ASYNC, AWAIT, OH… WAIT!

YOU'D BETTER WATCH YOUR STEP

December 2016 – Alt.Net Paris

@tpierrain (use case driven)

“USE CASE DRIVEN”, BUT ALSO…

42skillz

@tpierrain

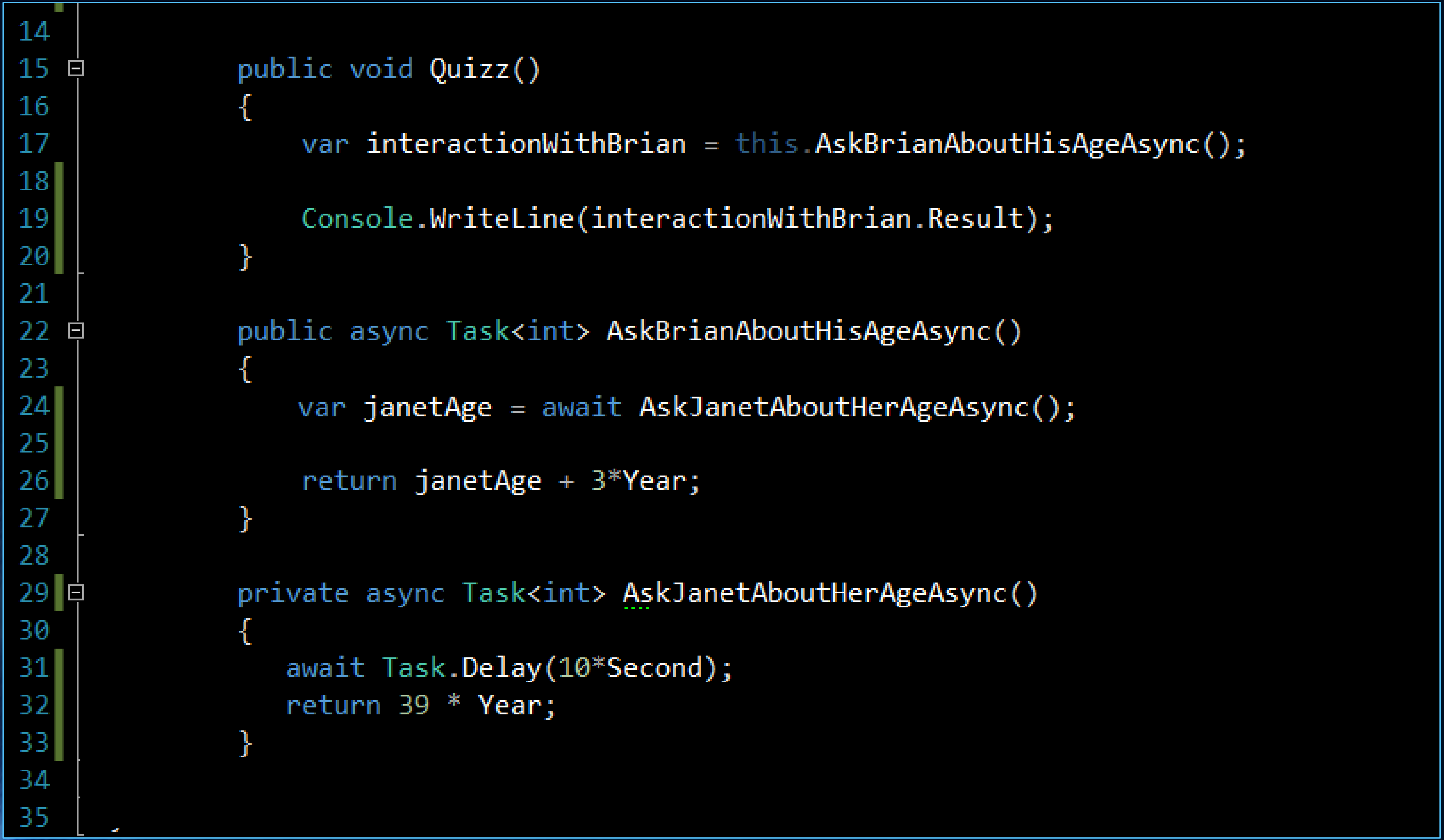
- Reactive Programmer (> 10 years)
- eXtreme Programmer (> 10 years)
- Programmer (> 18 years)

NFluent

Value (DDD)

