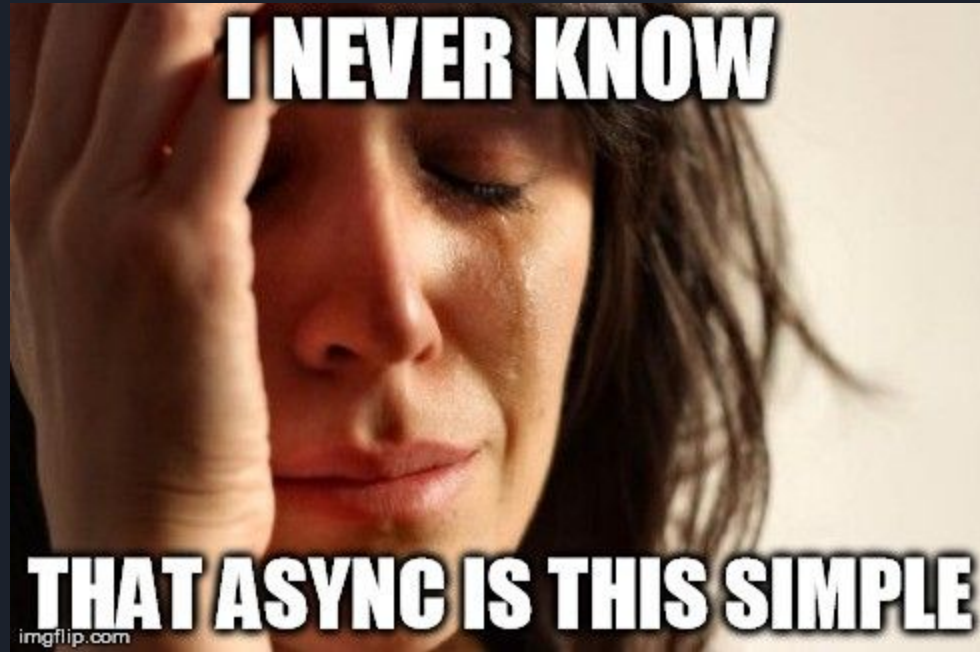


A decorative graphic in the top-left corner consisting of two overlapping parallelograms. The front one is blue and the back one is light green. Both are tilted at an angle.

# Използване на класа Task





# Асинхронно програмиране

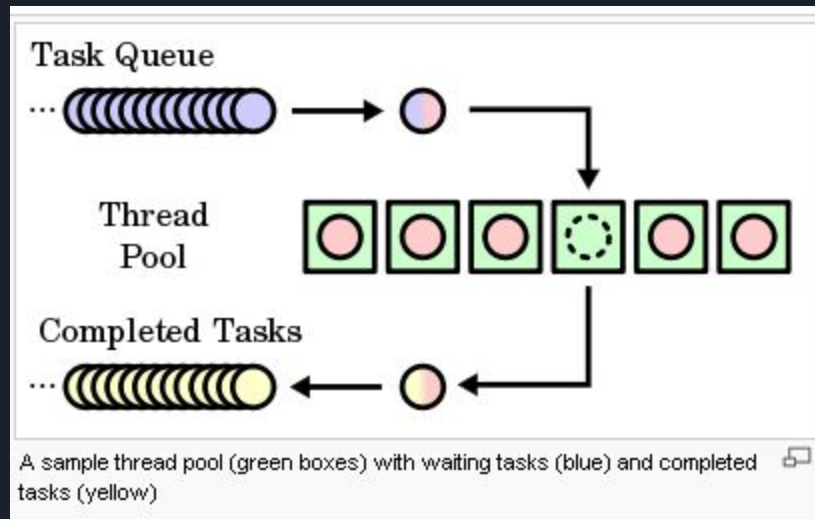
- Вид конкурентно програмиране
- Всяка група от команди се изпълняват на отделна нишка
- Създава се обект, който информира главната нишка при завършване или провал.
- Този обект се използва и за следене на прогреса.



