реализации одного и того же метода в разных контекстах).

### Оператор is

Оператор is используется для проверки, является ли объект экземпляром определенного типа. Он возвращает true, если объект может быть приведен к указанному типу, и false в противном случае.

### Оператор as

Оператор as используется для безопасного приведения типов. Если приведение типа не удается, оператор as возвращает null, вместо того чтобы выбросить исключение. Это позволяет избежать ошибок времени выполнения.

В программировании ключевое слово **sealed** используется для того, чтобы запретить наследование класса или переопределение методов, свойств, и других членов.

**Переопределение** — это процесс, при котором производный (наследуемый) класс изменяет реализацию метода, унаследованного от базового класса. В C# для этого используется ключевое слово override.

**1. Для чего используют статические классы?**

Статический класс в C# действительно используется для хранения методов и данных, которые не зависят от конкретного экземпляра (объекта) класса. Все члены такого класса являются статическими, и для их вызова не требуется создавать объект класса.

**2. Что может содержать статический класс?**

Статический класс может содержать:

* Статические методы
* Статические свойства
* Статические поля
* Статические конструкторы

Статический класс не может содержать экземплярные члены (не статические методы или поля).

**3. Что такое производный и базовый классы?**

* **Базовый класс** (или родительский класс) — это класс, от которого наследуются другие классы.
* **Производный класс** (или дочерний класс) — это класс, который наследует свойства и методы базового класса и может добавлять свои собственные.

**4. Как используют ключевое слово base?**

Ключевое слово base используется для доступа к членам базового класса из производного класса. Это может включать вызов конструктора базового класса, а также доступ к методам и свойствам базового класса.

**5. В чем заключена основная задача наследования?**

Основная задача наследования заключается в повторном использовании кода и создании иерархии классов, где производные классы могут наследовать функциональность базовых классов, упрощая поддержку и расширение кода.

**6. Пусть базовый класс содержит метод basefunc(), а производный класс не имеет метода с таким именем. Может ли объект производного класса иметь доступ к методу basefunc()? Если да, то при каких условиях?**

Да, объект производного класса может получить доступ к методу basefunc(), если он унаследовал его от базового класса. При создании экземпляра производного класса, методы базового класса становятся доступными через экземпляр производного класса.

**7. Напишите объявление конструктора без аргументов для производного класса B, который будет вызывать конструктор без аргументов базового класса A.**

csharp

Копировать код

class A

{

public A() { }

}

class B : A

{

public B() : base() { }

}

**8. Что такое полиморфизм? Приведите пример.**

Полиморфизм\*\*: способность объекта принимать множество форм

Полиморфизм позволяет использовать объекты разных классов через один и тот же интерфейс. Это может быть достигнуто через виртуальные методы.

**Пример**:

csharp

Копировать код

class Animal

{

public virtual void Speak()

{

Console.WriteLine("Animal speaks");

}

}

class Dog : Animal

{

public override void Speak()

{

Console.WriteLine("Dog barks");

}

}

class Program

{

static void Main()

{

Animal myDog = new Dog();

myDog.Speak(); // Выведет: Dog barks

}

}

**9. Определите назначение виртуальных функций.**

Виртуальные функции позволяют производным классам переопределять поведение метода, предоставляя возможность реализации разного поведения для одного и того же метода в зависимости от типа объекта.

Определяются в базовом классе с ключевым словом virtual. Это указывает, что метод может быть переопределён в производных классах.

Могут быть переопределены в производных классах с использованием ключевого слова override.

**10. Кому доступны переменные с модификатором protected?**

Переменные с модификатором protected доступны:

* В базовом классе
* В производных классах
* Нельзя получить доступ к ним из внешних классов, если они не наследуют базовый класс.

**11. Наследуются ли переменные с модификатором private?**

Нет, переменные с модификатором private не наследуются. Они доступны только в пределах своего класса.

**12. as, is – что это, как применяется? В чем между ними отличие?**

* **is**: Оператор, используемый для проверки типа объекта.
* **as**: Оператор, используемый для безопасного приведения типов. Если приведение не удается, возвращает null.

