1. Свойства внутренних и вложенных классов:

- Внутренние классы(private)

- Имеют доступ ко всем членам содержащего класса, включая приватные.

- Не могут быть статическими.

- Могут использоваться для логической группировки классов.

- Вложенные классы:

**Вложенный класс** в C# — это класс, определённый внутри другого класса. Он может быть объявлен с любым модификатором доступа, что позволяет управлять его доступностью как изнутри, так и снаружи внешнего класса.

**- Может быть статическим**: Вложенный класс может быть статическим, что позволяет использовать его без создания экземпляра внешнего класса. Статический вложенный класс имеет доступ только к статическим членам внешнего класса..

- Могут содержать статические и нестатические члены.

**- Доступ к внешнему классу**: Вложенный класс может получить доступ к членам внешнего класса (включая private), но для этого нужен экземпляр внешнего класса, как в методе DisplayOuterField.

- Часто используются для создания вспомогательных классов.

2. Статический класс и его свойства:

* прямой потомок System.Object
* экземпляры такого класса создавать запрещено
* не должен реализовывать никаких интерфейсов (не вызвать)
* нельзя использовать в качестве поля, параметра метода или локальной переменной
* от него запрещено наследовать
* все элементы такого класса должны явным образом объявляться с модификатором static
* может иметь статический конструктор
* Компилятор не создает автоматически конструктор по умолчанию

3. Назначение перегрузки операторов:

- Позволяет определять, как операторы (например, +, -, \*, /) работают с пользовательскими типами данных, что делает код более читаемым и интуитивно понятным.

4. Использование ключевого слова operator:

- Ключевое слово operator используется для объявления перегруженного оператора в классе или структуре.

5. Операции, которые нельзя перегружать в C#:

- . (точка), :: (двойное двоеточие), ?: (тернарный оператор), присваивание (=), условный доступ (?.),new (оператор создания объекта), sizeof, typeof, checked, unchecked.

6. Можно ли перегрузкой отменить очередность выполнения операции?

- Нет, перегрузка операторов не может изменить приоритет операций.

7. Истинно ли следующее утверждение: операция >= может быть перегружена?

**Да**, операция >= может быть перегружена, но только в паре с операцией <=.

8. Сколько аргументов требуется для определения перегруженной унарной операции?

- Для унарной операции требуется один аргумент.

9. Можно ли перегружать операцию []?

Да, можно перегружать индексатор (операция []) через определение индексаторов с ключевым словом this. Например:

10. Можно ли перегружать операцию ->?

Нет, оператор -> в C# не существует и, соответственно, не может быть перегружен.

11. Пример оператора приведения типа:

public static explicit operator int(MyClass obj)

{

return obj.Value;

}

MyClass myObj = new MyClass();

int value = (int)myObj; // Явное преобразование

12. Что такое метод расширения? Как и где его можно использовать?

Методы расширения (extension methods) позволяют добавлять новые методы в уже существующие **типы** без создания нового производного класса.

Обычно используется для улучшения функциональности стандартных классов или интерфейсов, например, для работы с коллекциями или добавления новых методов к стандартным типам.

* 1) Методы расширения должны быть объявлены в статическом необобщенном классе (первого уровня)
* 2) this перед первым аргументом и только один
* 3) надо помнить, что метод расширения никогда не будет вызван, если он имеет ту же сигнатуру, что и метод, изначально определенный в типе.

13. Форма преобразования в приведенном коде:

Это **неявное (implicit)** преобразование типа. Оно используется, когда необходимо выполнить преобразование автоматически, без явного указания.

 14. Верные утверждения о методах расширения:

- 1) получает доступ к public членам расширяемого класса..

15. Неверное правило перегрузки операторов для C#:

- 4) должны объявляться как protected (это неверно; перегруженные операторы могут быть объявлены с любым модификатором доступа).

Console.WriteLine(collection3 > collection);//не является подмножеством

Console.WriteLine(collection4 < collection2);//является подмножеством