Перечислите свойства внутренних и вложенных классов?

* Вложенный класс может быть объявлен как *частный, общедоступный, защищенный, внутренний, защищенный внутренний или частный защищенный* .
* Внешнему классу не разрешен прямой доступ к внутренним членам класса
* Вам разрешено создавать объекты внутреннего класса во внешнем классе.
* Внутренний класс может получить доступ к статическому члену, объявленному во внешнем классе
* Внутренний класс может получить доступ к нестатическому члену, объявленному во внешнем классе
* Область вложенного класса ограничена областью его вложенного класса.
* По умолчанию вложенный класс является закрытым.
* В C # пользователю разрешено наследовать класс (включая вложенный класс) в другой класс.
* В C # пользователю разрешено наследовать вложенный класс от внешнего класса.
* Внутренний класс ведет себя как обычный класс за тем исключением, что его объекты могут быть созданы только внутри внешнего класса.
* Внутренний класс имеет доступ ко всем полям внешнего класса, в том числе закрытым с помощью модификатора private.
* Аналогично внешний класс имеет доступ ко всем членам внутреннего класса, в том числе к полям и методам с модификатором private.
* Ссылку на объект внешнего класса из внутреннего класса можно получить с помощью выражения Внешний\_класс.this, например, Person.this.
* Объекты внутренних классов могут быть созданы только в том классе, в котором внутренние классы опеределены. В других внешних классах объекты внутреннего класса создать нельзя.
* Еще одной особенностью внутренних классов является то, что их можно объявить внутри любого контекста, в том числе внутри метода и даже в цикле

Что такое статический класс и какие у него свойства?

 Статические вложенные классы позволяют скрыть некоторую комплексную информацию внутри внешнего класса.  
Определение класса статическим, означает то, что этот класс является набором только статических сущностей (полей, методов, свойств и т.п.), и, **внимание**, создавать объекты такого класс нельзя!

Кроме этого, на статический класс, накладываются как минимум следующие ограничения:

* статический класс не может служить базовым для других классов, а так же, сам не может быть наследником какого-либо произвольного класса;
* статический класс не может содержать сущности с атрибутами доступа **protected** и protected **internal**(да они и не нужны, все равно наследование запрещено);
* класс не может иметь перегруженных операций.

1. Каково назначение перегрузки операторов?

Перегрузка операций – способ объявления новых операций для типа.  
**public static возвр\_тип operator оператор(параметры) {..}**

Перегруженные операторные методы дб:  
1) открытыми  
2) тип одного из парам. или возвр. значения совпадал с типом, в котором   
определен операторный метод

*class BigInt {  
 public int Value { get; set; }  
 public static BigInt operator +(BigInt a, BigInt b) {  
 return new BigInt  
 {Value = a.Value + b.Value };  
 }  
 public static bool operator >)BigInt a, BigInt b) {  
 if (a.Value > b.Value) return true;  
 else return false;  
 }  
} …  
BigInt x = new BigInt { Value = 50 };  
BigInt y = new BigInt { Value = 105 };  
Console. WriteLine(x>y); //false  
Console.WriteLine((a+b).Value); //155*

Подлежат перегрузке:  
+, -, !, ++, --  
true, false (попарно)  
+, -, \*, /, %, &, |, ^, <<, >>  
==, !=, <, >, <=, >= (перегрузка парами)

2. Как используется ключевое слово operator?

**public static возвр\_тип operator оператор(параметры) {..}**

3. Какие операции нельзя перегружать в C#?

[] (но есть индексатор)  
() (можно опр. новые оп-ры преобразования)  
+=, -=, \*=, /=… (получ. авто- при перегрузке бинарной операции)  
&&, ||  
=, ., ?:, ??, ->, =>, f(x), as, checked, unchecked, default, delegate, is, new, sizeof, typeof

4. Можно ли перегрузкой отменить очередность выполнения операции?

**перегрузка** операторов не изменяет порядок **выполнения** **операций** и их приоритет;

5. Истинно ли след. утверждение: операция >= может быть перегружена

да

6. Сколько аргументов требуется для определения перегруженной унарной операции?

1 – объект класса, в кот. данный оператор определен

7. Можно ли перегружать операцию []?

нет, но есть индексатор

8. Можно ли перегружать операцию ->?

нет

9. Приведите пример оператора приведения типа

преобразует объект исходного класса в другой тип

*class Point3D {  
 public int x, y;  
 //НЕЯВНОЕ преобр. 2D -> 3D:  
 public static* ***implicit*** *operator Point3D (Point2D p2d) {  
 Point3D p3d = new Point3D();  
 p3d.x = p2d.x;  
 p3d.y = p2d.y;  
 return p3d;  
 }  
….. p2d = (Point2D)p3d;*

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

позвол. добавлять новые методы в уже существующие типы   
без создания нового производного класса

1) создать статич. класс  
2) в него статич. метод

12. Верно? Метод расширения может:  
1) получать доступ к public членам расширяемого класса  
2) получать доступ к protected членам расширяемого класса  
3) получать доступ к internal членам расширяемого класса  
4) быть объявлен в любом классе  
5) быть без параметров

13. Неверное правило перегрузки операторов для C#:  
1) префиксные операции ++ и – перегружаются парами  
2) операции сравнения перегружаются парами: == и !=; < и >; <= и >=  
3) перегруженные операции обязаны возвращать значения  
4) должны объявляться как ~~protected~~ **public и static**  
5) true и false можно перегружать

Класс - однонаправленный список List.  
Дополнительно перегрузить следующие операции:  
! – инверсия элементов;  
+ - объединить два списка;  
= = - проверка на равенство;  
< - добавление одного списка к другому.

Методы расширения:  
1) Усечение строки до заданной длины  
2) Сумма элементов списка\

13.Пусть дан фрагмент кода определения оператора преобразования типа. Определить форму преобразования.

public static implicit operator Point2D(Point3D a) {/\* код\*/;}

После модификаторов **public static** идет ключевое слово explicit (если преобразование явное, то есть нужна операция приведения типов)/implicit (если преобразование неявное)  
Затем идет ключевое слово operator и далее возвращаемый тип, в который надо преобразовать объект. В скобках в качестве параметра передается объект, который надо преобразовать.

int x = 50;

byte y = (byte)x; // явное преобразование от int к byte

int z = y;  // неявное преобразование от byte к int