1. Что такое делегат? Как определить делегат?

*Делегат* – объект, предназначенный для хранения ссылок на методы (указатель на методы) и можно вызвать данные методы.

public delegate void D(int i)  
class Class1  
 private static void Hello(int i) { }  
 static void Main()  
 D del //создаём переменную делегата  
 del = Hello //присваиваем переменной адрес метода  
. //del = new D(Hello)  
 del.Invoke(4) //вызываем метод  
 //del(4)

delegate void Message(); - т.е. делегат может указывать на любой метод, который не принимает никаких параметров и ничего не возвращает.

2. Назначение делегатов.

1) возможности определять вызываемый метод не при компиляции, а динамически во время выполнения программы;

2) обеспечения связи между объектами по типу «источник —наблюдатель»;

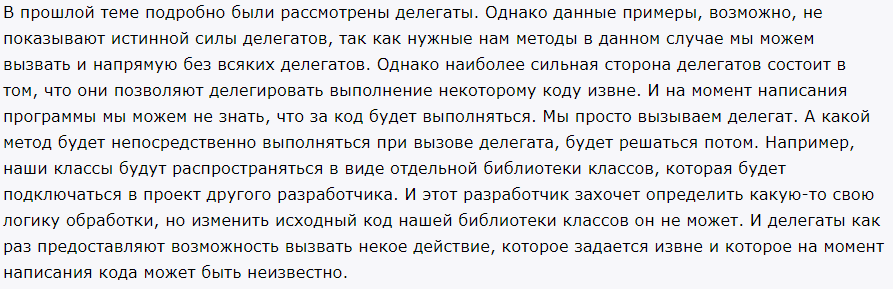
3) создания универсальных методов, в которые можно передавать другие методы;

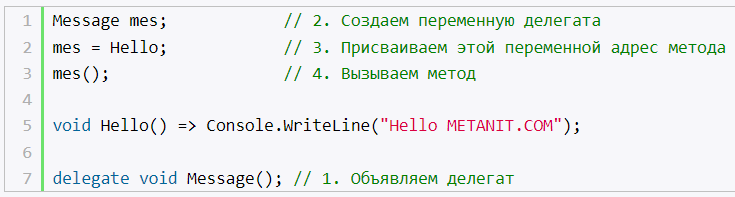
4) поддержки механизма обратных вызовов.

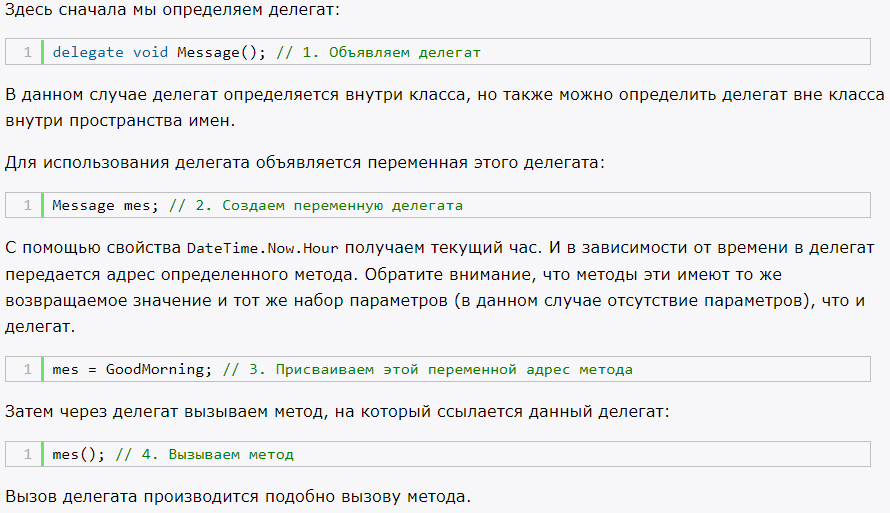
Передача методов как параметры другим методам, создание событий, обратных вызовов.

