Вот ответы на ваши вопросы по объектно-ориентированному программированию (ООП) и C#:

**1. Принципы ООП**

* **Инкапсуляция**: Скрытие внутренней реализации объекта и предоставление доступа к данным через публичные методы и свойства. Это защищает данные от неправомерного доступа и изменения.
* **Наследование**: Позволяет создавать новые классы на основе существующих, наследуя их свойства и методы. Это облегчает повторное использование кода.
* **Полиморфизм**: Позволяет объектам разных классов обрабатывать одинаковые команды, благодаря чему можно использовать один интерфейс для работы с разными типами объектов.
* **Абстракция**: Позволяет выделить только важные характеристики объекта, скрывая несущественные детали. Это упрощает взаимодействие с объектами.

**2. Класс .NET, от которого наследуются все классы**

Все классы в .NET наследуются от класса System.Object.

**3. Открытые методы System.Object**

* **Equals**: Сравнивает два объекта на равенство.
* **GetHashCode**: Возвращает хэш-код объекта.
* **GetType**: Возвращает тип текущего экземпляра.
* **ToString**: Возвращает строковое представление объекта.

**4. Закрытые методы System.Object**

Закрытых методов в System.Object нет, так как все методы являются открытыми или защищёнными.

**5. Пример определения класса**

csharp

Copy

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

}

**6. Ключевые слова для определения класса**

* public
* private
* protected
* internal
* abstract
* sealed
* static
* partial

**7. Отличие между объектом и классом**

* **Класс**: Шаблон или чертёж, описывающий свойства и методы.
* **Объект**: Конкретный экземпляр класса, который содержит данные и может выполнять методы.

**8. Конструктор**

Конструктор — это специальный метод, используемый для инициализации объектов. Он вызывается при создании нового экземпляра класса.

**9. Свойства конструктора**

* Имеет то же имя, что и класс.
* Не имеет возвращаемого типа.
* Вызывается автоматически при создании объекта.

**10. Деструктор**

Деструктор — это специальный метод, который вызывается перед уничтожением объекта. Он используется для освобождения ресурсов.

**11. this**

this — это ссылка на текущий экземпляр класса. Используется для доступа к членам класса внутри его методов.

**12. Результат выполнения кода**

plaintext

Copy

5 5

7 7

Первая строка выводит первоначальное значение Num, вторая — обновлённое значение.

**13. Спецификаторы доступа**

* public
* private
* protected
* internal
* protected internal
* private protected

**14. Модификатор protected internal**

Этот модификатор позволяет доступ к членам класса как для производных классов, так и для классов в том же сборке (assembly).

**15. Использование ref и out**

* ref: Передача параметра по ссылке, позволяющая изменять значение переменной вне метода.
* out: Также передаёт по ссылке, но требует, чтобы переменная была инициализирована внутри метода.

**16. Пример необязательных и именованных параметров**

csharp

Copy

public void DisplayInfo(string name, int age = 30) { /\* ... \*/ }

DisplayInfo("Alice", age: 25);

**17. Пример полей класса**

csharp

Copy

public class Example

{

public static int StaticField; // Статическое поле

public const double Pi = 3.14; // Константное поле

public readonly int ReadOnlyField; // Поле только для чтения

}

**18. Пример определения свойств**

csharp

Copy

public class Employee

{

public string Name { get; set; } // Свойство с автоматической реализацией

}

Свойства позволяют инкапсулировать данные, обеспечивая контроль доступа.

**19. Явное имя параметра в методе set**

Имя параметра в методе set называется value.

**20. Автоматические свойства**

Свойства, которые автоматически создают скрытое поле для хранения значения, например:

csharp

Copy

public string Name { get; set; }

**21. Индексаторы класса**

Индексаторы позволяют доступ к элементам класса через индекс (например, как массив). Ограничения: могут использоваться только с типами, которые могут хранить значения (например, не могут быть статическими).

**22. Перегруженный метод**

Методы с одинаковым именем, но разными параметрами (тип, количество), позволяющие выполнять разные операции в зависимости от переданных аргументов.

**23. Partial класс**

Классы, которые могут быть определены в нескольких файлах. Преимущества: упрощение разработки и разделение кода.

**24. Анонимный тип**

Тип, создаваемый без явного определения класса, обычно используется для хранения набора значений с автоматически созданными свойствами.

**25. Статические классы**

Используются для группировки методов и свойств, которые не требуют создания экземпляра класса, например, утилиты.

**26. Отличие статического поля от экземплярного**

Статическое поле принадлежит классу и разделяется всеми экземплярами, тогда как экземплярное поле принадлежит конкретному объекту.

**27. Работа статических конструкторов**

Статические конструкторы вызываются один раз при первом доступе к статическим членам класса и используются для инициализации статических данных.

**28. Поверхностное vs глубокое копирование**

* **Поверхностное копирование**: Копирует только ссылки на объекты, а не сами объекты.
* **Глубокое копирование**: Создаёт полную копию объекта, включая все его вложенные объекты.

**29. Равенство vs тождество объектов**

* **Равенство**: Сравнение значений объектов.
* **Тождество**: Сравнение ссылок на объекты (являются ли они одним и тем же экземпляром).

**30. Частичные классы и частичные методы**

* **Частичные классы**: Позволяют разбивать реализацию класса на несколько файлов.
* **Частичные методы**: Методы, которые могут быть определены в одном из частичных классов, но не обязательно должны иметь реализацию.

**31. Результат выполнения кода**

В результате выполнения кода из первого фрагмента на консоль будет выведен тип объекта, полученного с помощью age.GetType(), то есть "System.Type".

**32. Результат выполнения кода**

В результате выполнения кода из второго фрагмента на консоль будет выведена строка "a = 1, b = 2", которая формируется внутри метода Console.WriteLine().

**33. Неверный конструктор**

Какой из конструкторов задан неверно?  
Согласно коду, неверным является конструктор public A(int someI) { } //3, так как он имеет параметр someI, но не использует его.

**34. Количество аргументов**

Пусть задан следующий класс.  
Сколько аргументов может быть задано при вызове конструктора данного класса?  
В классе Motorcycle определен конструктор с двумя параметрами: int intensity и string name. Поэтому при вызове конструктора можно передать два аргумента.

**35. Создание объектов**

Почему не удается создать объект класса A?  
Согласно коду, нельзя создать объект класса A, потому что конструктор A(int a, int b) является приватным.

**36. Результат выполнения кода**

Что будет выведено в консоль при выполнении данной программы?  
При выполнении программы в консоль будут выведены строки "A static" и "A". Это происходит потому, что в классе A определены два статических метода, которые выводят эти строки.

**37. Ошибка компиляции**

Какая строка приведенного далее класса вызовет ошибку компиляции?  
В приведенном классе Points ошибка компиляции возникнет на строке public static readonly Int32 b = new Int32();. Это связано с тем, что нельзя инициализировать readonly поле с помощью new Int32(), вместо этого необходимо использовать константное значение.