1. Для чего используют статические классы?

Статические классы в языке программирования C# используются для группировки связанных по функциональности методов и свойств, которые могут быть вызваны без необходимости создания экземпляра класса.

1. Что может содержать статический класс?

Статический класс может содержать только статические члены, такие как статические методы, статические свойства, статические события и статические поля.

1. Что такое производный и базовый классы?

Базовый класс предоставляет основу для производного класса, который наследует его свойства, методы и другие члены. Производный класс может добавлять новые члены, переопределять или расширять унаследованные члены базового класса.

1. Как используют ключевое слово base?

Ключевое слово base используется в C# для обращения к членам базового класса из производного класса.

1. В чем заключена основная задача наследования?

Основная задача наследования в объектно-ориентированном программировании заключается в создании иерархии классов, где производные классы наследуют свойства и методы базового класса.

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 иметь доступ к методу basefunc() базового класса, если этот метод имеет модификатор доступа, позволяющий это (например, public или protected).

7. Напишите объявление конструктора без аргументов для производного класса B,

который будет вызывать конструктор без аргументов базового класса A.

public class B : A

{

public B() : base()

{

// Код инициализации производного класса B

}

}

8. Что такое полиморфизм? Приведите пример.

Полиморфизм позволяет объектам одного типа проявлять различное поведение в зависимости от контекста. Это означает, что один и тот же метод может иметь различную реализацию в разных классах. Полиморфизм позволяет программисту работать с объектами разных классов с использованием общего интерфейса, что упрощает разработку гибкого и расширяемого кода.

9. Определите назначение виртуальных функций.

Виртуальные функции в языке программирования C# предназначены для создания полиморфизма и переопределения методов в производных классах. Они позволяют базовому классу объявить метод, который может быть переопределен в производных классах.

Виртуальные функции позволяют производным классам переопределять их для изменения поведения базового класса.

10. Кому доступны переменные с модификатором protected?

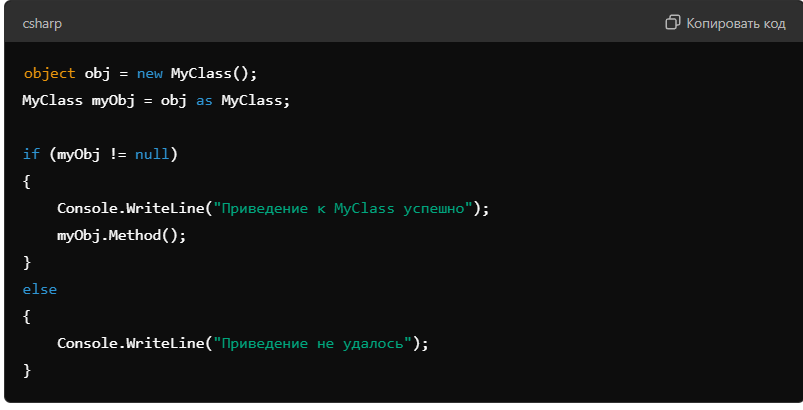
Переменные с модификатором protected доступны внутри класса, а также внутри производных классов, которые наследуют базовый класс. Они не доступны вне класса или вне иерархии наследования.

11. Наследуются ли переменные с модификатором private?