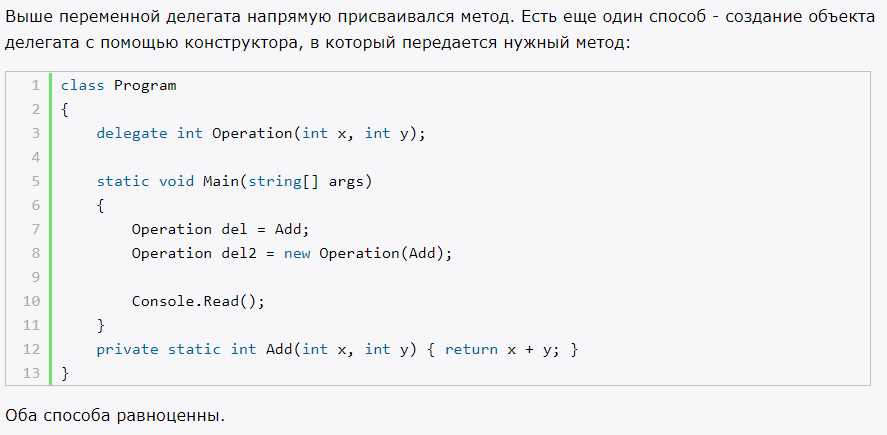
MyDelegate myDelegate = SomeMethod;

myDelegate += AnotherMethod;



3. Какие есть способы присваивания делегату адреса метода? 

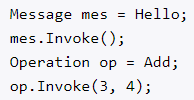


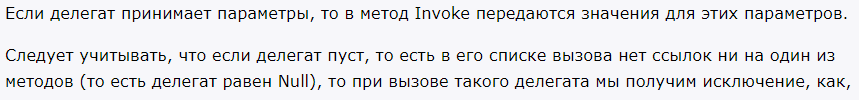


4. Поясните назначение метода Invoke.

Вызов делегата

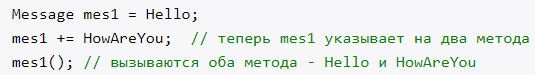


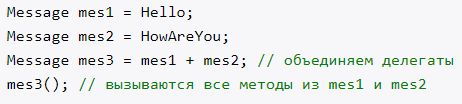


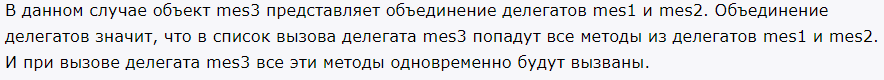


5. Что такое групповая адресация делегата?

Создание списка, или цепочки вызовов, для методов, кот. вызываются авто- при обращении к делегату







6. Как создать событие?

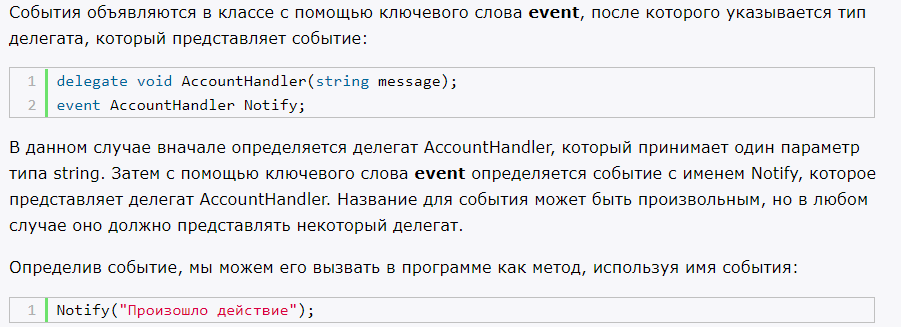
**Событие** – элемент класса, позволяет ему посылать другим объектам уведомления об изменении своего состояния.

1) описание делегата, задающего сигнатуру обработчиков событий  
2) описание события  
3) описание метода (методов), инициирующие событие

7. Как события связаны с делегатами? Опишите и поясните схему взаимодействия.

События построены на основе делегатов: с помощью делегатов вызываются методы-обработчики событий.





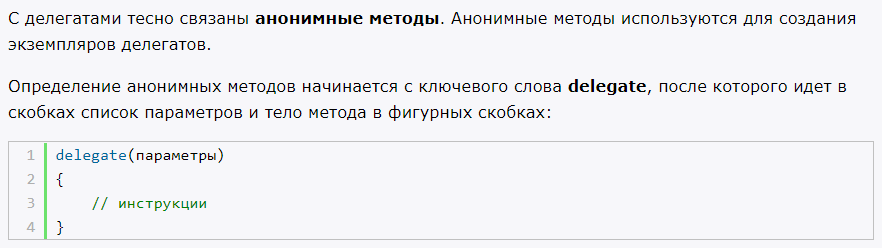
8. Что такое лямбда-выражения? Приведите пример лямбда-выражения с несколькими параметрами.

