Выполните все приведенные ниже задания:

*Для этого вам понадобится загрузить шаблон* [AsyncAwait.Task*s*](https://epam.sharepoint.com/:u:/r/sites/NetMentoringprogramA2Belarus/Shared%20Documents/Module%202.%20Asynchronous%20programming/Task/AsyncAwait.Tasks.zip?csf=1&e=zBtGtP)*.*

1. **Asynchronous calculation and cancellation tokens**

(AsyncAwait.Task1.CancellationTokens.csproj)

***Перед выполнением необходимо изучить:***

* [*Module 2. Asynchronous programming*](https://videoportal.epam.com/playlist/nR4AYzJD)

**Задание**

Изучите код данного приложения для расчета суммы целых чисел от 0 до N, а затем измените код приложения так, чтобы выполнялись все нижеследующие требования:

1. Расчет должен производиться асинхронно.
2. N задается пользователем из консоли. Пользователь вправе внести новую границу в процессе вычислений, что должно привести к перезапуску расчета.
3. При перезапуске расчета приложение должно продолжить работу без каких-либо сбоев.

Предоставьте ментору ответы на вопросы:

*вопросы заданы на английском, потому что так проще сохранить терминологию, отвечать можно на русском языке*

* Is the method for sum calculation CPU-consuming or IO-bound?
* Does your code use async void? Why?
* Is there any alternative for *Thread.Sleep()* when it comes to artificial delay? What is the benefit of it?
* How do you handle cancellation in your code? Do you use *OperationCancelledException* or *TaskCanceledException*? What is the difference between them?
* Do you throw cancellation exception when the cancellation is triggered right before the work is done? What is the best way to handle such a situation?
* What goes first – input validation or cancellation checks? Why?
* What can you tell about *CancellationToken.CanBeCanceled*? Could you describe a situation where it can be useful?

1. **ASP MVC challenge**

(AsyncAwait.Task2.CodeReviewChallenge.csproj, AsyncAwait.Task2.CodeReviewChallenge.Tests.csproj)

***Перед выполнением необходимо изучить:***

* [*Asynchronous programming. Part III Dos and Don’ts.mp4*](https://videoportal.epam.com/video/GobXNDJy)

**Задание**

Допустим, вы эксперт в асинхронном программировании. Представьте, что вас попросили сделать code review одного ASP.Net Core приложения на предмет корректной реализации асинхронных операций.

1) Проанализируйте код приложения ***AsyncAwait.CodeReviewChallenge***, выпишите замечания (если таковые найдутся), дайте рекомендации по их устранению.

2) Выполните собственные рекомендации относительно приложения, убедитесь, что после этого все тесты успешно завершаются и приложение работает стабильно (возможно, это удобнее делать в отдельной ветке, чтобы было удобно демонстрировать оба задания).

Обсудите рекомендации и полученный результат с ментором. Будьте готовы дать пояснения по поводу того, что при этом происходит «под капотом».

***Оценка по модулю во многом должна зависеть от того, насколько качественно проведена экспертиза, и насколько корректно mentee понимает суть найденных замечаний и предложенных рекомендаций***.

Рекомендуемая литература:

1. [.NET Core Source Browser](https://source.dot.net/) - поможет в исследовании особенностей асинхронного кода в .Net.
2. [AspNet Core Diagnostic scenarios](https://github.com/davidfowl/AspNetCoreDiagnosticScenarios/blob/master/AsyncGuidance.md#async-void)
3. [Don't Block on Async Code](https://blog.stephencleary.com/2012/07/dont-block-on-async-code.html)