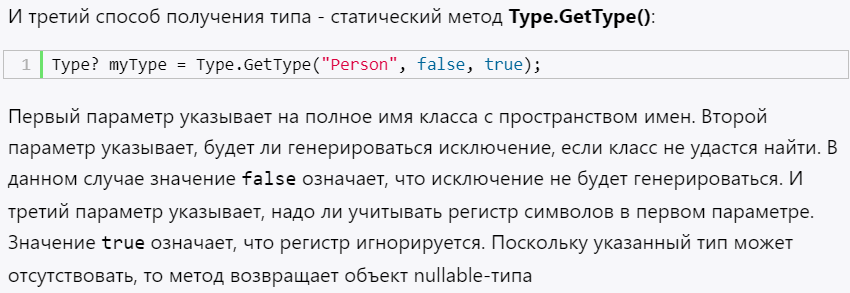
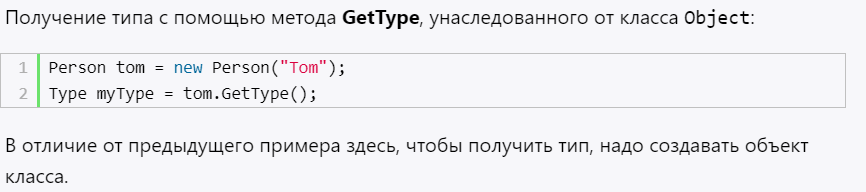
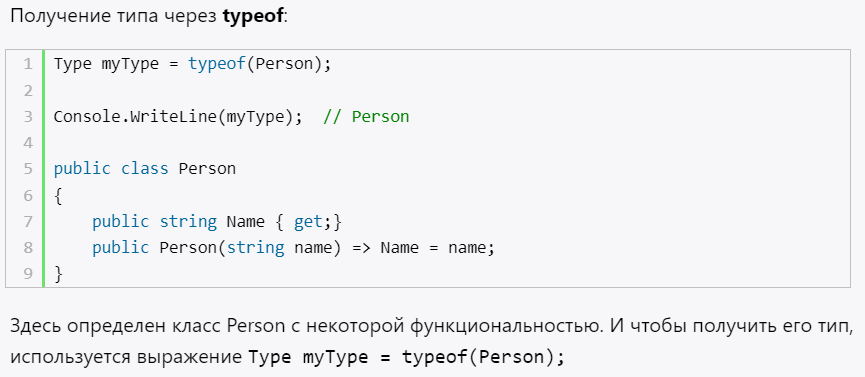
1. Что такое Рефлексия?

**Рефлексия** – процесс выявления типов во время выполнения приложения.

2. Какие способы получения объекта типа Type. Приведите три способа получения типа.

1) Type t = admin.GetType (); - С помощью метода GetType() класса Object  
2) Type t1 = Type.GetType (“System.Int32”); - статический метод GetType  
3) Type t2 = typeof (Point) – с помощью оператора typeof



3. Охарактеризуйте классы из пр-ва имен System.Reflection

**Assembly:** сборка, манипулирование этой сборкой

**AssemblyName:** информация о сборке

**MemberInfo:** базовый абстрактный класс, определяющий общий функционал для классов EventInfo, FieldInfo, MethodInfo и PropertyInfo

**EventInfo:** информация о событии

**FieldInfo:** информация о поле

**MethodInfo:** информация о методе

**PropertyInfo:** информация о свойстве

**ConstructorInfo:** информация о конструкторе

**Module:** доступ к определенному модулю внутри сборки

**ParameterInfo:** информация о параметре метода

4. Как можно использовать класс System.Type? Перечислите его свойства и методы

**SystemType** – класс, позволяет получить информацию о членах типа.Класс Type представляет изучаемый тип, инкапсулируя всю информацию о нем. С помощью его свойств и методов можно получить эту информацию.

**Методы:**  
**FindMembers()** – возвращает массив объектов MemberInfo данного типа  
**GetConstructors()** – возвр. все конструкторы данного типа в виде набора объектов ConstructorInfo  
**GetEvents()** – возвр. все события данного типа в виде массива объектов EventInfo  
**GetFields()** – возвр. все поля данного типа в виде массива объектов FieldInfo  
**GetInterfaces()** – получает все реализуемые данным типом интерфейсы в виде массива объектов Type  
**GetMembers()** – возвр. все члены типа в виде массива объектов MemberInfo  
**GetMethods()** – получает все методы типа в виде массива объектов MethodInfo  
**GetProperties()** –получает все свойства в виде массива объектов PropertyInfo

**Свойства:**

**Name** – возвращает имя типа

**Assembly –** возвращает название сборки, где определен тип

**Namespace** – возвращает название пространства имен, где определен тип  
**IsAbstract** – возвращает тру, если тип абстрактный  
**IsArray** – возвращает тру, если тип - массив  
**IsClass** – возвращает тру, если тип - класс  
**IsEnum** – возвращает тру, если тип - перечисление  
**IsInterface** – возвращает тру, если тип – интерфейс

(а еще IsValueType – структура ли, FullName – полное имя типа)

5. Что такое позднее (раннее) связывание?

Позднее связывание позволяет создавать экземпляры некоторого типа, а также использовать его во время выполнения приложения с помощью динамической загрузки. Позволяет создавать расширяемые приложения, когда дополнительный функционал программы неизвестен и его могут разработать и подключить сторонние разработчики.

Раннее: работа с объектами и их членами решается по время компиляции. Используется, когда типы известны заранее. Программный код формируется на основе известной информации о типе (класс) ссылки. Как правило, это ссылка на базовый класс в иерархии классов.

6. Как динамически загрузить сборку в приложение?

Надо использовать статические методы Assembly. LoadFrom() или Assembly. Load(). Метод LoadFrom(«путь к сборке.dll») принимает в качестве параметра путь к сборке.

7. Что такое позднее (раннее) связывание?

8. Для чего предназначены BindingFlags? Какую комбинацию флагов необходимо использовать, чтобы иметь возможность получать приватные члены класса

Это перечисление флагов, которые определяют область поиска для получения членов типа с использованием рефлексии.

BindingFlags.NonPublic | BindingFlags.Instance