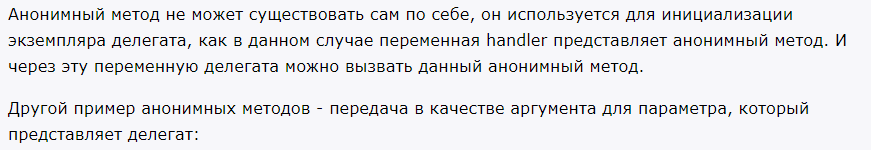
- Упрощённая запись анонимных методов (нет return)

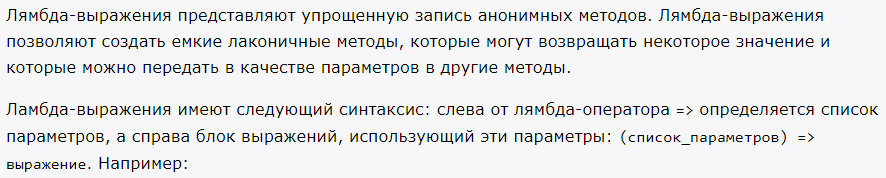
(x,y) => x + y;  
i => i \* i

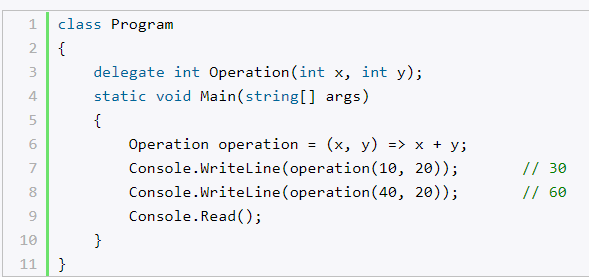
Анонимные функции – безымянный кодовый блок, передаваемый конструктору делегата

delegate int Summator(int b);  
static int result = 0;  
Summator someDelegat = delegate (int number)  
{  
 for (i=0; i<=number; i++)  
 result +=i; //захват переменной  
 return result;  
}







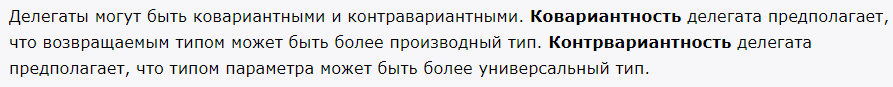


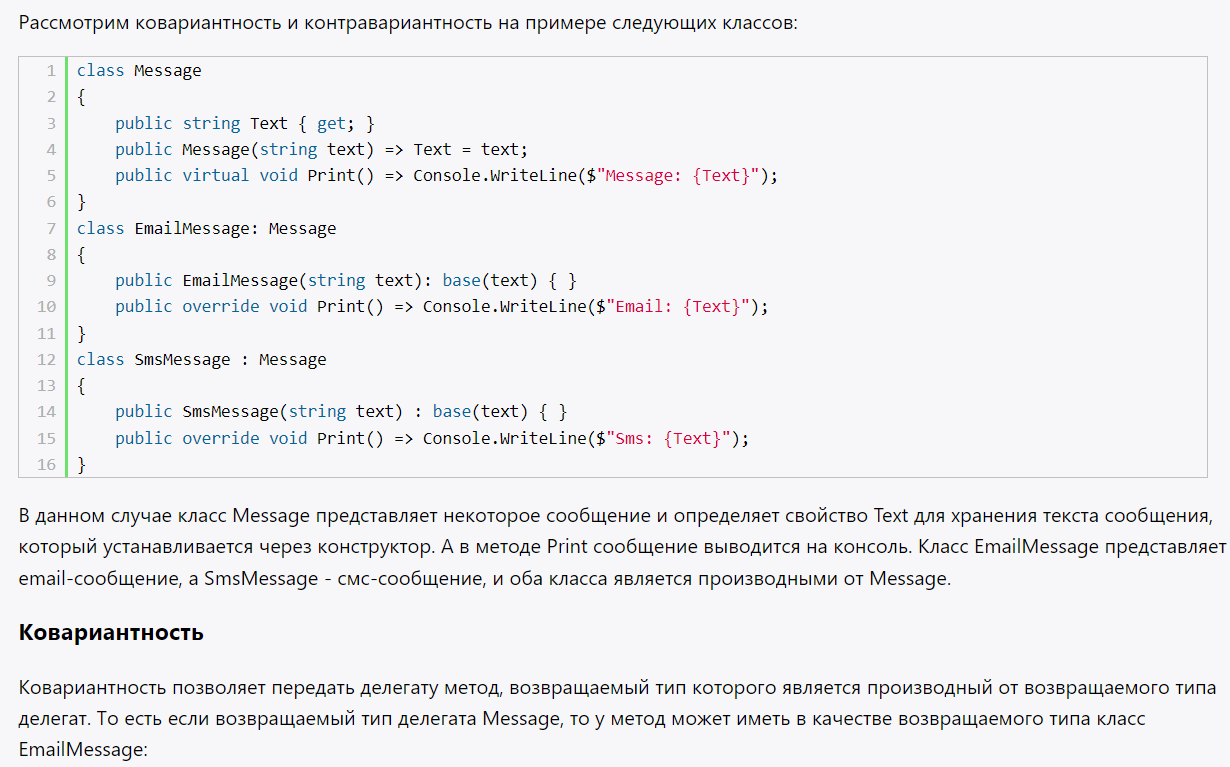
9. Что такое ковариантность и контравариантность делегатов? Что это дает?