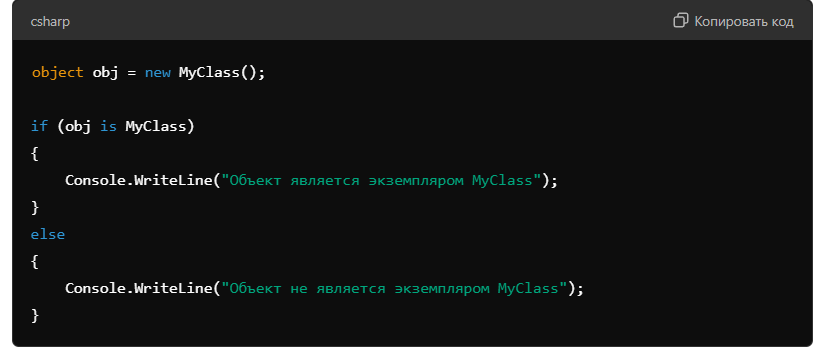
Нет

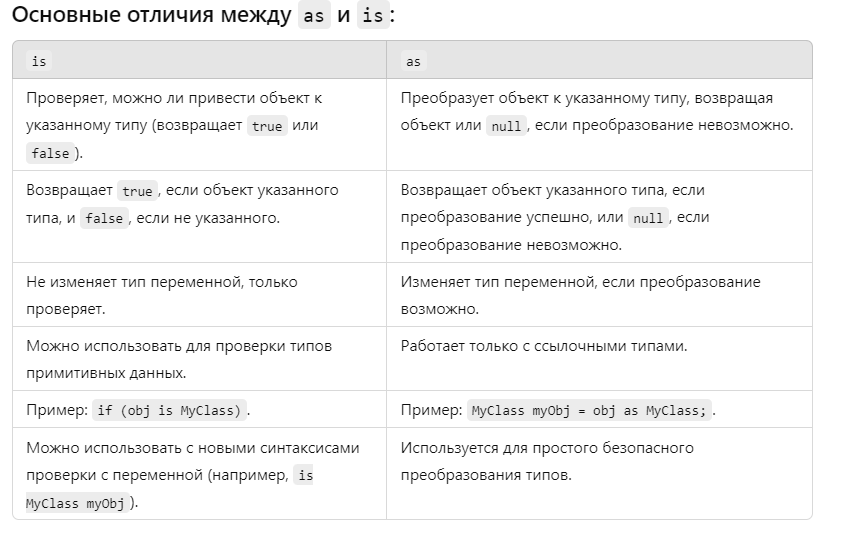
12. As, is – что это, как применяется? В чем между ними отличие ?

as и is - это операторы в языке C#, используемые для проверки и приведения типов. Оператор as используется для попытки явного приведения типа объекта к другому типу. Если приведение типа возможно, то оператор as возвращает объект с приведенным типом. Если приведение типа невозможно, то оператор as возвращает null, а не генерирует исключение. Например: var result = obj as MyClass;



Оператор is используется для проверки, является ли объект экземпляром определенного типа. Он возвращает булевое значение true, если объект является экземпляром указанного типа, и false в противном случае. Например: if (obj is MyClass) { ... }





13. Поддерживает ли C# множественное наследование?

Нет, C# не поддерживает множественное наследование классов, то есть класс не может наследовать от нескольких классов одновременно. Однако, в C# есть возможность реализации множественного наследования интерфейсов, где класс может реализовывать несколько интерфейсов.

14. Можно ли запретить наследование от класса?

можно запретить наследование от класса, используя ключевое слово sealed.

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

Можно если у метода sealed а у класса public

16. Что такое абстрактный класс?

Абстрактный класс в C# это класс, который не может быть инстанциирован напрямую, а служит в качестве базового класса для других классов. Он может содержать как абстрактные методы (методы без реализации), так и обычные методы с реализацией.

Абстрактный класс — это класс, который не может быть создан как экземпляр и может содержать абстрактные методы без реализации, которые должны быть реализованы в производных классах.

**Абстрактные классы в объектно-ориентированном программировании** — это базовые классы, которые можно наследовать, но нельзя реализовывать. То есть на их основе нельзя создать объект.

17. В каком случае вы обязаны объявить класс абстрактным?

Класс должен быть объявлен абстрактным, если в нём есть хотя бы один абстрактный метод.

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

виртуальными и абстрактными методами?

Абстрактный класс – это класс, который не может быть инстанциирован напрямую, а служит в качестве базового класса для других классов. Виртуальный класс – это класс, который может быть инстанциирован, но может также быть использован в качестве базового класса для других классов.

Абстрактный класс может содержать абстрактные методы, которые не имеют реализации в самом абстрактном классе, а должны быть реализованы в производных классах. Виртуальный класс не может содержать абстрактные методы, только методы с реализацией.

