Задание №4. Создание бизнес-логики

Описание задания:

1. Создайте новую ветку в локальном репозитории под названием «features/3_add_data_structures». Перейдите в новую ветку.
2. Удалите в проекте Model класс Class1.
3. Добавьте в проект Model классы бизнес-логики согласно ТЗ своего варианта (кроме ProjectSerializer – он реализуется в следующем задании). Названия файлов должно совпадать с названием классов, включая регистры.
4. Разработайте классы согласно ТЗ: определите в классах поля, свойства, конструктор и, при необходимости, методы.
5. Убедитесь, что поля классов являются строго закрытыми (private), а доступ к значениям полей осуществляется строго через свойства.
6. Реализуйте проверку правильности значений в сеттерах свойств. В случае попытки присвоения в поле неправильного значения, сеттер должен кидать исключение ArgumentException.
7. Конструктор должен инициализировать поля класса строго через свойства класса, а не напрямую в поля.
8. Если классы реализованы правильно, сделайте фиксацию в репозитории. В комментарии к фиксации укажите, что именно было добавлено в код.
9. Убедитесь, что все элементы классов названы правильно, согласно требованиям оформления кода.
10. Добавьте ко всем членам классов xml-комментарии согласно требованиям оформления кода.
11. Проверьте оформление классов: переносы строк, пустые строки, xml-комментарии, грамматические ошибки.
12. В Visual Studio перейдите в ветку develop. Затем выполните слияние (merge) ветки features/3_add_data_structures в ветку develop. Теперь все изменения из ветки с добавленным решением должны переместиться в ветку develop. Убедитесь в этом с помощью GitHub.

Защита задания:

1. Во время защиты студент должен показать проект Model и созданные в нём классы бизнес-логики согласно ТЗ.
2. Классы должны быть названы существительным в единственном числе, с заглавной буквы, название файла должно совпадать с именем класса. Например, правильно – Note, неправильно – note, Notes.
3. Поля классов должны быть закрыты (private), названы в стиле именования camel* с нижнего подчеркивания, существительным в единственном числе (в случае коллекций – в множественном). Например, правильно - _dateOfBirth, неправильно DateOfBirth, dateOfBirth, _date_of_birth и т.д.
4. Свойства классов должны быть названы в стиле именования Pascal** существительным в единственном числе (в случае коллекций – в множественном). Свойство должно именоваться аналогично полю, только в другом стиле написания. Например, если поле называется _dateOfBirth, то соответствующее ему свойство должно называться DateOfBirth, а не Birthday или BirthDate.
5. Методы класса должны именоваться в стиле Pascal от глагола, отвечая на вопрос «Что сделать?», например SortContactsBySurname («Отсортировать контакты по фамилии»), но не SortingContactsBySurname («Сортировка контактов по фамилии»).
6. Если метод реализует проверку данных и возвращает булево значение, он должен именоваться со слова «Is» (например, IsPhoneNumberCorrect).
7. Все классы, а также все члены класса (поля, свойства, конструкторы, методы и т.д.) должны иметь xml-комментарии.
8. Между всеми членами класса должна быть одна пустая строка.
9. Длина строк исходного кода не должна превышать 100 символов.
10. Студент должен показать на GitHub, что при выполнении задания была создана отдельная ветка features/3_add_data_structures, и она была слита в develop.

Пример правильного оформления исходного кода класса:

```
/// <summary>
/// Описывает книгу.
/// </summary>
public class Book
{
    /// <summary>
    /// Название книги.
    /// </summary>
    private string _title;
```

```
        /// <summary>
        /// Год выпуска книги.
        /// </summary>
        private int _year;

        /// <summary>
        /// Возвращает или задает название книги.
        /// </summary>
        public string Title
        {
                get
                {
                        return _title;
                }
                set
                {
                        _title = value;
                }
        }

        /// <summary>
        /// Возвращает или задает год выпуска книги.
        /// </summary>
        public int Year
        {
                get
                {
                        return _year;
                }
                set
                {
                        if (value > 2022)
                        {
                                throw  new  ArgumentException($"Year
must be less than current year."
                                + $"But was {value}");
                        }
                        _year = value;
                }
        }

        /// <summary>
```

```
    /// Возвращает или задает авторов книги.
    /// </summary>
    public List<string> Authors {get; set;} = new
List<string>();

    /// <summary>
    /// Создает экземпляр <see cref="Book">.
    /// </summary>
    public Book(string title, int year, List<string> authors)
    {
            Title = title;
            Year = year;
            Authors = authors;
    }
}
```

* camel – стиль именования в программировании, когда все слова в имени пишутся слитно, каждое новое слово с заглавной буквы кроме первого. Например, surname, phoneNumber, publishingDate. В именовании закрытых полей часто добавляется нижнее подчеркивание в начале, например, _phoneNumber.

** Pascal – стиль именования в программировании, когда все слова в имени пишутся слитно, каждое новое слово с заглавной буквы. Например, Surname, PhoneNumber, PublishingDate.