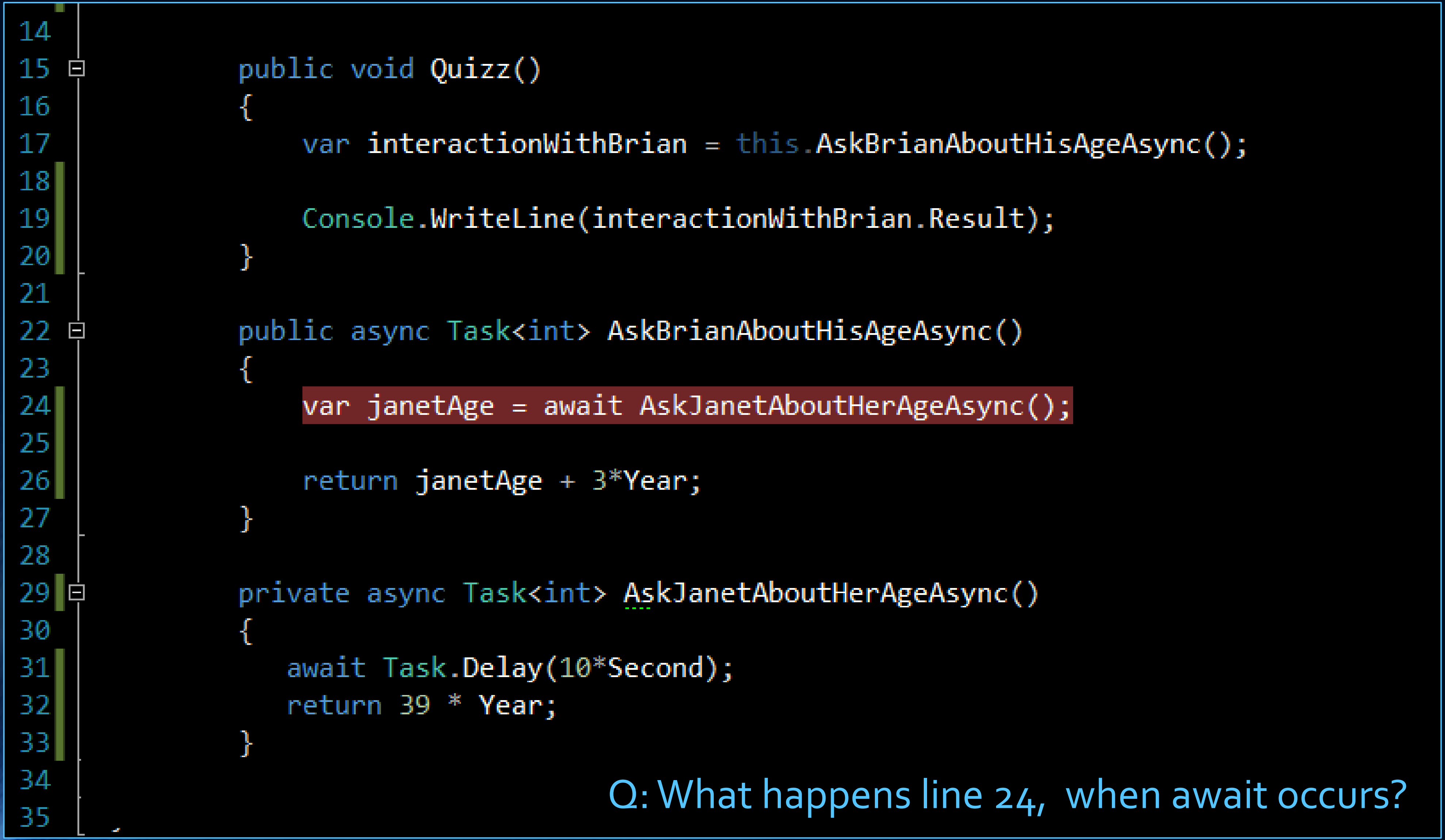
KYOCERA
advanced ceramics

LET’S START WITH A QUESTION…



```
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…

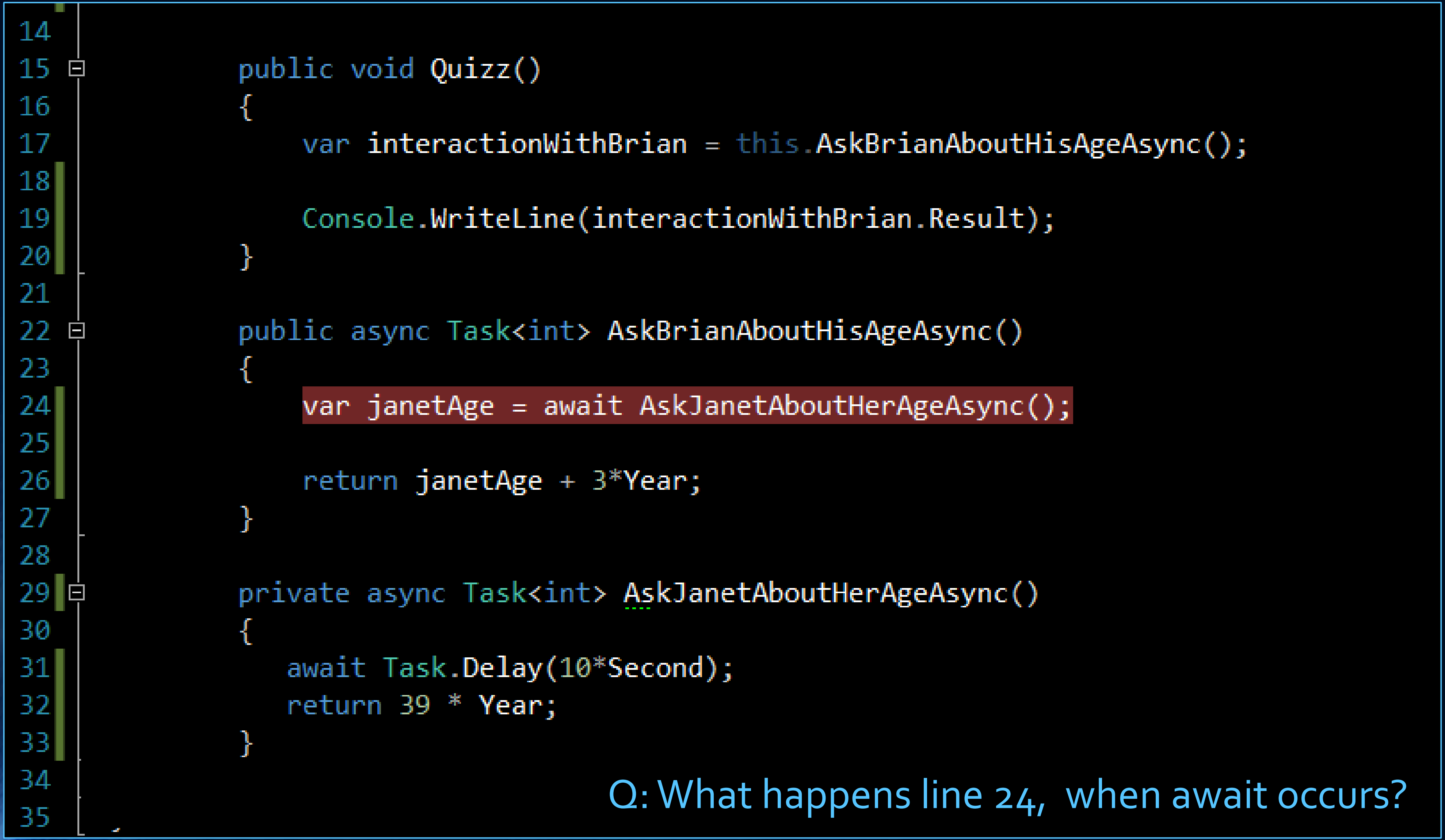