19. Какие компоненты класса могут быть виртуальными?

Свойства, методы, индексаторы

20. Что такое интерфейс?

Интерфейс представляет ссылочный тип, который может определять некоторый функционал - набор методов и свойств без реализации. Затем этот функционал реализуют классы и структуры, которые применяют данные интерфейсы.

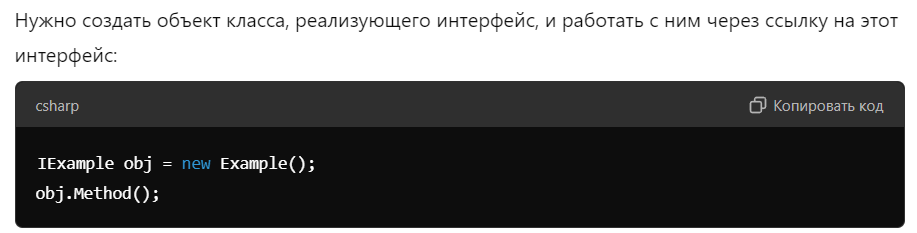
Перечень сигнатур методов

21. Что может содержать интерфейс?

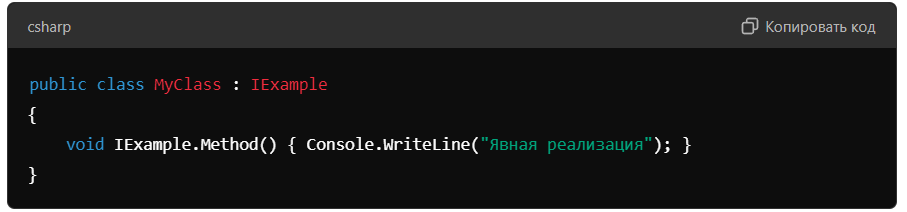
Методы и свойства (события)

22. Как работать с объектом через унаследованный интерфейс?

Создайте экземпляр класса и присвойте его переменной типа интерфейса.



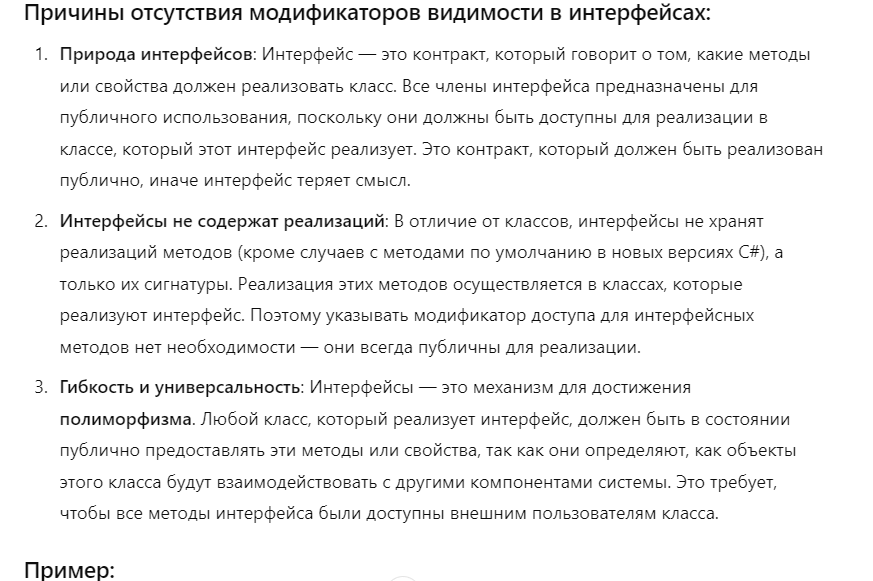
23. Приведите пример явной реализации интерфейса.



???????????????

24. Почему нельзя указать модификатор видимости для методов интерфейса?

Если бы можно было указывать модификатор видимости для методов интерфейса, возникла бы проблема неправильного нарушения контракта: классы, реализующие интерфейс, могли бы сделать методы недоступными для внешнего кода, что нарушило бы ожидания, связанные с интерфейсом.