# Task

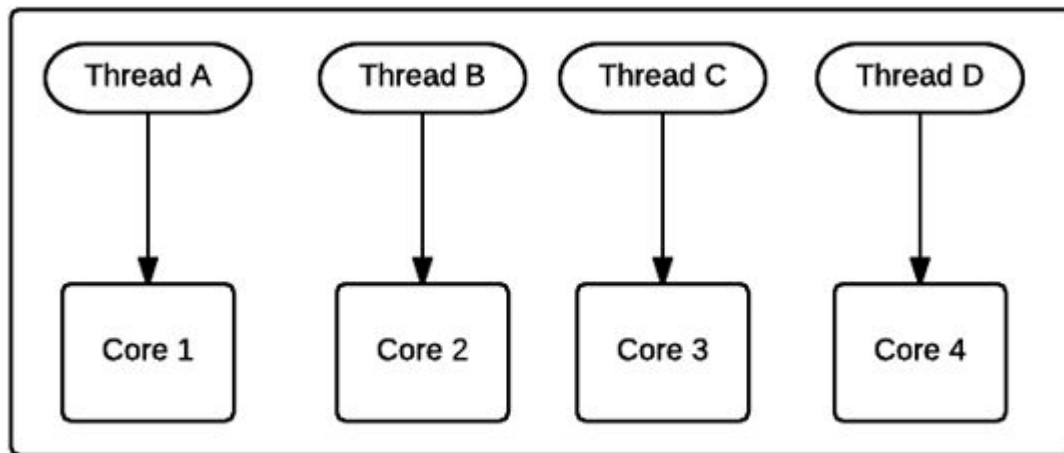
- Task = Обект, който позволява асинхронното изпълнение на команди.
- Task = обект, представящ команди, които трябва да се изпълнят
- Task позволява да разберем дали работата е приключена и спомага за получаване на резултат от работата

# Task



# Нишки - да припомним

- Нишка - поредица от команди, които могат да бъдат изпълнени





# Кога да използваме Task

- Когато желаем да изпълняваме няколко действия паралелно
- Когато действията са свързани с нещо външно
  - Например: Достъпване на страница от уеб сървър
- Ако трябва да получим резултат от изпълнението
- Ако искаме да имаме възможност да отменим изпълнението
- Предпочитан избор в C#
- В други програмни езици има аналогични обекти - обещание (Promise)



# Разлики м/у Thread и Task

- Thread използва абстракциите на ОС за работа с нишки
- Task е начинът за изпълнение на асинхронен код и е част от Task Parallel Library библиотеката на C#
- Task може да върне резултат. Няма директен начин да изкараме резултат от нишка.
- Task поддържа отмяна (cancellation) чрез cancellation tokens. Нишките нямат такава възможност
- Новата нишка е самостоятелен обект, а Task се базира на ThreadPool.





## Индиректно създаване на задача

- `Parallel.Invoke(() => метод1());`
- Където `метод1()` е метод реализиращ дадена операция.
  
- Може да подадете и повече извиквания накуп:  
`Parallel.Invoke(() => DoSomeWork(), () =>  
DoSomeOtherWork());`



# Създаване на задача чрез обект

- Обект от класа Task може да се създаде и изрично:  
`Task taskA = new Task( () => Console.WriteLine("Hello from taskA.")).`
- Изпълнението започва при извикване на метод Start:

```
taskA.Start();
```

