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

Статические классы в C# используются для группировки методов и данных, которые не привязаны к конкретным экземплярам класса. Их применяют, когда необходимо хранить глобальные данные или функции, которые могут использоваться без создания объекта класса. Например, такие классы как Math, которые предоставляют математические функции, могут быть статическими.

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

Статический класс может содержать только статические члены: методы, поля, свойства, конструкторы (неявные или явные) и события. Он не может содержать экземплярные члены и не может быть создан экземпляр статического класса.

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

* **Базовый класс** (или суперкласс) — это класс, от которого наследуются другие классы.
* **Производный класс** (или подкласс) — это класс, который наследует члены базового класса и может расширять его функциональность.

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

Ключевое слово base используется в производном классе для обращения к членам базового класса: например, для вызова методов или обращения к полям. Также base может использоваться для вызова конструктора базового класса в конструкторе производного.

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

Основная задача наследования — это возможность создавать новые классы на основе уже существующих, чтобы повторно использовать код и добавлять или изменять поведение без дублирования. Это способствует созданию иерархии классов с общими чертами.

**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 будет иметь доступ к методу basefunc() при условии, что метод basefunc() является публичным, защищённым (protected) или внутренним (при наследовании внутри одного сборочного модуля). Если метод приватный, то доступ к нему невозможен.

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

Копировать код

class B : A

{

public B() : base()

{

// Дополнительная инициализация производного класса

}

}

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

Полиморфизм — это способность объектов разных типов обрабатывать одно и то же сообщение (вызов метода) по-разному.

* способность к изменению функций, унаследованных от базового класса

Пример:

Копировать код

class Animal

{

public virtual void MakeSound()

{

Console.WriteLine("Animal sound");

}

}

class Dog : Animal

{

public override void MakeSound()

{

Console.WriteLine("Bark");

}

}

Animal myAnimal = new Dog();

myAnimal.MakeSound(); // Выведет "Bark"

**9. Определите назначение виртуальных функций.**

Виртуальные функции позволяют подклассам переопределять их для изменения поведения, наследуемого от базового класса. Они обеспечивают динамическое связывание и полиморфизм, позволяя вызывать переопределенные версии методов производных классов через ссылки на базовый класс.

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

Переменные с модификатором protected доступны самому классу, его производным классам и классам внутри того же сборочного модуля (при использовании модификатора protected internal).

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

Переменные с модификатором private не наследуются напрямую в производные классы. Однако они существуют в базовом классе и могут быть доступны через специальные методы доступа (getter/setter) или через конструкцию protected.

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

* is проверяет, является ли объект экземпляром указанного типа: if (obj is MyClass).
* as пытается выполнить приведение к указанному типу, и если не удаётся, возвращает null: MyClass obj = someObj as MyClass.

Разница: is используется только для проверки, а as для приведения объекта к типу.

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

Нет, C# не поддерживает множественное наследование классов. Однако C# поддерживает множественное наследование интерфейсов.

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

Да, можно запретить наследование, объявив класс с модификатором sealed:

Копировать код

sealed class MyClass { }

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

Да, для этого метод необходимо объявить с модификатором sealed в производном классе:

public sealed override void SomeMethod() { }

**16. Что такое абстрактный класс?**

* Служит только для порождения потомков - предоставляют базовый функционал для классов-наследников.
* Задает интерфейс для всей иерархии
* Может содержать и полностью определенные методы, переменные, конструкторы, свойства
* Создавать объект абстрактного класса нельзя!!!!!!! (ссылку можно)

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

Если класс содержит хотя бы один абстрактный метод, он должен быть объявлен как абстрактный.

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

* Виртуальный класс в C# отсутствует как концепция.
* Разница между абстрактными и виртуальными методами: абстрактный метод не имеет реализации в базовом классе, а виртуальный метод может иметь реализацию, которая может быть переопределена в производном классе.

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

Методы, свойства, события и индексаторы могут быть виртуальными.

**20. Что такое интерфейс?**

Интерфейс — это контракт, который определяет набор методов и свойств без реализации. Классы или структуры, реализующие интерфейс, обязаны предоставить реализацию всех его членов.

**21. Что может содержать интерфейс?**

Интерфейс может содержать методы, свойства, события(без реализации) и индексаторы. С C# 8.0 интерфейсы также могут содержать статические члены — методы и поля (с C# 8.0), но не могут быть наследуемыми.

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

Работа с объектом через интерфейс осуществляется следующим образом:

Копировать код

IMyInterface obj = new MyClass();

obj.MyMethod();

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

Копировать код

interface IMyInterface

{

void MyMethod();

}

class MyClass : IMyInterface

{

void IMyInterface.MyMethod()

{

Console.WriteLine("Явная реализация");

}

}

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

Потому что все методы интерфейса по умолчанию public, и доступность должна быть одинаковой для всех реализующих классов.

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

Да, класс может реализовывать несколько интерфейсов:

class MyClass : IInterface1, IInterface2 { }

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

* Интерфейс не может содержать полей, а абстрактный класс может.
* Класс может реализовать несколько интерфейсов, но наследоваться только от одного абстрактного класса.
* Интерфейс не содержит реализации методов (за исключением методов с реализацией по умолчанию), а абстрактный класс может.

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

* **ICloneable**: для создания копий объектов.
* **IComparable**: для сравнения объектов, например, при сортировке.
* **IComparer**: для сравнения двух объектов.
* **IEnumerable**: для предоставления механизма итерации по коллекциям.