#10 Коллекции

Задание:

1. Создайте класс по вариантам, содержащий какую-либо полезную нагрузку.
2. Изучите стандартные коллекции .NET:
    a. https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic?view=netframework-4.8
    b. https://docs.microsoft.com/en-us/dotnet/api/system.collections?view=netframework-4.8
    c. https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent?view=netframework-4.8
    d. https://docs.microsoft.com/en-us/dotnet/api/system.collections.specialized?view=netframework-4.8
3. Изучите стандартные интерфейсы для коллекций .NET (те же ссылки, секция **Interfaces**)
4. Познакомьтесь на практике со стандартными коллекциями .NET:
    a. В секции «Варианты», для каждого варианта перечислены 2 класса и/или интерфейса, с которыми необходимо ознакомиться.
    b. Если в условии присутствует класс – необходимо создать объект данной коллекции и продемонстрировать работу с ним.
    c. Если в условии указан интерфейс – необходимо реализовать данный интерфейс на вашем классе.
    d. Если класс и/или интерфейс являются дженериками, используйте объекта вашего класса.

Повышенный уровень:

1. Познакомьтесь с понятием *DI – Dependency Injection*:
   https://ru.wikipedia.org/wiki/Внедрение_зависимости
   https://designpatternsphp.readthedocs.io/ru/latest/Structural/DependencyInjection/README.html
2. Познакомьтесь с nuget-пакетом **Ninject**:
   http://www.ninject.org/
   https://github.com/ninject/Ninject
3. Избавьтесь от синглтонов в конкретных реализациях логгеров.

4. Создайте интерфейс **IReflector** на основе класса **Reflector**. Преобразуйте **Reflector** из статического класса в обычный и добавьте наследование от интерфейса **IReflector**.
5. Добавьте классу Reflector зависимость от **ILogger** через конструктор.
6. Добавьте **Ninject** в проект вашей библиотеки. Создайте статический класс **ModuleLoader** с методом **Load**.
7. Метод должен принимать параметрами объект интерфейса **IKernel** и значение перечисления **LoggerType**.
8. **ModuleLoader** внутри себя должен осуществлять привязку интерфейса **ILogger** к конкретной реализации, в зависимости от значения параметра **loggerType** (*см. метод When интерфейса **IBindingWithSyntax***).
9. **ModuleLoader** внутри себя должен осуществлять привязку **IReflector** к **Reflector**.
10. Используйте привязку как «*singleton*» для логгеров и привязку типа «*scope*» для **Reflector**.
11. В проекте л.р. #10 создайте класс **DIExperiment**, принимающий в конструкторе объекты интерфейсов **IReflector** и **ILogger**.
12. Создайте метод **ReflectorTest**, который принимает объект вашего класса и выводит всю информацию о нем используя метод **Analyze** интерфейса **IReflector**.
13. Создайте метод **LoggerTest**, который принимает объект вашего класса и выводит результат вызова метода **ToString()** на объекте в лог, используя метод Log интерфейса **ILogger**.
14. Добавьте **Ninject** в проект текущей работы (**Note:** версии библиотек **Nuget** в текущем проекте и проекте вашей библиотеки должны совпадать, иначе возникнут проблемы совместимости).
15. Создайте модуль, который наследуется от **NinjectModule**.
16. Создайте объект ядра **IKernel**, загрузите в него все биндинги из текущей сборки, а также вызовите метод **Load** из вашей библиотеки для загрузки ваших привязок.
17. Создайте с помощью **IKernel.Get** объект класса **DIExperiment** и продемонстрируйте выполнение двух его методов.

Вопросы:

1. На какие основные виды/типы делятся все коллекции .NET? Охарактеризуйте каждый из них.
2. Что такое generic-коллекции? Назовите примеры известных вам generic-коллекций.
3. В чем разница между ArrayList и Array?
4. Охарактеризуйте коллекции, которые вы использовали в своем варианта.
5. Чем отличаются коллекции, расположенные в пространстве имен System.Collections.Concurrent?
6. Какое пространство имен необходимо подключить в проект, чтобы иметь возможность использовать generic-коллекции?
7. Что такое наблюдаемая коллекция? Как ее можно использовать?
8. Охарактеризуйте интерфейсы IEnumerator, IEnumerator<T>. В чем отличие назначений интерфейсов IEnumerator и IEnumerable.
9. Поясните принцип работы коллекций:
    a. LinkedList<T>
    b. HashSet<T>
    c. Dictionary<TKey, TValue>
    d. ConcurrentBag<TKey, TValue>
    e. Stack<T>, Queue<T>,
    f. SortedList<T>, SortedList<TKey, TValue>

Повышенный уровень:

1. Что такое **DI**?
2. Расскажите про методы/классы библиотеки **Ninject**, которые вы использовали в работе.
3. Охарактеризуйте принцип работы **Ninject**. Каким образом библиотека находит и загружает кастомные модули?
4. В чем разница между привязкой как «*singleton*» и как «*scope*»

Варианты:

#1:

1. Класс – Автомобиль
2. Коллекции – IList<T>, Dictionary<TKey, TValue>

#2:

1. Класс – Книга
2. Коллекции – IDictionary<TKey, TValue>, List<T>

#3:

1. Класс – Магазин
2. Коллекции – IOrderedDictionary, ConcurrentBag<T>

#4:

1. Класс – Студент
2. Коллекции – Stack, Hashtable

#5:

1. Класс – Дерево
2. Коллекции – IEnumerable<T>, Queue<T>

#6:

1. Класс – Телефон
2. Коллекции – Dictionary<TKey, TValue>, HashSet<T>

#7:

1. Класс – Продукт
2. Коллекции – ConcurrentDictionary<TKey, TValue>, LinkedList<T>

#8:

1. Класс – Вентилятор
2. Коллекции – SortedList<TKey, TValue>, Queue

#9:

1. Класс – Диван
2. Коллекции – List<T>, IProducerConsumerCollection<T>

#10

1. Класс – Дом (помещение)
2. Коллекции – ArrayList, IList<T>

#11

1. Класс – Треугольник
2. Коллекции – ISet<T>, LinkedList<T>

#12

1. Класс – Точка
2. Коллекции – Dictionary<TKey, TValue>

#13

1. Класс – Игра
2. Коллекции – BlockingCollection<T>, IEnumerable<T>

#14

1. Класс – Квадрат
2. Коллекции – IEnumerator, Stack<T>

#15

1. Класс – Фигура
2. Коллекции - List<T>, IList (не дженерик)

#16

1. Класс – Предмет (школьный, университетский)
2. Коллекции – Dictionary<TKey, TValue>, ICollection<T>

#17

1. Класс – Курс
2. Коллекции – ConcurrentDictionery<TKey, TValue>, IList<T>

#18

1. Класс – Покупатель
2. Коллекции – Hashtable, HashSet<T>

#19

1. Класс – Продавец
2. Коллекции – IOrderedDictionary, Queue<T>

#20

1. Класс – Мороженое ☺
2. Коллекции – List<T>, Dictionary<TKey, TValue>