**Отличие**: is используется для проверки типа, а as для приведения. Если приведение через as не удалось, вы получите null, в то время как is вернет false.

**13. Поддерживает ли C# множественное наследование?**

Нет, C# не поддерживает множественное наследование для классов. Однако классы могут реализовывать несколько интерфейсов.

**14. Можно ли запретить наследование от класса?**

Да, можно запретить наследование от класса, используя модификатор sealed.

**15. Можно ли разрешить наследование класса, но запретить перекрытие метода?**

Да, можно использовать модификатор sealed на виртуальном методе, чтобы запретить его перекрытие в производных классах.

**16. Что такое абстрактный класс?**

Абстрактный класс — это класс, который не может быть создан как экземпляр и может содержать абстрактные методы (методы без реализации) и обычные методы. Он предназначен для наследования.

**17. В каком случае вы обязаны объявить класс абстрактным?**

Вы обязаны объявить класс абстрактным, если он содержит хотя бы один абстрактный метод, и если вы хотите запретить создание его экземпляров.

**18. В чем разница между абстрактными и виртуальными классами? Между виртуальными и абстрактными методами?**

* **Абстрактные классы**: Не могут быть инстанциированы, могут содержать абстрактные и обычные методы.
* **Виртуальные классы**: Могут быть инстанциированы, содержат методы, которые могут быть переопределены.
* **Абстрактные методы**: Не имеют реализации и должны быть переопределены в производных классах.
* **Виртуальные методы**: Имеют реализацию по умолчанию, которую можно переопределить.

**19. Какие компоненты класса могут быть виртуальными?**

Виртуальными могут быть:

* Методы
* Свойства
* Индексаторы
* События

**20. Что такое интерфейс?**

Интерфейс — набор абстрактных методов, свойств, событий и индексаторов, которые должны быть реализованы в производных классах

**21. Что может содержать интерфейс?**

Интерфейс может содержать:

* Методы
* Свойства
* Индексаторы
* События

**22. Как работать с объектом через унаследованный интерфейс?**

Вы можете создать объект класса, который реализует интерфейс, и использовать его через ссылку на интерфейс:

csharp

Копировать код

public interface IMyInterface

{

void MyMethod();

}

public class MyClass : IMyInterface

{

public void MyMethod() { }

}

IMyInterface obj = new MyClass();

obj.MyMethod();

**23. Приведите пример явной реализации интерфейса.**

csharp

Копировать код

public interface IMyInterface

{

void MyMethod();

}

public class MyClass : IMyInterface

{

void IMyInterface.MyMethod()

{

Console.WriteLine("Явная реализация метода интерфейса.");

}

}

MyClass obj = new MyClass();

// obj.MyMethod(); // Ошибка, нельзя вызвать

IMyInterface interfaceObj = (IMyInterface)obj;

interfaceObj.MyMethod(); // Вывод: Явная реализация метода интерфейса.

**24. Почему нельзя указать модификатор видимости для методов интерфейса?**

Методы интерфейса по умолчанию являются публичными, так как интерфейсы предназначены для определения контракта, который будет реализован классами. Указание модификаторов видимости было бы избыточным.

**25. Можно ли наследовать от нескольких интерфейсов?**

Да, в C# класс может реализовывать несколько интерфейсов.

**26. Назовите отличия между интерфейсом и абстрактным классом.**

* **Интерфейс**:
  + Не может содержать реализацию (до C# 8.0, сейчас возможна реализация по умолчанию).
  + Поддерживает множественное наследование.
  + Все члены по умолчанию публичные.
* **Абстрактный класс**:
  + Может содержать как абстрактные, так и обычные методы с реализацией.
  + Не поддерживает множественное наследование.
  + Может иметь модификаторы доступа для своих членов.