```
14
15          public void Quizz()
16          {
17              var interactionWithBrian = this.AskBrianAboutHisAgeAsync();
18
19              Console.WriteLine(interactionWithBrian.Result);
20          }
21
22          public async Task<int> AskBrianAboutHisAgeAsync()
23          {
24              var janetAge = await AskJanetAboutHerAgeAsync();
25
26              return janetAge + 3*Year;
27          }
28
29          private async Task<int> AskJanetAboutHerAgeAsync()
30          {
31             await Task.Delay(10*Second);
32             return 39 * Year;
33          }
34
35
```

Q: What happens line 24, when await occurs?

OUPS! LET’S ADD SOME CONTEXT

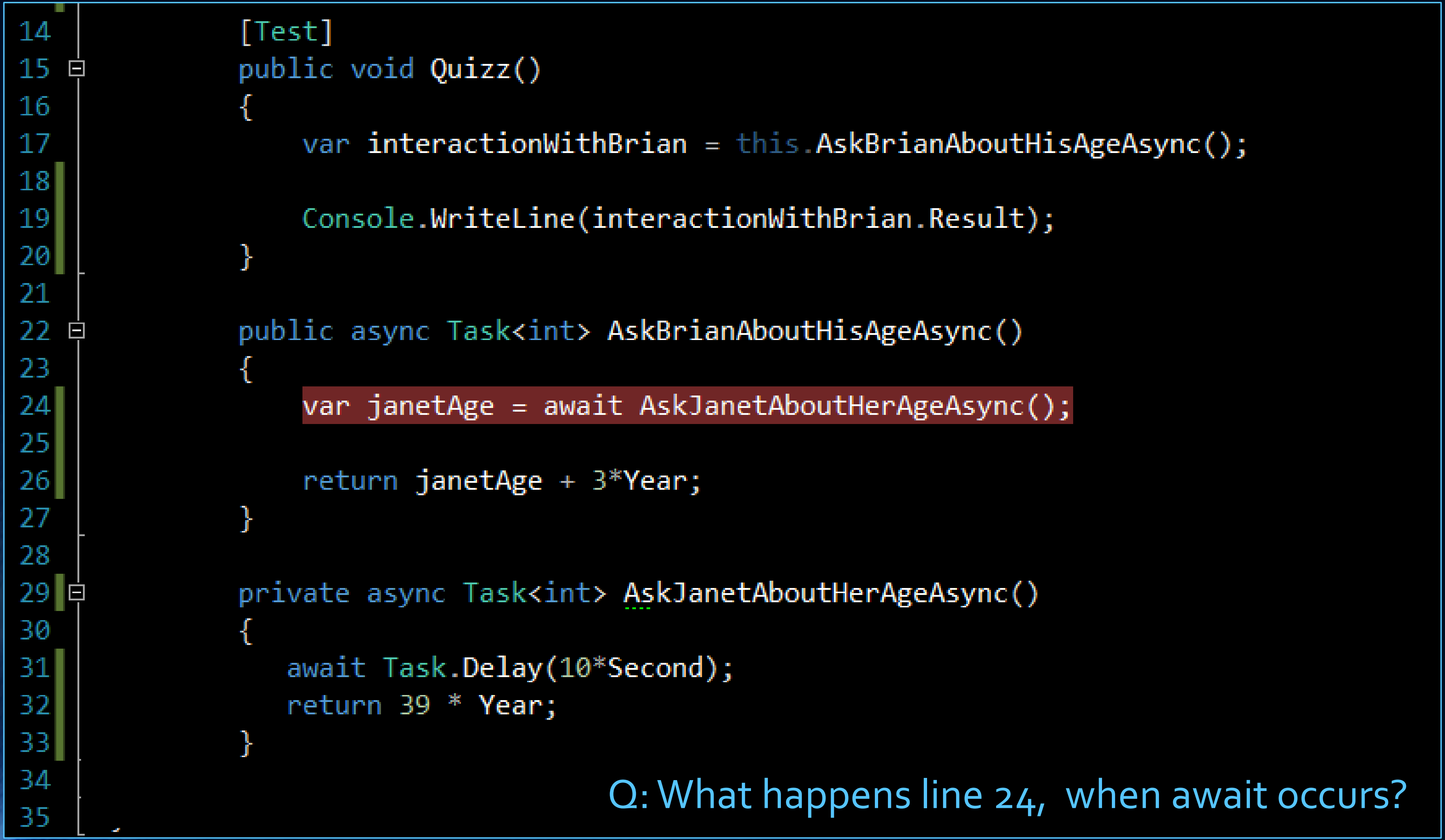
LET'S START WITH A QUESTION…



```
14
15          public void Quizz()
16          {
17              var interactionWithBrian = this.AskBrianAboutHisAgeAsync();
18
19              Console.WriteLine(interactionWithBrian.Result);
20          }
21
22          public async Task<int> AskBrianAboutHisAgeAsync()
23          {
24              var janetAge = await AskJanetAboutHerAgeAsync();
25
26              return janetAge + 3*Year;
27          }
28
29          private async Task<int> AskJanetAboutHerAgeAsync()
30          {
31             await Task.Delay(10*Second);
32             return 39 * Year;
33          }
34
35
```

Q: What happens line 24, when await occurs?

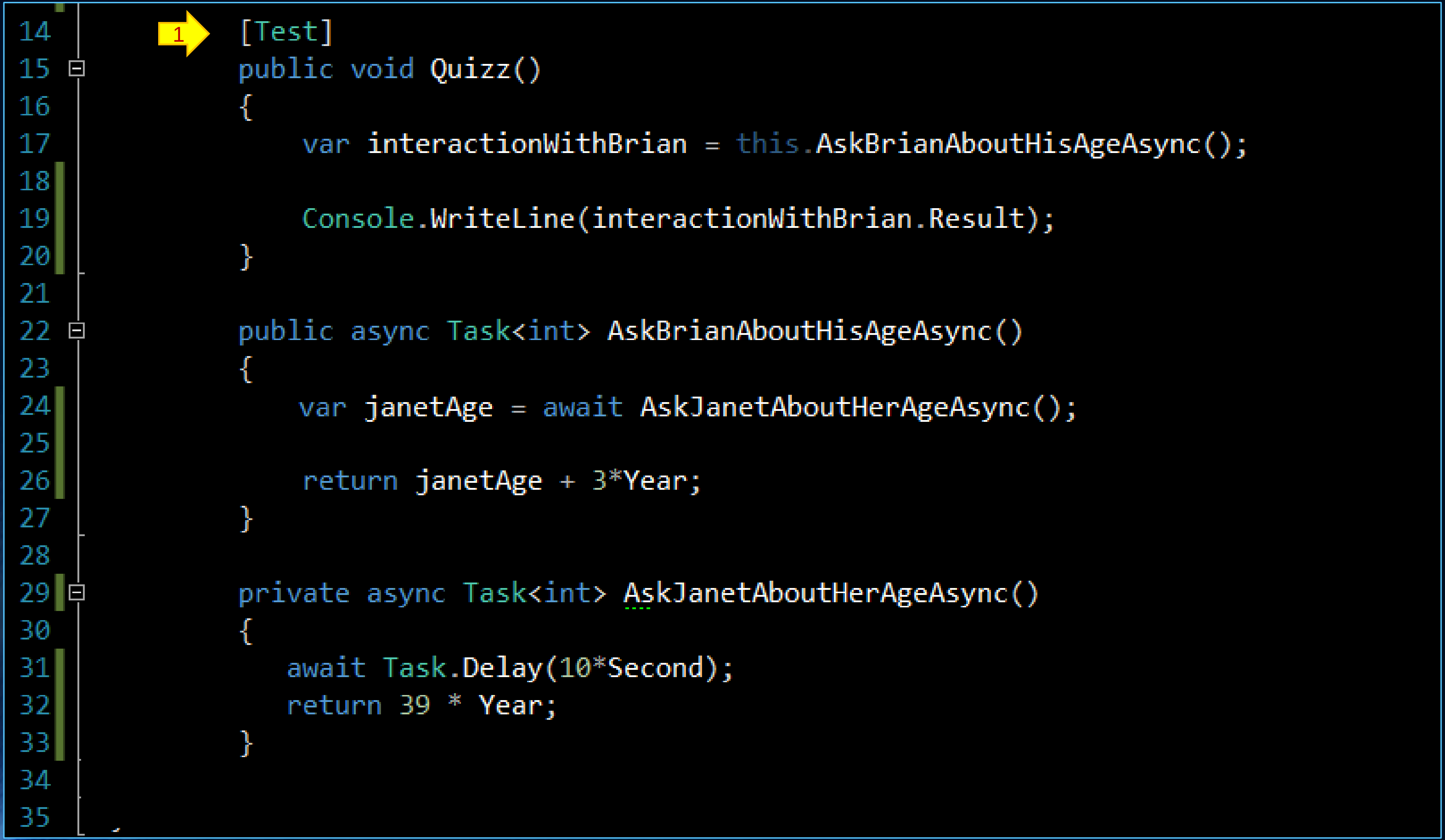
LET’S START WITH A QUESTION…