25. Можно ли наследовать от нескольких интерфейсов?

Да

26. Назовите отличия между интерфейсом и абстрактным классом.

В интерфейсе можно определить только сигнатуры методов, без их реализации. Абстрактный класс может содержать как абстрактные методы (без реализации), так и методы с реализацией.

Абстрактный класс может содержать поля (переменные состояния) и определять общее состояние для классов-наследников. Интерфейс не может содержать переменные состояния или определять общее состояние.



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

ICloneable (System.ICloneable): Этот интерфейс определяет метод Clone(), который позволяет создать копию объекта.

IComparable (System.IComparable): Этот интерфейс определяет метод CompareTo(), который позволяет сравнить текущий объект с другим объектом того же типа.

IComparer (System.Collections.IComparer): Этот интерфейс определяет метод Compare(), который позволяет сравнивать два объекта для определения их относительного порядка.

IEnumerable (System.Collections.IEnumerable): Этот интерфейс определяет метод GetEnumerator(), который позволяет перебирать элементы коллекции последовательно.

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

}

Ошибка содержится в строке 1. В C# ключевые слова virtual и abstract не могут использоваться вместе при определении метода.

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

class A

{

public int x = 1;

}

class B : A

{

public new int x = 2;

public void m(int a, int b)

{

x = a;

base.x = b;

Console.Write(x + " " + base.x); }

}

class Test

{

static void Main(string[] args)

{

A a = new A();

B b = new B();

b.m(3, 4);

}

}

3 4

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

class A

{

public class B : A

{

public override void mA()

{

Console.WriteLine("B ");

}

}

public virtual void mA()

{

Console.WriteLine("A ");

}

}

class Prоgrаm

{

static void Main(string[] args)

{

A a = new A();

A.B b = new A.B();

a.mA();

b.mA();

}

}

A B

31. Чем может быть M4 если дано следующее определение:

**public class** C1 : M1, M2 { }

**public struct** S1 : M3, M4 { };

Варианты ответа:

1) M4 - только интерфейс

2) M4 - интерфейс или класс

3) M4 - только класс

4) M4 - только структура

5) M4 - делегат

32. Выберите верное присваивание для объектов, определенных в листинге.class A { }

class B : A { }

class C : B { }

class D { }

class Test

{

static void Main(string[] args)

{

A a = new A();

B b = new B();

C c = new C();

D d = new D();

}

}

Варианты ответа:

1) b = a;

2) a = b;

3) c = a;

4) d = a;

5) с = b;

33. Что будет выведено на консоль в результате выполнения следующего фрагмента, если

раскомментировать строчку 1?

public abstract class A

{

public virtual void method()

{ Console.Write("A "); }

}

public class B : A

{

public override void method()

{ // base.method(); // 1

// this.method(); // 2

Console.Write("B ");

}

}

class Program2

{

static void Main(string[] args)

{

A my = new B();

my.method();

}

}

Варианты ответа:

1) B

2) A

3) A B

4) B A

5) 0

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

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());

}

}35. Почему приведенный ниже фрагмент листинга содержит ошибку?

abstract class Student //1

{

public int Age { get; set; } //2

public string Name { get; set; } //3

}

static void Main(string[] args)

{

Student Olga = new Student();//4

}

36. В какой строке может быть ошибка компиляции?

class A{}

class B : A { }

class C : A { } //1

class Program4

{

static void Main()

{

A one = new B(); //2

A two = new C(); //3

one = two; //4

}

}

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

interface Interface1

{

void f();

void g();

}

class A

{

public void f() { System.Console.WriteLine("F"); }

public void g() { System.Console.WriteLine("G"); }

}

class B : A, Interface1

{

new public void g() { System.Console.WriteLine("new G"); }

}

class Program5

{

static void Main(string[] args)

{

//Interface1 obj = new B();

//obj.g();

B obj = new B();

obj.g();

}

}

New G