**27. Для чего используются стандартные интерфейсы ICloneable, IComparable, IComparer, IEnumerable?**

* **ICloneable**: Используется для создания копий объектов.
* **IComparable**: Определяет метод для сравнения объектов, что позволяет сортировать коллекции.
* **IComparer**: Используется для сравнения двух объектов, предоставляя способ для сортировки.
* **IEnumerable**: Позволяет объекту быть перечисляемым в цикле foreach, предоставляя доступ к элементам коллекции.

28. В какой строке приведенного ниже фрагмента листинга не содержится ошибки?

class A { public virtual abstract void m() { } //1

public virtual void g() { }//2

public virtual new new void f() { }//3

public static virtual void h() { }//4 }

2)Корректной является только строка **2**: public virtual void g() { }.

29. Что будет выведено на консоль в результате выполнения следующего фрагмента?

 Первое значение (3) — это значение поля x класса B.

 Второе значение (4) — это значение поля x класса A, доступное через base.x.

30. Что будет выведено на консоль в результате выполнения следующего фрагмента кода

A

B

31. Чем может быть M4 если дано следующее определение:

Правильный ответ: **1) M4 - только интерфейс.**

Разбор:

* В C#, структура **(struct)** не может наследоваться от класса или другой структуры. Она может реализовать только **интерфейсы**.
* Так как S1 является структурой и реализует M3 и M4, эти два элемента должны быть **интерфейсами**.

**Обоснование:**

1. Структуры в C# могут наследовать только интерфейсы. Они не могут наследовать классы, структуры или делегаты.
2. Поэтому, если S1 реализует M4, то M4 обязательно должен быть интерфейсом.

Таким образом, **M4** может быть **только интерфейсом**.

32. Выберите верное присваивание для объектов, определенных в листинге.

Правильные варианты — **2) a = b;** и **5) c = b;**.

33. Что будет выведено на консоль в результате выполнения следующего фрагмента, если раскомментировать строчку 1?

A B

34. В какой строке приведенного ниже фрагмента листинга содержится ошибка?

Давайте рассмотрим каждый из заданных вопросов по очереди.

**34. Ошибка в коде**

Фрагмент кода:

csharp

Копировать код

public abstract class A

{

public virtual string m() { return "A"; } //1

}

public class B : A

{

public override new string m() { return "B"; } //2

}

public class C : B

{

public string m() { return "C"; } //3

}

class Prоgrаm

{

static void Main(string[] args)

{

A ac = new C(); //4

Console.WriteLine(ac.m());

}

}

**Ошибка:**

* В строке 2 происходит объявление метода m() как override new, что не является корректным, поскольку new не имеет смысла в контексте переопределения виртуального метода. Если вы хотите переопределить метод, следует использовать только override, то есть:

csharp

Копировать код

public override string m() { return "B"; }

**Вывод в строке 4:**

* Если вы исправите строку 2 и запустите код, то будет выведено "A" из-за того, что метод m() вызывается через ссылку типа A и возвращает реализацию базового класса.

35. Почему приведенный ниже фрагмент листинга содержит ошибку?

**Ошибка:**

* Ошибка возникает в строке 4, так как нельзя создавать экземпляр абстрактного класса. Вы можете создать объект только производного класса, который реализует все абстрактные методы и свойства, если таковые имеются. Вам нужно создать класс, производный от Student, и затем создать экземпляр этого производного класса.

36. В какой строке может быть ошибка компиляции?

**Ошибка:**

* В строке 4 будет ошибка компиляции. Вы не можете присвоить значение переменной one, которая является экземпляром класса B, значению переменной two, которая является экземпляром класса C. Компилятор не позволяет такое присваивание, так как one будет ссылаться на экземпляр класса B, а two на экземпляр класса C.

37. Что будет выведено на консоль в результате выполнения следующего фрагмента листинга:

**Вывод:**

* Если закомментировать строки, содержащие Interface1, и запустить программу, то в результате выполнения будет выведено:

arduino

Копировать код

new G

**Объяснение:**

* Метод g() в классе B переопределяет метод g() класса A с использованием ключевого слова new, что делает его новым методом, который скрывает оригинальный метод. Когда вы вызываете obj.g(), метод g() класса B будет вызван и выведет "new G".