```
14          [Test]
15          public void Quizz()
16          {
17              var interactionWithBrian = this.AskBrianAboutHisAgeAsync();
18
19              Console.WriteLine(interactionWithBrian.Result);
20          }
21
22          public async Task<int> AskBrianAboutHisAgeAsync()
23          {
24              var janetAge = await AskJanetAboutHerAgeAsync();
25
26              return janetAge + 3*Year;
27          }
28
29          private async Task<int> AskJanetAboutHerAgeAsync()
30          {
31             await Task.Delay(10*Second);
32             return 39 * Year;
33          }
34
35
```

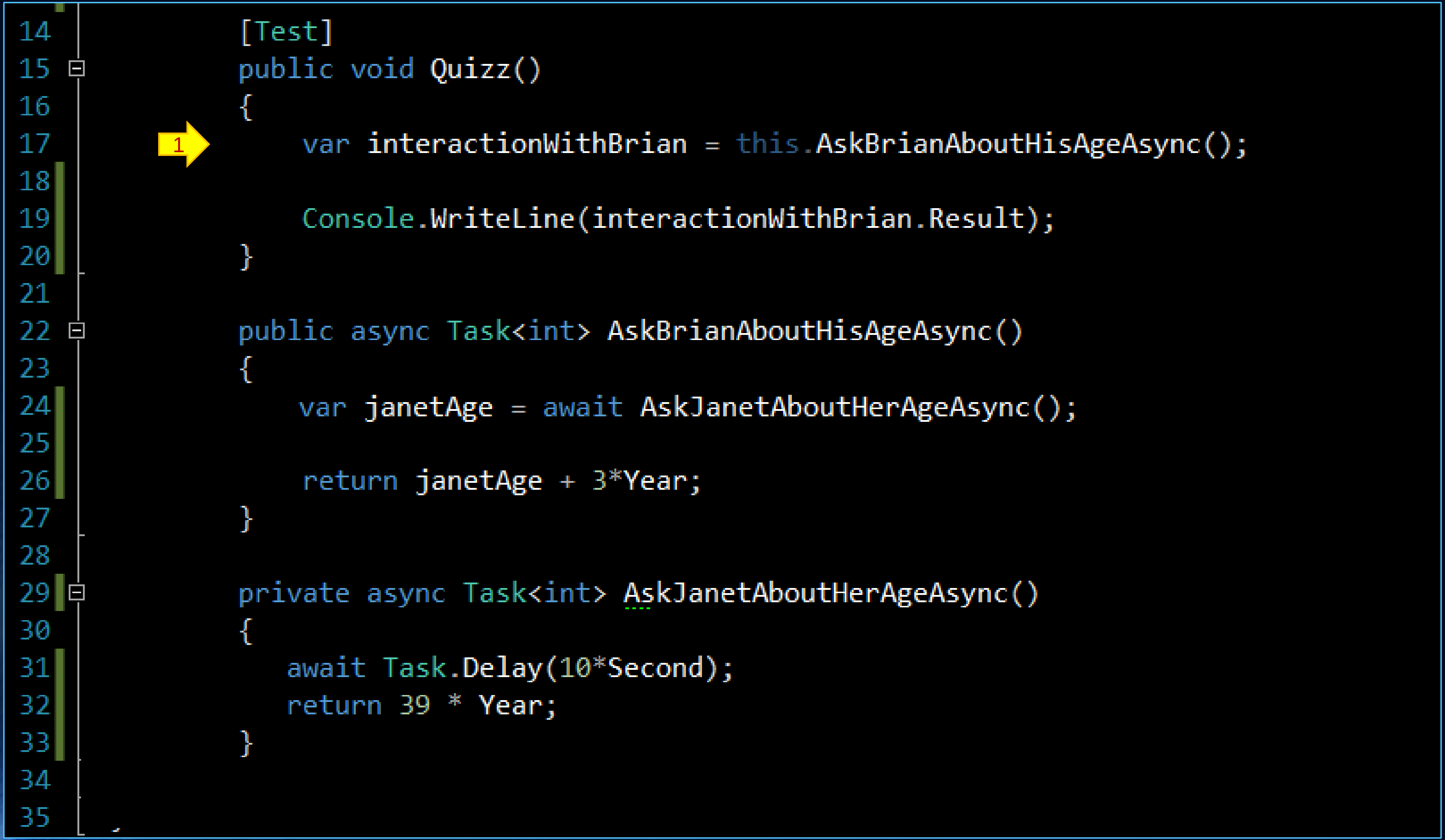
Q: What happens line 24, when await occurs?

LET'S START WITH A QUESTION…



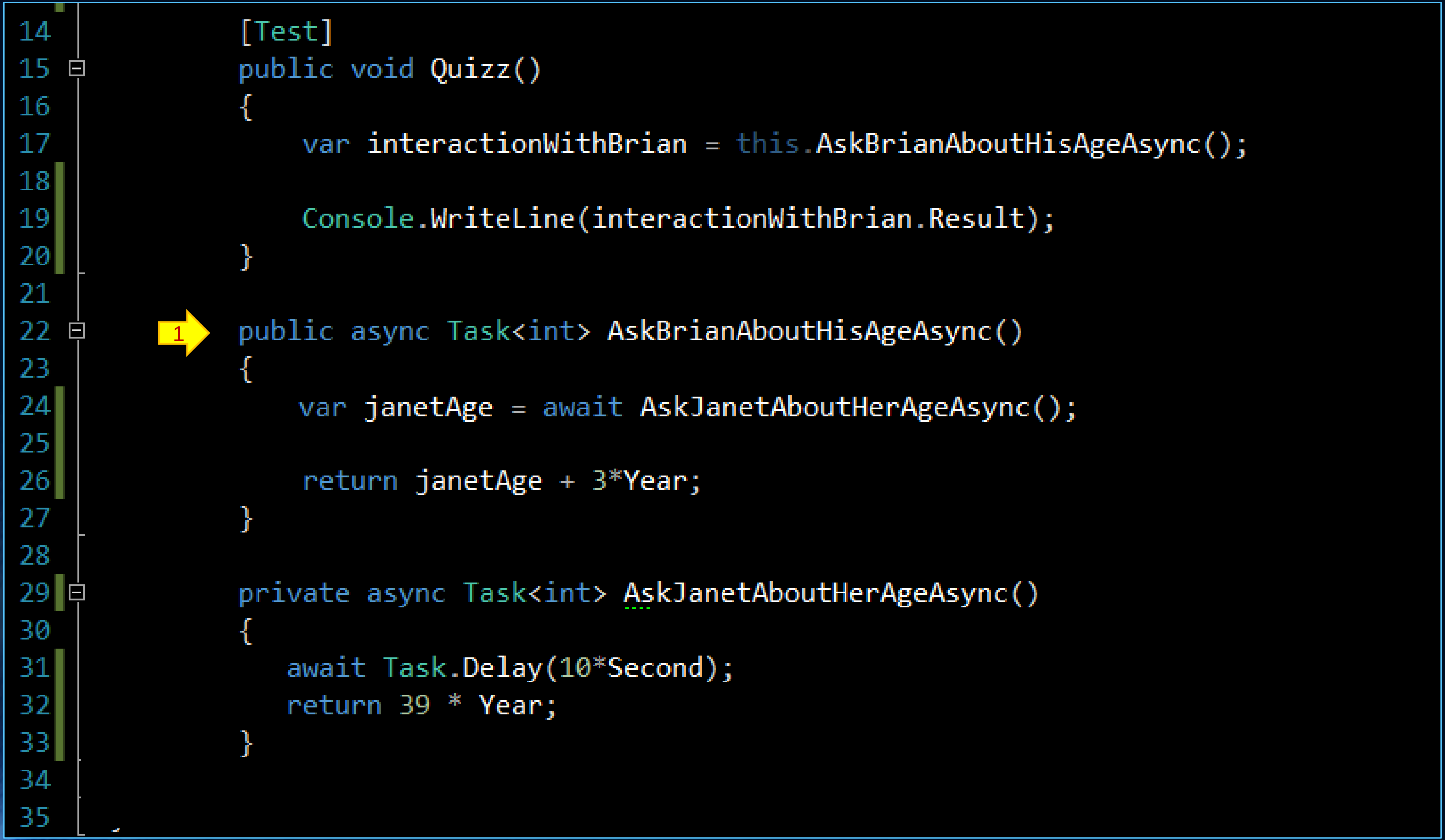
```
[Test]
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