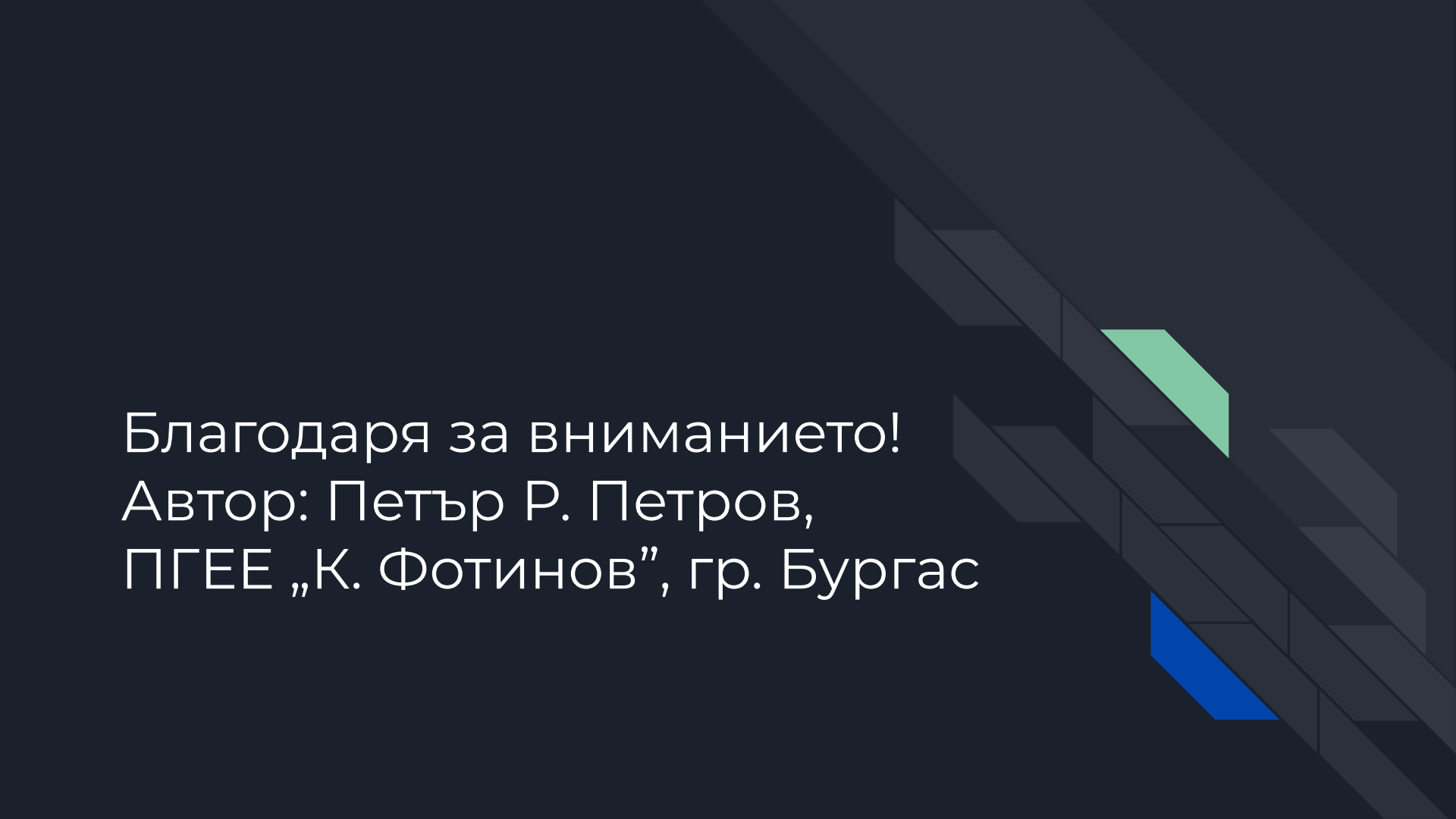
- Може да изчакате да приключи изпълнението на задачата чрез Wait:

```
taskA.Wait();
```



## Създаване и директно изпълнение на задача:

- Чрез метод Run:
- `Task taskA = Task.Run( () => Console.WriteLine("Hello from taskA."));`



Благодаря за вниманието!  
Автор: Петър Р. Петров,  
ПГЕЕ „К. Фотинов“, гр. Бургас