1. Назовите принципы ООП. Поясните каждый из них.(наследование, полиморфизм, инкапсуляция)

Инкапсуляция- сокрытие состояния объекта от прямого доступа из вне.

Полиморфизм- способность вызывать один и тот же метод для разнвх объектов и каждый объект будет реагировать по разному.

2. Назовите класс .NET, от которого наследуются все классы. (Все классы в .NET наследуются от класса System.Object.)

3. Охарактеризуйте открытые методы System.Object.( Класс System.Object предоставляет несколько открытых методов, включая:

* ToString(): Возвращает строковое представление объекта.
* Equals(Object obj): Определяет, равен ли указанный объект текущему объекту.
* GetHashCode(): Возвращает хэш-код для данного объекта.
* GetType(): Возвращает объект Type, представляющий тип текущего экземпляра.

)

4. Охарактеризуйте закрытые методы System.Object. (Класс System.Object не имеет закрытых методов, так как все его методы являются открытыми или защищенными. Однако он может иметь защищенные члены, доступные только производным классам.)

5. Приведите пример определения класса.

6. Какие ключевые слова можно использовать при определении класса?

**class** — ключевое слово для объявления класса.

**static** — ключевое слово, указывающее, что класс или его члены (методы, поля) принадлежат классу, а не конкретному экземпляру.

**sealed** — предотвращает наследование от данного класса.

**virtual** — позволяет переопределять методы в наследниках.

**partial** — позволяет определять класс или метод в нескольких файлах.

**const** — для создания констант, значения которых не могут быть изменены после объявления.

7. В чем отличие между объектом и классом? (Класс это абстрактный тип данных, определяемый программистом (пользователем). С помощью классов определяются свойства объектов. Объекты это экземпляры класса.)

8. Что такое конструктор? Когда вызывается конструктор?( Конструктор – это метод класса, имеющий имя класса. Конструктор автоматически вызывается на этапе компиляции при создании экземпляра данного класса. Попытка вызвать конструктор явным образом вызовет ошибку компиляции.)

9. Перечислите свойства конструктора? (На конструкторы накладываются следующие ограничения: 1. Конструктор не может иметь возвращаемого значения даже void 2. Как следствие 1 нельзя использовать оператор return() 3. Конструкторы нельзя объявлять виртуальными. Имя конструктора совпадает с именем класса)

10. Что такое деструктор (destructor) ? (В то время как [**конструкторы**](https://ravesli.com/urok-116-konstruktory/) предназначены для инициализации класса, деструкторы предназначены для очистки памяти после него.)

11. Что такое this?( В языке C# имеется ключевое слово this, которое обеспечивает доступ к текущему экземпляру класса.)

12. вывод программы( 5 5; 7 7)

13. Спецификаторы доступа для класса и методов класса в C#

1 public: Доступен из любого места.

2 private: Доступен только внутри самого класса.

3 protected: Доступен внутри класса и его производных.

4 internal: Доступен только в пределах одной сборки.

5 protected internal: Доступен внутри класса, производных классов и в пределах одной сборки.

6 private protected: Доступен только внутри класса и его производных, но только в пределах одной сборки.

14. Модификатор protected internal

Модификатор protected internal позволяет члену класса быть доступным как для производных классов, так и для классов в пределах одной сборки. Это комбинированный доступ, который предоставляет большую гибкость, чем просто protected или internal.

15. Использование ref и out параметров

В отличии от функций С, при передаче методу параметров по адресу, необходимо указывать ключевое слово ref или out. Эти ключевые слова сообщают компилятору, что адреса параметров функции совпадают с адресами переменных, передаваемых в качестве параметров. Любое изменение значения параметров в этом случае приведет к изменению и переменных вызывающего кода.

16. Пример необязательных и именованных параметров метода

public void DisplayInfo(string name, int age = 18) // необязательный параметр

{

Console.WriteLine($"Name: {name}, Age: {age}");

}

// Использование именованных параметров

DisplayInfo(name: "Alice", age: 25);

DisplayInfo(name: "Bob"); // Age будет 18 по умолчанию

17. Пример полей класса

public class Example

{

public static int StaticField = 10; // статическое поле

public const double Pi = 3.14; // константное поле

private readonly string name; // поле только для чтения

public Example(string name)

{

this.name = name; // инициализация поля только для чтения

}

}

18. Пример определения свойств класса

public class Person

{

private string name;

public string Name // свойство

{

get { return name; }

set { name = value; }

}

}

Свойства помогают инкапсулировать доступ к полям класса, позволяя добавлять логику при получении или установке значений.

19. Явное имя параметра, передаваемого в метод set свойства

Имя параметра в методе set обычно называется value. Пример:

set { name = value; }

20. Автоматические свойства

Автоматические свойства позволяют быстро создавать свойства без явного определения полей. C# автоматически создает скрытое поле для хранения значения.

public class Person

{

public string Name { get; set; } // автоматическое свойство

}

21. Индексаторы класса

Индексаторы позволяют объектам класса вести себя как массивы. Они определяются с помощью ключевого слова this.

public class SampleCollection

{

private int[] arr = new int[100];

public int this[int index] // индексатор

{

get { return arr[index]; }

set { arr[index] = value; }

}

}

Ограничения: индексатор может иметь только один модификатор доступа, и его можно использовать только с типами, которые могут иметь методы.