LET'S START WITH A QUESTION…



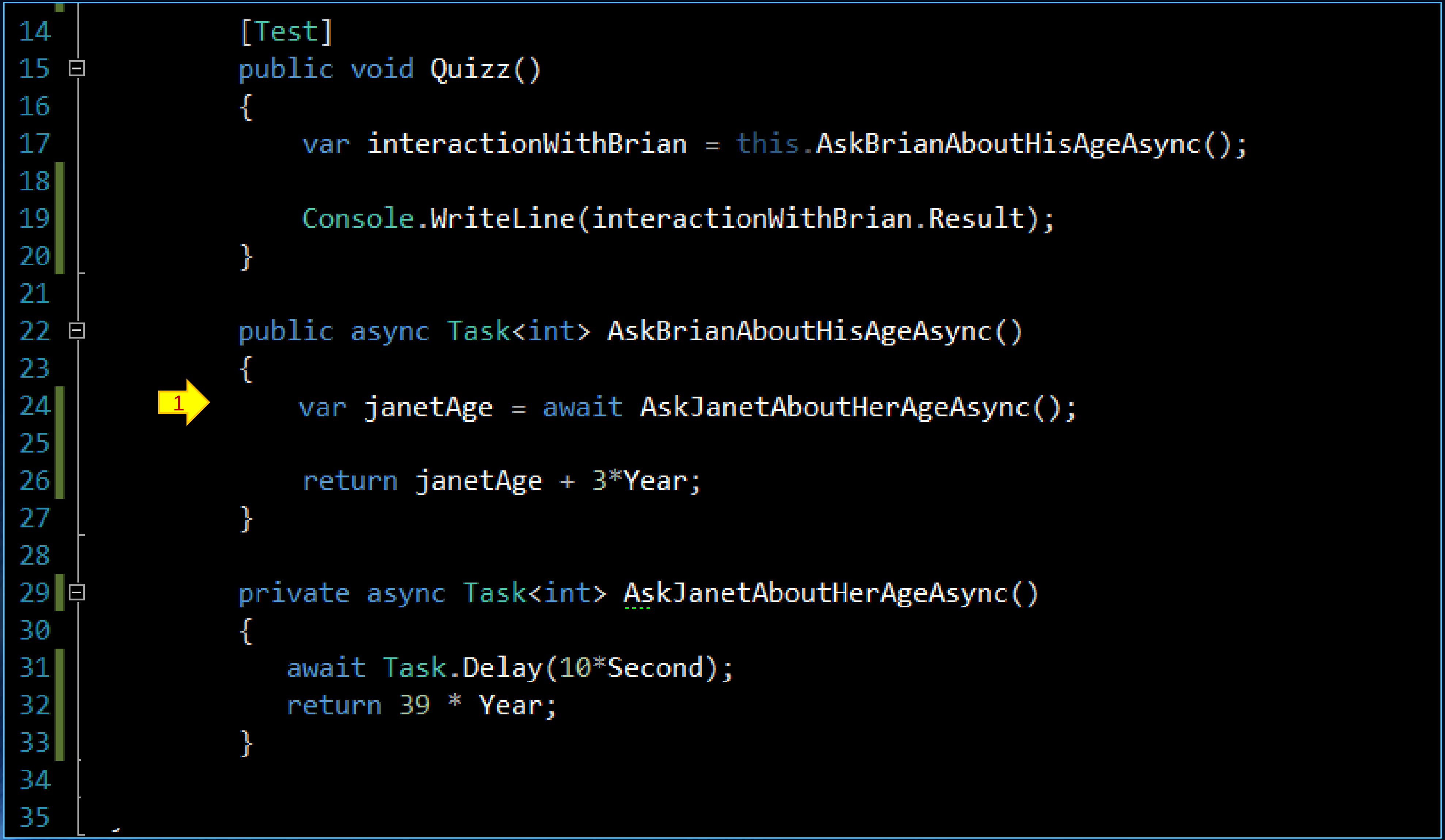
```
        [Test]
        public void Quizz()
        {
1           var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



