

#8 Обобщения

Задание:

1. Разработайте и реализуйте иерархию наследования (один базовый класс и минимум 2 наследника).
2. Создайте обобщенный класс-коллекцию, применив ограничение параметра на базовый класс.
3. Реализуйте методы добавления, удаления и поиска (по предикату) объектов.
4. Продемонстрируйте работу вашего обобщенного класса с различными членами иерархии.
5. Изучите стандартные обобщенные коллекции .NET:
<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic?view=netframework-4.8>
6. Продемонстрируйте работу с любыми двумя коллекциями, параметризованными вашими классами и любым примитивным типом (**int**, **double**, etc.)

Повышенный уровень:

1. Создайте новый проект типа **Class Library** с именем **AwesomeLibrary** (можете изменить на свое). Вынесите полностью логгер из л.р. #7 в новый проект. Извне проекта должны быть доступны только лишь необходимые объекты.
2. Ознакомьтесь с концепцией рефлексии в .NET:
<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/reflection>
3. Создайте структуру, которая будет хранить информацию об объектах вашего класса:
 - a. Есть ли публичный конструктор? (*bool*)
 - b. Имя типа (*string*)
 - c. Имя сборки, в которой он определен (*string*)
 - d. Имена публичных полей (*IEnumerable<string>*)
 - e. Имена публичных свойств (*IEnumerable<string>*)
 - f. * Имена приватных полей (*IEnumerable<string>*)
 - g. * Имена приватных свойств (*IEnumerable<string>*)
 - h. Имена публичных методов (*IEnumerable<string>*)
 - i. * Имена приватных методов (*IEnumerable<string>*)

Note: * - если приватные члены класса доступны 😊

4. Добавьте два новых свойства, описывающих любую иную информацию об объекте (на ваш выбор).
5. Создайте в вашей библиотеке статический класс **Reflector**, который будет собирать информацию, используя рефлексию.
6. Реализуйте в классе метод **Analyze**, собирающий информацию об объектах вашего класса (используя структуру выше) и выводящий ее в лог.
7. **Reflector** должен работать лишь с интерфейсом **ILogger**. У пользователя должна быть возможность легко переопределить конкретную реализацию логгера, который будет использовать **Reflector**.
8. Реализуйте в классе метод **Save**, собирающий информацию об объектах вашего класса (используя структуру выше) и сохраняющий её в **json**-файл. [Пример](#).

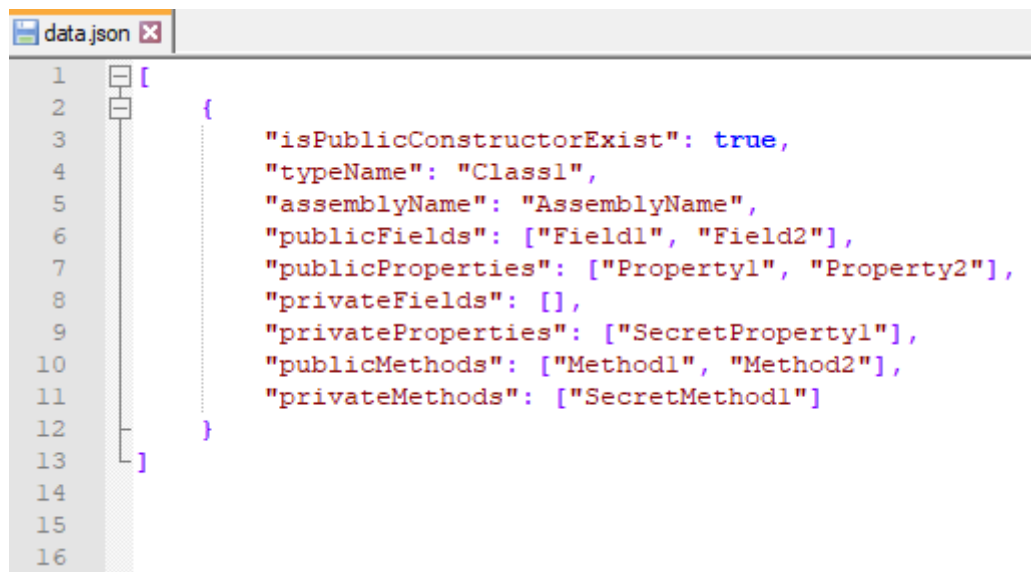
Вопросы:

1. Что такое обобщение?
2. В чем заключаются плюсы и минусы обобщений в .NET
3. Можем ли мы сделать обобщенный метод в обычном классе? Если да, то как? Продемонстрируйте на тестовом классе и методе.
4. Какие существуют ограничения на обобщения? Назовите все.
5. Можно ли наложить несколько ограничений одновременно? Если да, то каким образом?
6. Что обозначает оператор **default**?

Повышенный уровень:

1. Что такое рефлексия?
2. Какую информацию нам может дать использование рефлексии о типе, классе, сборке?
3. Возможно ли создавать новые классы во время выполнения с помощью рефлексии?
4. Охарактеризуйте методы и классы Reflection API, которые вы использовали в работе.
5. Доступны ли **private**-члены класса через рефлексию?

Пример json-файла с данными:



```
1  [
2  {
3      "isPublicConstructorExist": true,
4      "typeName": "Class1",
5      "assemblyName": "AssemblyName",
6      "publicFields": ["Field1", "Field2"],
7      "publicProperties": ["Property1", "Property2"],
8      "privateFields": [],
9      "privateProperties": ["SecretProperty1"],
10     "publicMethods": ["Method1", "Method2"],
11     "privateMethods": ["SecretMethod1"]
12  }
13 ]
14
15
16
```