22. Перегруженный метод

Перегруженный метод — это метод с тем же именем, но разными параметрами (типом, количеством или порядком параметров). Это позволяет создавать несколько версий одного метода для разных случаев.

23. Partial класс и его преимущества

Частичные классы позволяют разделить реализацию класса на несколько файлов. Это удобно для работы в команде, позволяет лучше организовать код и упрощает его поддержку.

24. Анонимный тип в C#

Анонимный тип — это тип, который создается без явного определения класса. Используется для создания объектов с набором свойств без создания отдельного класса.

var person = new { Name = "Alice", Age = 25 };

25. Статические классы

Статические классы предназначены для хранения статических членов и не могут быть инстанцированы. Их используют для группировки методов и свойств, которые не требуют создания экземпляра класса.

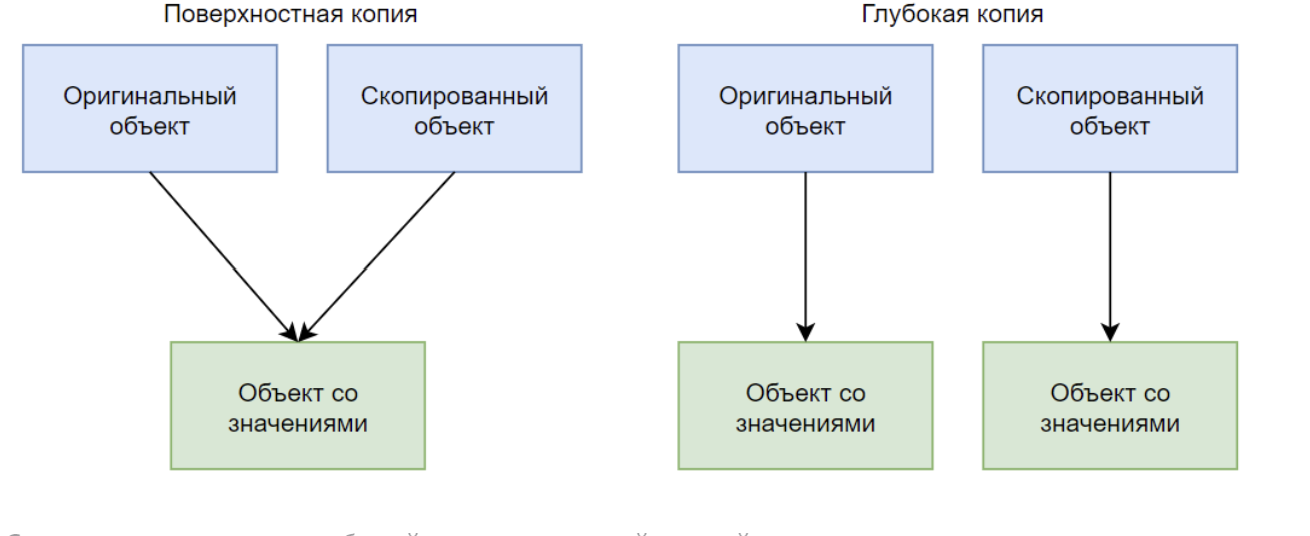
26. Разница между статическим полем и экземплярным • Статическое поле: Одно значение для всего класса, общее для всех экземпляров.

• Экземплярное поле: Каждое значение уникально для конкретного экземпляра класса.

27. Работа статических конструкторов

Статические конструкторы инициализируют статические члены класса. Они вызываются автоматически перед первым обращением к статическим членам или созданием экземпляра класса. Не могут принимать параметры и не могут быть вызваны явно.

28. Поверхностное и глубокое копирование



• Поверхностное копирование: Создает копию объекта, но ссылки на вложенные объекты остаются прежними.

• Глубокое копирование: Создает копию объекта и всех его вложенных объектов, полностью независимую от оригинала.

29. Разница между равенством и тождеством объектов

• Равенство (Equals): Сравнивает значения объектов. Может быть переопределено для сравнения содержимого.

• Тождество (ReferenceEquals): Сравнивает ссылки на объекты. Проверяет, указывают ли ссылки на один и тот же объект в памяти.

30. Частичные классы и частичные методы

• Частичные классы: Позволяют разделить реализацию класса на несколько файлов.

• Частичные методы: Методы, которые могут быть определены только в частичных классах. Они могут не иметь реализации, и если реализация отсутствует, они игнорируются компилятором.

Если нужно больше деталей или примеры по какому-то из пунктов, дайте знать!

31.вывод: System.Int32

32.вывод: a=2, b=1

33.неверный - 2, этот код является ошибочным, так как конструкторы **не могут иметь возвращаемый тип**.

34.может быть 0,1,2 аргумента

35.

36.вывод( A static

A)статический вызывается 1 раз при первом создании нового объекта

37. b = 30; //3

Ошибка возникает потому, что static readonly поле можно изменять только в статическом конструкторе, но не в обычных конструкторах.