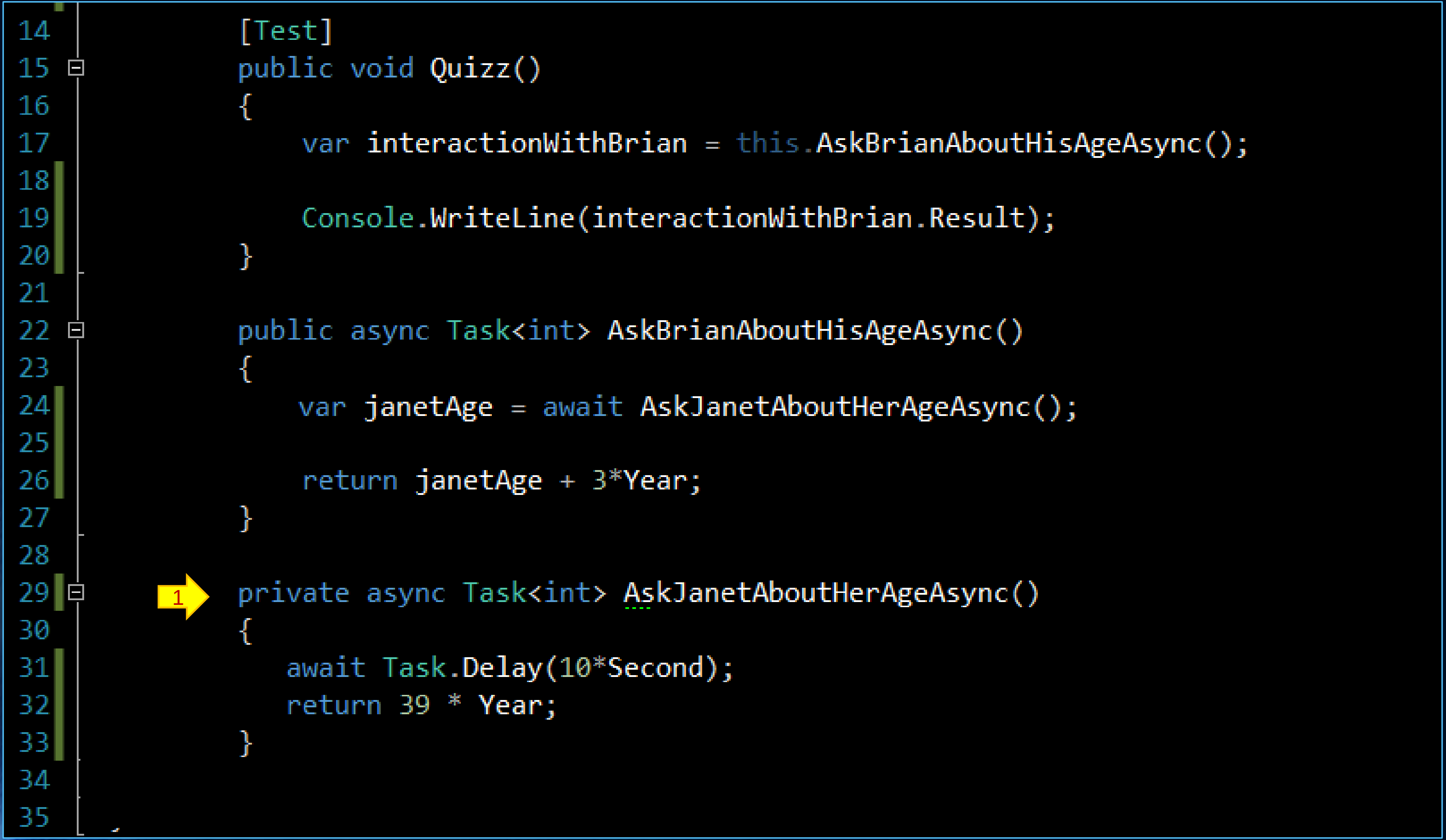
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



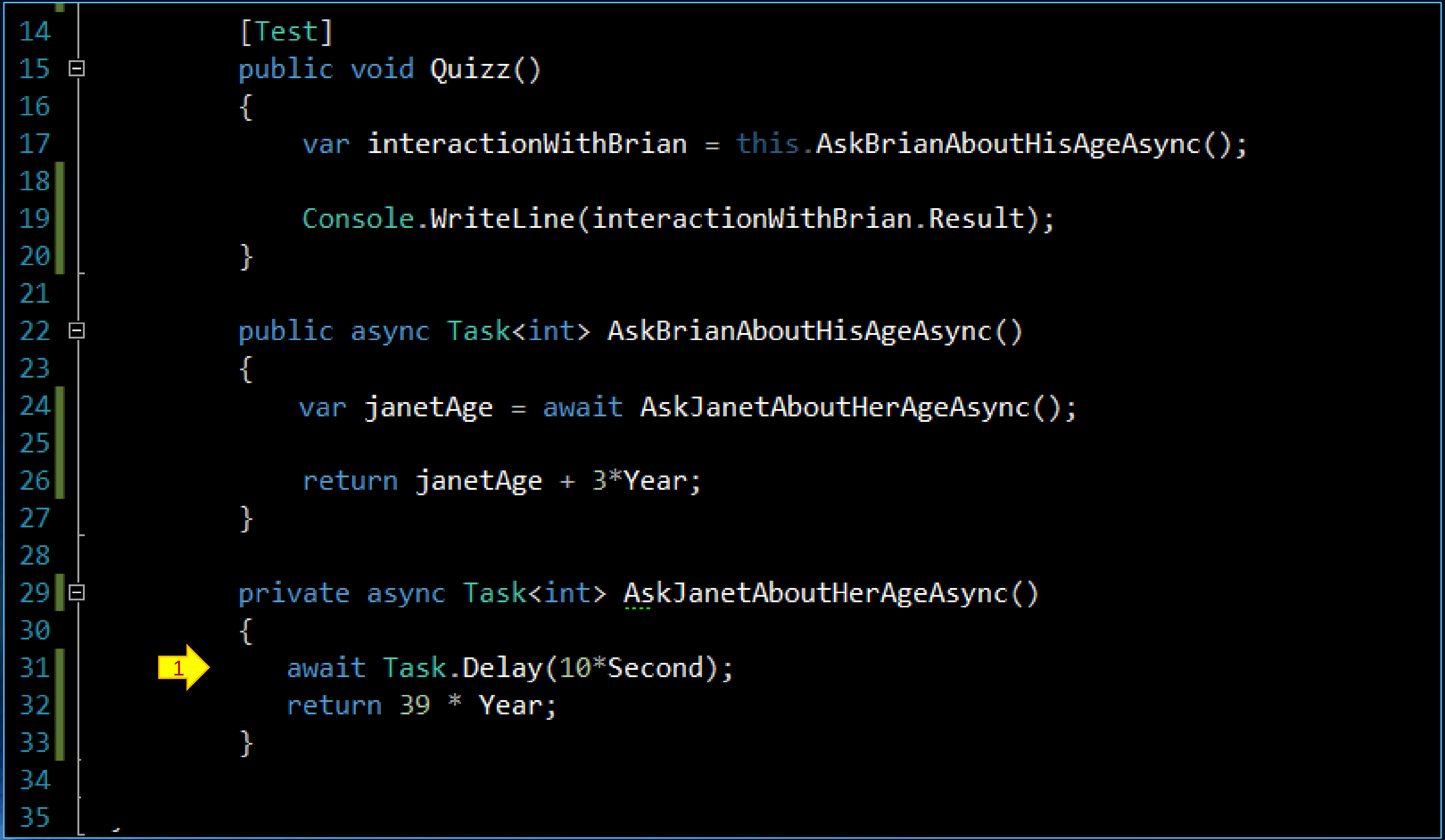
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



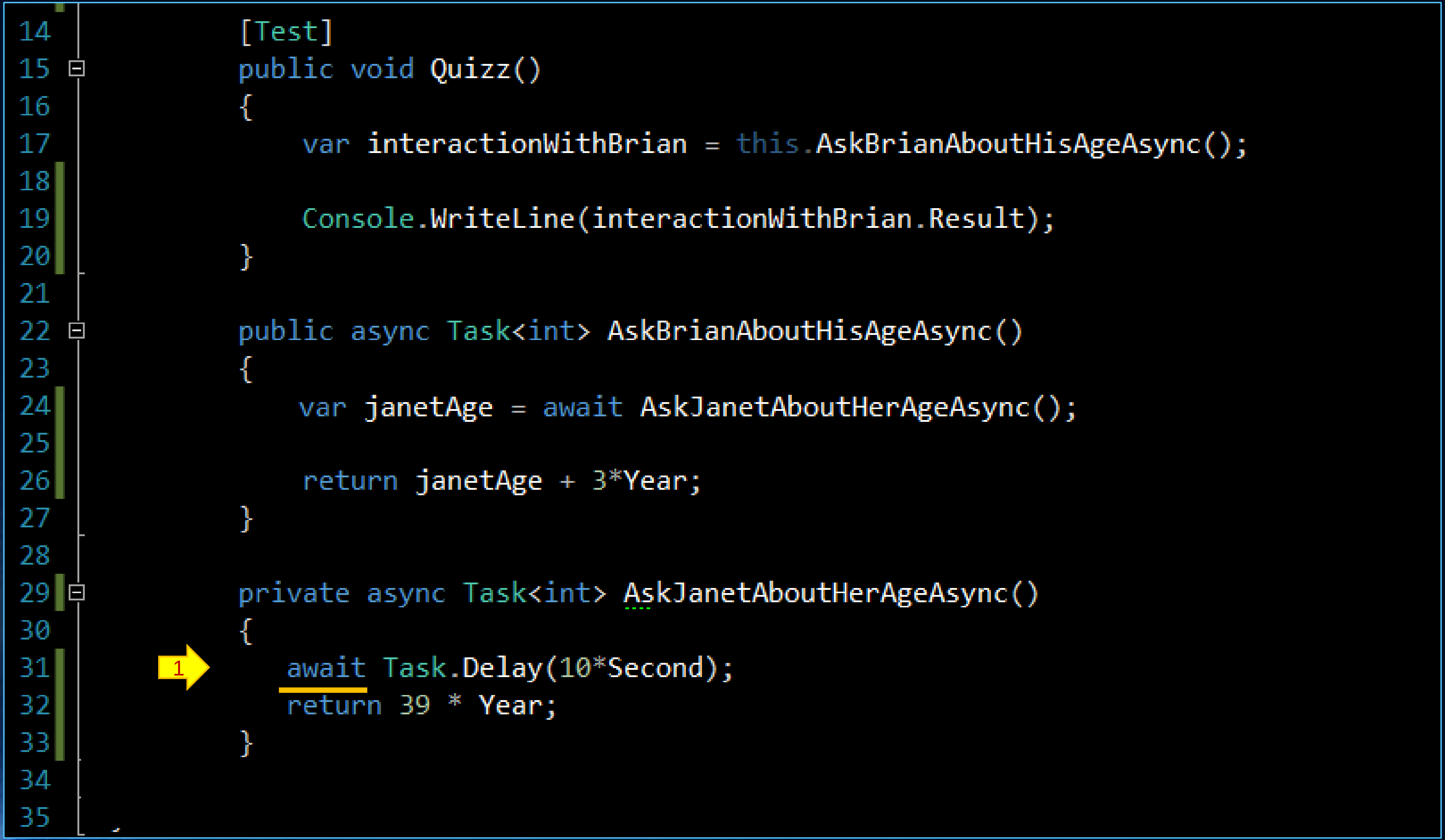
