ТЕОРИЯ

**Индексаторы** позволяют индексировать объекты и обращаться к данным по индексу. Фактически с помощью индексаторов мы можем работать с объектами как с массивами. По форме они напоминают свойства со стандартными блоками get и set, которые возвращают и присваивают значение. В отличие от свойств индексатор не имеет названия. Вместо него указывается ключевое слово **this.**

**Перегрузка –** объявление новых операций для типа. Операции должны быть открытыми и статическими, должны быть совпадения типов. Позволяет определить поведение операторов для пользовательских типов данных. Это позволяет использовать знакомые операторы, такие как +, -, \*, / и другие, для выполнения операций с объектами.

Перегрузить можно: +, -, !, ++, --, true-false, >, <, >=, <= и др.

Нельзя: (), +=, -=, &&, ||, =, ., ?:

**Вложенный класс** (nested class) - это класс, объявленный внутри другого класса. Вложенный класс имеет доступ к членам внешнего класса, включая приватные члены, и может быть использован внутри внешнего класса, а также за его пределами. Логическая связь, упрощение доступа, улучшение организации кода.

**Методы расширения** (extension methods) позволяют добавлять новые методы в уже существующие типы без создания нового производного класса. Эта функциональность бывает особенно полезна, когда нам хочется добавить в некоторый тип новый метод, но сам тип (класс или структуру) мы изменить не можем, поскольку у нас нет доступа к исходному коду. Собственно метод расширения - это обычный статический метод, который в качестве первого параметра всегда принимает такую конструкцию: this имя\_типа название\_параметра, то есть в нашем случае this string str. Если метод расширения имеет ту же сигнатуру (имя и параметры) как и метод, изначально определенный в типе, то метод расширения не будет вызываться, и вместо него будет вызываться оригинальный метод, определенный в типе. Первым параметром метода расширения указывается тип, к которому метод будет добавлен. Этот параметр должен быть модифицирован ключевым словом this, за которым следует тип данных.

ВОПРОСЫ

1. Свойства внутренних и вложенных классов:

Внутренний класс (inner class):

* имеет доступ ко всем членам внешнего класса, включая его закрытые (private) члены
* может содержать свои собственные свойства, поля, методы и события.
* объявляется внутри другого класса и может быть помечен модификаторами доступа(`public`, `private`, `protected`, `internal` или `protected internal`)

Вложенный класс (nested class):

* внутри другого класса, но не имеет доступа к закрытым (private) членам внешнего класса
* собственные свойства, поля, методы и события
* тоже модификаторы

2. Статический класс (static class) в C# - это класс, который не может быть создан как объект, и все его члены должны быть статическими. Используется для группировки связанных методов и свойств, которые могут быть вызваны без создания экземпляра класса. Не может иметь экземпляров.

3. Перегрузка позволяет определить поведение операторов для пользовательских типов данных.

4. Ключевое слово `operator` указывает на то, что метод будет перегружать оператор.

5. В C# нельзя перегружать следующие операции:

- Присваивание (=)

- Вызов метода (.)

- Условные логические операторы (&&, ||)

- Тернарный условный оператор (?:)

- Членское разрешение (::)

- Приведение типов (as, is)

- Создание экземпляра (new)

6. Нет. Перегрузка определяет способ, как оператор будет работать с пользовательским типом данных, но не изменяет общую логику выполнения операций.

7. Да, операция >= может быть перегружена в C#.

8. Унарная – один, бинарная – два. Один из параметров – тип определения оператора.

9. Да, операцию [] можно перегружать в C#. Она позволяет обращаться к объекту с помощью индекса, подобно доступу к элементам массива.

10. В C# операцию -> нельзя перегружать. Она используется в указателях и недоступна для пользовательской перегрузки.

11. Пример оператора приведения типа:

|  |
| --- |
| public static explicit operator int(MyClass obj)  {  return obj.Value;  } |

Перегружаем оператор приведения типа `explicit operator int`, который позволяет явно преобразовывать объекты типа `MyClass`в целочисленные значения типа `int`.

12. **Методы расширения** (extension methods) позволяют добавлять новые методы в уже существующие типы без создания нового производного класса. Эта функциональность бывает особенно полезна, когда нам хочется добавить в некоторый тип новый метод, но сам тип (класс или структуру) мы изменить не можем, поскольку у нас нет доступа к исходному коду.

13. В представленном фрагменте кода определен оператор преобразования типа из класса Point3D в класс Point2D. Ключевое слово implicit указывает на неявное преобразование типа, что означает, что преобразование будет выполняться автоматически без явного вызова оператора.

14. Метод расширения может получать доступ к public членам расширяемого класса.

15. Должны объявляться как protected.