Работен лист – Уроци 3 и 4

Тема: Коментари, анти-патерни и XML документация (ASP.NET Core MVC)

Клас: 12. Предмет: Софтуерно инженерство (Модул 3)
Проект за упражнения: SchoolLibrary (ASP.NET Core MVC, .NET 8, EF Core + SQLite)

Цели на упражненията

- Разпознаваш вредни коментари и анти-патерни (дублиране, „гробище“, неактуални/подвеждащи коментари).
- Подобряваш четимостта чрез рефакторинг вместо чрез коментари.
- Пишеш кратки и смислени коментари само за „защо“, бизнес правило, ограничение или edge case.
- Документираш публични методи с XML документация (///) и проверяваш резултата чрез IntelliSense.

Инструкции за предаване

1. Работи по екипи по 2-ма или индивидуално (по указание на учителя).
2. Предай: (1) линк към репо/папка с проекта, (2) кратък текстов отчет (1/2 страница) какво рефакторира и защо, (3) screenshot(и) на IntelliSense от XML docs.
3. Не оставяй закоментиран код. Ако пазиш стара версия, използвай Git commit (или копие в отделен файл „_old“ – само ако учителят го позволява).

Урок 3 – Анти-патерни при коментари и рефакторинг

Упражнение 3.1 – Класифициране на коментари (полезен/вреден)

За всеки коментар отбележи дали е полезен или вреден и предложи по-добра алтернатива (ако е вреден).

Коментар / контекст	Полезен или вреден?	Подобрение (ако е вреден)
// return view		
// borrow book		
// NOTE: Business rule – a book can be borrowed only if IsBorrowed == false		
// TODO: Fix later		
// HACK: Temporary workaround for legacy data where Year can be 0		
// This method gets books from database		

Упражнение 3.2 – Рефакторинг: изнасяне на бизнес логика от Controller в Service

Даден е контролер с бизнес логика и излишни коментари. Целта е да го направиш кратък и четим.

Код за рефакторинг:

```
public async Task<IActionResult> Borrow(int id, string student)
{
    // borrow book
    var book = await _db.Books.FindAsync(id);
    if (book == null) return RedirectToAction("Index"); // not found

    // if borrowed do nothing
    if (book.IsBorrowed) return RedirectToAction("Index");

    book.IsBorrowed = true; // set borrowed
    _db.SaveChanges(); // save
    return RedirectToAction("Index");
```

```
}
```

Задачи:

4. Създай интерфейс IBorrowService и клас BorrowService.
5. Премести бизнес логиката за заемане в BorrowService.BorrowAsync(bookId, studentName).
6. Controller action да съдържа само: вход, валидация, извикване на service, избор на резултат (Redirect/NotFound/BadRequest).
7. Премахни излишните коментари. Ако е нужно, остави 1 кратък NOTE за бизнес правило или edge case.
8. Добави минимална валидация: studentName да не е празно (иначе връщай BadRequest или показвай validation message – според вашия проект).

Критерии за успех:

- Controller методът е кратък и четим (без бизнес логика).
- Service методът има ясна отговорност и е лесен за тестване.
- Коментарите добавят стойност (не повтарят кода).

Упражнение 3.3 – Премахване на „гробище“ и смислен TODO/NOTE

Проблемен код:

```
// TODO: fix later
// if (user == null) { ... }
// old logic below:
/*
var x = 5;
DoSomething(x);
*/
```

Задачи:

9. Премахни закоментирания код.
10. Ако остава TODO, направи го конкретен: какво липсва, защо е важно и къде трябва да се добави.
11. Добави NOTE само ако обяснява реално ограничение/причина за решение.

Урок 4 – XML документация (///) и IntelliSense

Упражнение 4.1 – Документиране на Service интерфейс

Попълни XML документация за методите по-долу. Използвай <summary>, <param>, <returns>, а при нужда <remarks> и <exception>.

```
public interface IBorrowService
{
    Task<bool> BorrowAsync(int bookId, string studentName);
    Task<bool> ReturnAsync(int bookId);
}
```

Минимум изисквания за документацията:

- В <summary> опиши какво прави методът с 1 изречение.
- В <param> обясни смисъла на параметрите (не повтаряй само името им).
- В <returns> опиши кога връща true и кога false.
- В <remarks> опиши поне 1 бизнес правило (напр. „книга може да се заеме само ако не е заета“).

Упражнение 4.2 – XML docs за MVC actions

Добави XML документация към поне 3 action-а в BooksController или BorrowController.

- Примерни action-и: Details(int id), Borrow(int id), Return(int id).
- Включи в документацията какво става при: липсващ запис (NotFound), невалиден вход (BadRequest/validation), успешно действие (Redirect/View).

Упражнение 4.3 – Генериране на XML документационен файл

Активирай генерирането на XML documentation file и провери IntelliSense.

12. В .csproj включи: <GenerateDocumentationFile>true</GenerateDocumentationFile>.
13. Build проекта и провери дали в output директорията се генерира .xml файл.
14. Покажи IntelliSense над документиран метод (скрийншот).

Бонус мини-задачи (по избор)

- Добави <example> за BorrowAsync с примерни входове/изход.
- Добави <exception> (само ако в service хвърляш ArgumentException при празно studentName).
- Премести повтаряща се логика в помощен метод и документирай само публичния метод (не всеки private).

Чеклист преди предаване

- Няма излишни/подвеждащи коментари.
- Няма закоментиран код („гробище“).
- Публичните service методи имат XML docs (summary/param/returns).
- Поне 3 MVC action-а имат смислени XML docs.
- README/архитектура (ако проектът ви го изисква) са актуални.