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



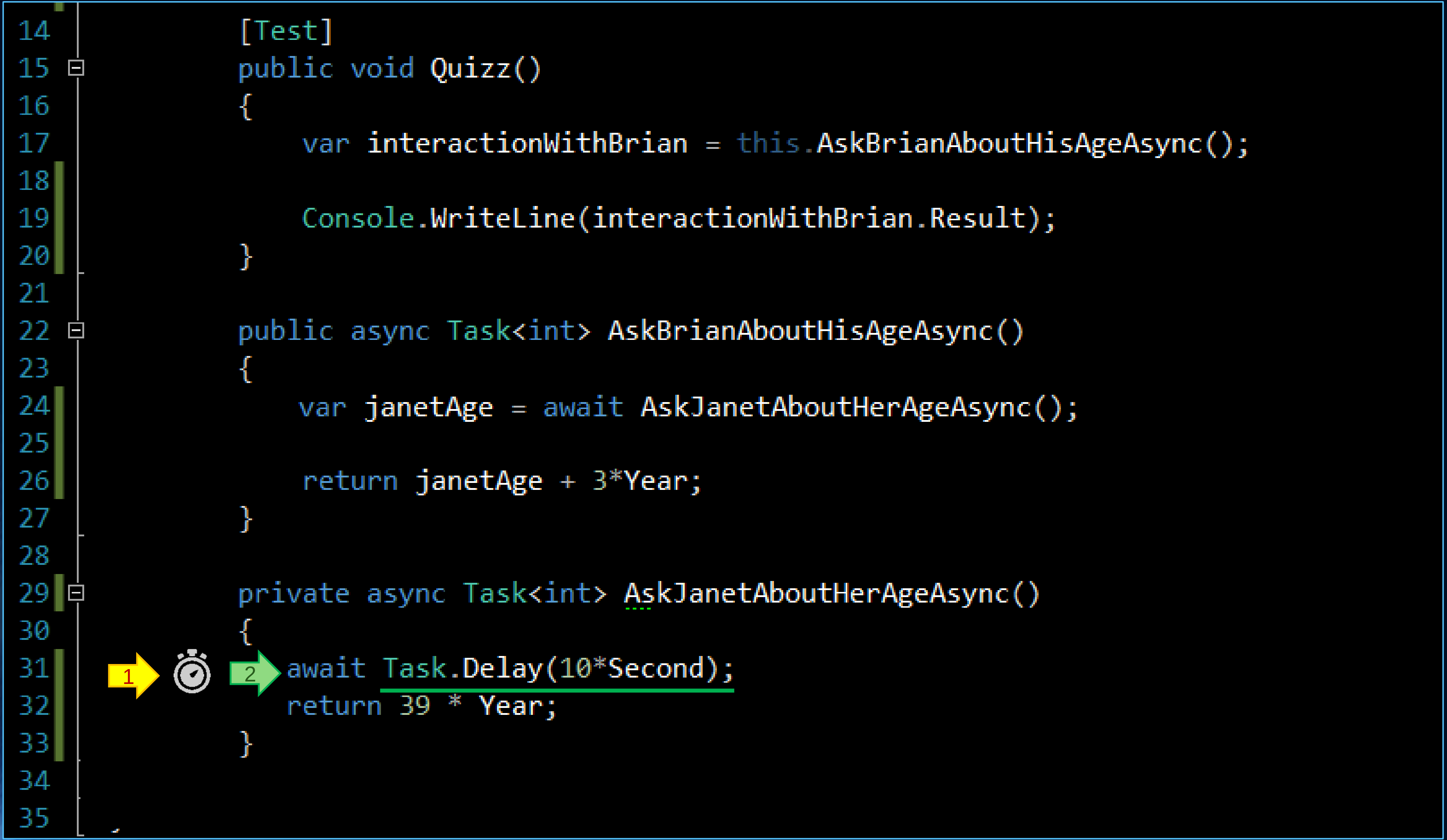
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
1          await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