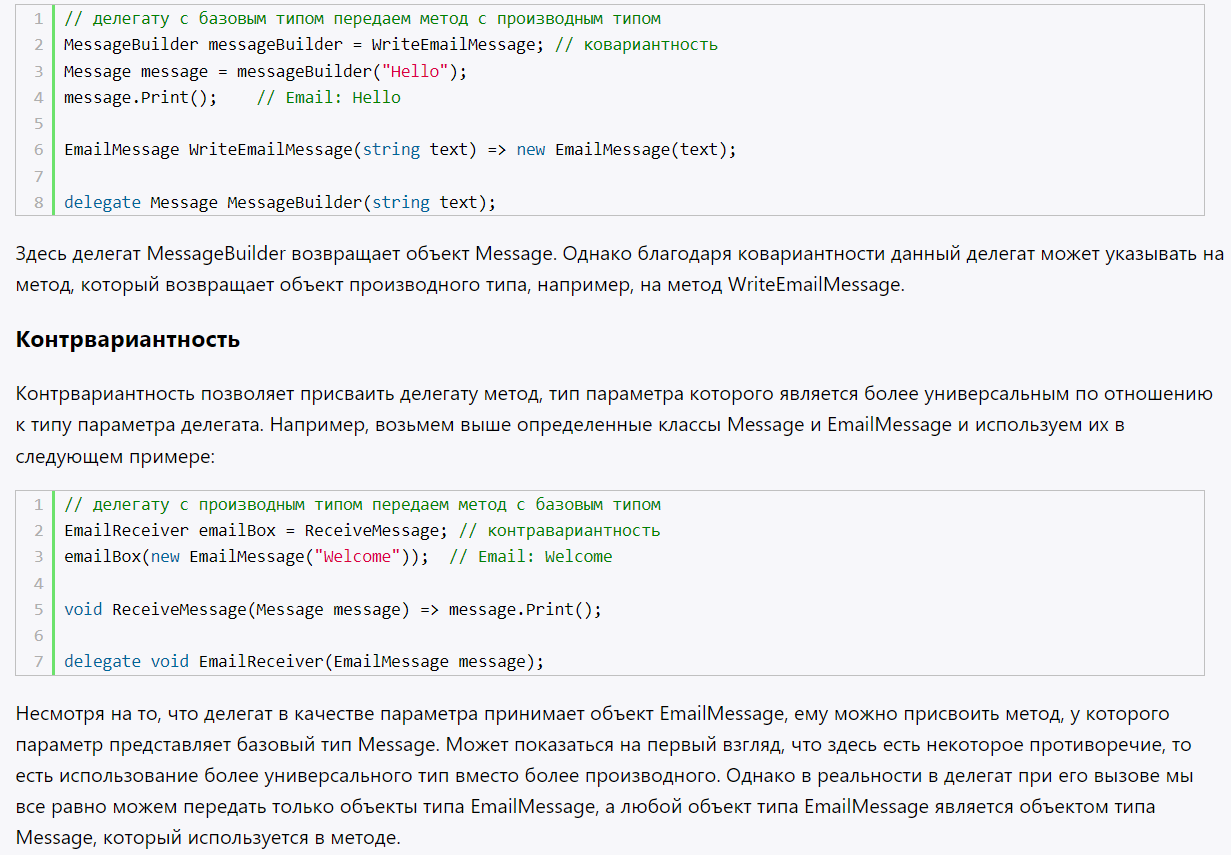
Ковариантность позволяет возвращать из метода объекта, тип которого является производным от типа, возвращаемого делегата.

Контравариантность предполагает возможность использования метода объекта, тип которого является универсальным по отношению к типу представителя.

Понятия ковариантности и контравариантности связаны с использованием в приложении вместо некоторого типа другой тип, который находится ниже в иерархии наследования.







10. Поясните разницу между встроенными делегатам Action и Func.

Action – принимает параметры, возвращает void  
Func – назначает значения типа int

**Action**, который не принимает никаких параметров и ничего не возвращает.

