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().

1. 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 System.Type в языке C# представляет метаданные типа данных во время выполнения программы. Он позволяет получать информацию о типе, такую как его имя, сборку, базовые классы, интерфейсы, свойства, методы и другие связанные с типом данные.

* 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
  1. Что такое позднее и раннее связывание?

Позднее связывание (dynamic binding) - это процесс определения вызываемого метода или свойства во время выполнения программы. В позднем связывании конкретный метод или свойство определяется по типу объекта, с которым происходит взаимодействие, во время выполнения программы.

Раннее связывание (early binding) - это процесс определения вызываемого метода или свойства во время компиляции программы. В раннем связывании конкретный метод или свойство определяются по типу переменной или выражения на этапе компиляции.

* 1. Как динамически загрузить сборку в приложение?

Assembly assembly = Assembly.LoadFrom("путь\_к\_файлу.dll");

8. Для чего предназначены BindingFlags? Какую комбинацию флагов

необходимо использовать, чтобы иметь возможность получать

приватные члены класса?

BindingFlags используются для управления доступом к членам типа (например, полям, свойствам, методам) и определения области поиска.

Чтобы получить приватные члены класса с использованием рефлексии, вам понадобится комбинация флагов BindingFlags.NonPublic и BindingFlags.Instance.