1. Что такое рефлексия?

Рефлексия представляет собой **процесс выявления типов во время выполнения приложения**

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

способа получения типа.

**с помощью оператора typeof, с помощью метода GetType() класса Object и применяя статический метод Type.** **GetType()**

3. Охарактеризуйте классы из пространства имен System.Reflection.

**Assembly: класс, представляющий сборку и позволяющий манипулировать этой сборкой**

**AssemblyName: класс, хранящий информацию о сборке**

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

**EventInfo: класс, хранящий информацию о событии**

**FieldInfo: хранит информацию об определенном поле типа**

**MethodInfo: хранит информацию об определенном методе**

**PropertyInfo: хранит информацию о свойстве**

**ConstructorInfo: класс, представляющий конструктор**

**Module: класс, позволяющий получить доступ к определенному модулю внутри сборки**

**ParameterInfo: класс, хранящий информацию о параметре метода**

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

и методы.

**Класс Type представляет изучаемый тип, инкапсулируя всю информацию о нем. С помощью его свойств и методов можно получить эту информацию. Некоторые из его свойств и методов:**

**Метод FindMembers() возвращает массив объектов MemberInfo данного типа**

**Метод GetConstructors() возвращает все конструкторы данного типа в виде набора объектов ConstructorInfo**

**Метод GetEvents() возвращает все события данного типа в виде массива объектов EventInfo**

**Метод GetFields() возвращает все поля данного типа в виде массива объектов FieldInfo**

**Метод GetInterfaces() получает все реализуемые данным типом интерфейсы в виде массива объектов Type**

**Метод GetMembers() возвращает все члены типа в виде массива объектов MemberInfo**

**Метод GetMethods() получает все методы типа в виде массива объектов MethodInfo**

**Метод GetProperties() получает все свойства в виде массива объектов PropertyInfo**

**Свойство Name возвращает имя типа**

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

**Свойство Namespace возвращает название пространства имен, где определен тип**

**Свойство IsArray возвращает true, если тип является массивом**

**Свойство IsClass возвращает true, если тип представляет класс**

**Свойство IsEnum возвращает true, если тип является перечислением**

**Свойство IsInterface возвращает true, если тип представляет интерфейс**

5. Что такое позднее и раннее связывание?

**Раннее связывание** – связанное с формированием кода на этапе компиляции. При **раннем связывании**, программный код формируется на основе известной информации о типе (класс) ссылки. Как правило, это ссылка на базовый класс в иерархии классов. **Позднее связывание** – связанное с формированием кода на этапе выполнения

6. Как динамически загрузить сборку в приложение?

Чтобы динамически загрузить сборку в приложение, надо **использовать статические методы Assembly.** **LoadFrom() или Assembly.** **Load()**. Метод LoadFrom() 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

Type tt = typeof(Int32);

var bf = BindingFlags.Public |

BindingFlags.NonPublic |

BindingFlags.Static |

BindingFlags.Instance;

FieldInfo[] fi = tt.GetFields(bf);