#3. Проектирование типов. Классы

Базовый уровень.

1. Спроектируйте класс (по вариантам), наделив его необходимыми свойствами (1-5). Определите в нем:
    a. Конструктор по умолчанию;
    b. Конструктор с параметрами;
    c. Статический конструктор;
    d. Конструктор копирования;
    e. Переопределить методы класса **Object**: *Equals*, *GetHashCode*, *ToString*;
    f. Деструктор;
    g. Статический метод;
    h. Статическое поле;
    i. Свойства (классическое, автоматическое, **get only**, **set only**);
    j. Поле для чтения;
    k. Методы с использованием **ref**/**out**-параметров.
2. Открытые методы и свойства обязаны обеспечивать проверку валидности вводимых данных.
3. Реализуйте созданный класс как **partial** класс.
4. Реализуйте статический класс, осуществляющий вывод информации о вашем классе на консоль, используя переопределенный метод *ToString* класса.
5. Создайте анонимный тип по примеру вашего класса.

Средний уровень.

1. Создайте класс-коллекцию, оперирующую с объектами вашего класса, в котором определите:
    a. Методы работы с коллекцией (добавление/удаление);
    b. Поиск объекта по какому-либо полю;
    c. Индексатор;
    d. Переопределите методы класса **Object** для класса-коллекции;
    e. Свойство **Count**, выводящее количество объектов в коллекции.
2. Открытые методы и свойства класса обязаны обеспечивать проверку валидности вводимых данных.
3. Реализуйте методы расширения для вашего класса-коллекции:
    a. **Bool**-метод, возвращающий наличие в коллекции объекта с заданным значением поля (любого);

    b. Метод, принимающий параметром объект коллекции и возвращающий новую коллекцию, состоящую из уникальных объектов двух коллекцией: текущей и коллекции-параметра.
4. Создать объект коллекции, продемонстрировать работу с описанными выше методами.

Повышенный уровень.

1. Ознакомиться с паттерном «**Одиночка**» (Синглтон): https://ru.wikipedia.org/wiki/Одиночка_(шаблон_проектирования)
2. Реализовать класс-коллекцию, описанную выше как **синглтон**.
3. Ознакомиться с «паттерном» **Dispose**: https://docs.microsoft.com/en-us/dotnet/standard/garbage-collection/implementing-dispose
4. Реализовать **Dispose** для класса или класса-коллекции.

Вопросы.

1. Что такое ООП? Какие основные принципы ООП?
2. Какой класс .NET является базовым для всех остальных? Охарактеризуйте его открытые методы.
3. Что такое конструктор класса? Какие виды конструкторов существуют в .NET?
4. В чем особенность статического конструктора?
5. Что такое деструктор? Для чего он используется? В какой момент вызывается деструктор?
6. Что такое «свойство класса»? В чем заключается отличие свойства от поля?
7. Перечислите виды свойств в .NET.
8. Что обозначает ключевое слово **readonly**?
9. Что обозначает ключевое слово **static**? В чем заключается особенность статического класса?
10. В чем отличие между **public static readonly** и **public const** членами класса?
11. Что такое модификаторы доступа? Какие модификаторы доступа вы знаете? Охарактеризуйте каждый из них.
12. Какие модификаторы доступа являются дефолтными (по умолчанию) для полей класса? Методов класса? Классов?
13. Что такое **ref**/**out** параметры? В чем их отличие?
14. Что такое **in** параметры?

15. Что обозначает ключевое слово **partial**? Что мы можем пометить как **partial** в .NET? Могут ли быть **partial** методы? Если да, то какие существуют ограничения на подобные методы?
16. Что такое метод расширения? Какие ограничения существуют для создания метода расширения? В каких случаях используются методы расширения?
17. За счет какого механизма происходит переопределение метода? Что обозначают ключевые слова **virtual** и **override**?
18. Что такое индексатор? Как определяется индексатор?
19. Что такое анонимные типы?
20. В чем разница между глубоким (**deep**) и поверхностным (**shallow**) копированием?
21. Что обозначает ключевое слово **sealed**?

Повышенный уровень.

22. Что такое паттерн «Одиночка»? Для чего он применяется? Какие главные характерные черты класса, реализующего данный паттерн?
23. Что такое паттерн **Dispose**? Для чего он применяется?
24. Для чего служит метод *Finalize*? Как он связан с деструктором класса?
25. Можем ли мы переопределить метод *Dispose*? *Finalize*?

Варианты заданий:

1. Дерево
2. Автомобиль
3. Стол
4. Здание
5. Ручка
6. Автобус
7. Товар
8. Телефон
9. Книга
10. Студент
11. Абитуриент
12. Самолет
13. Заказчик
14. Магазин
15. Университет
16. Аккаунт
17. Альбом (музыкальный)
18. Исполнитель
19. Приложение
20. Игровая студия

Ответы на все вопросы есть в документации MSDN. Стартовая точка: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/index