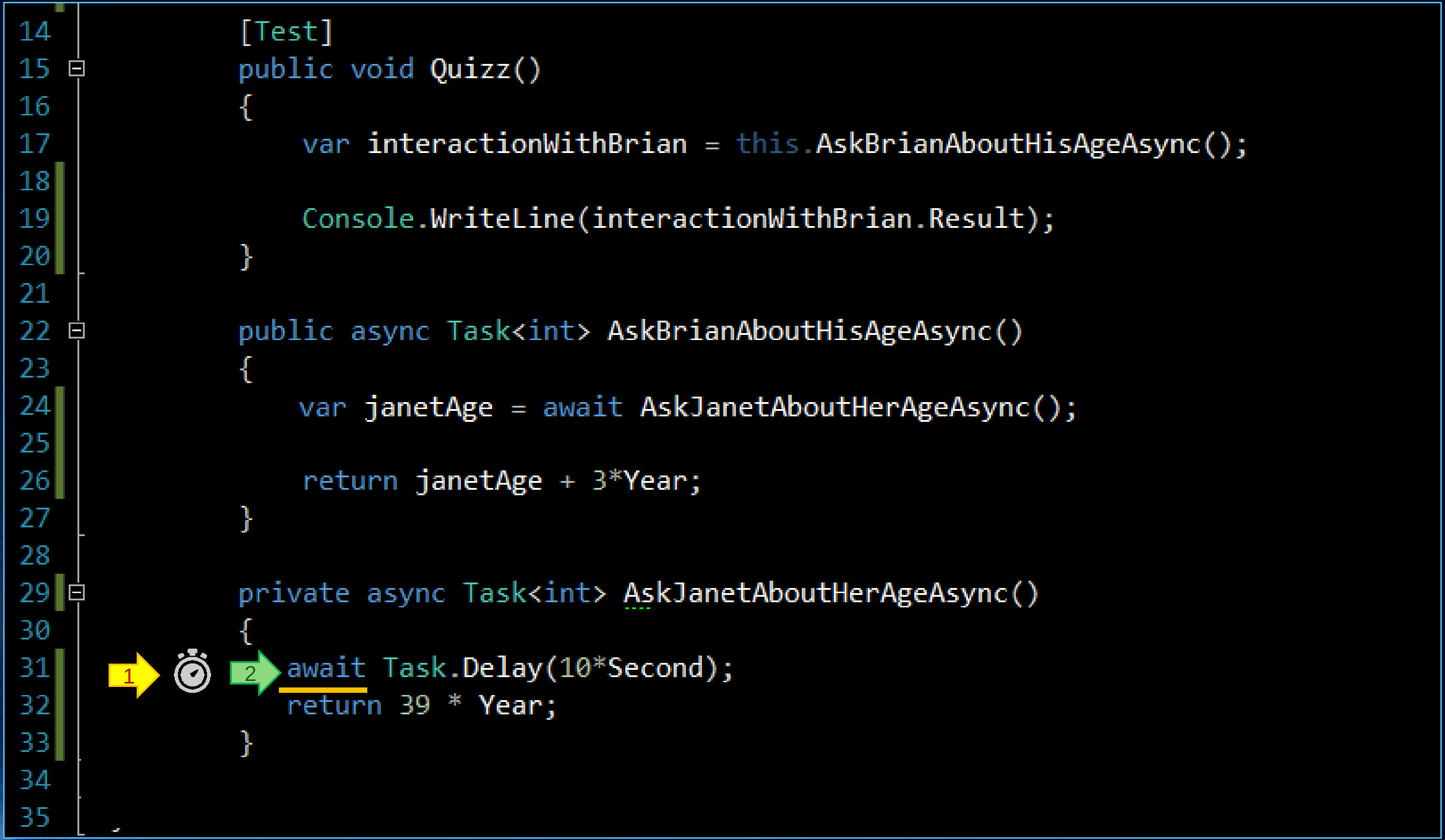
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
     1 ➔   await Task.Delay(10*Second);
            return 39 * Year;
        }
```

LET’S START WITH A QUESTION…



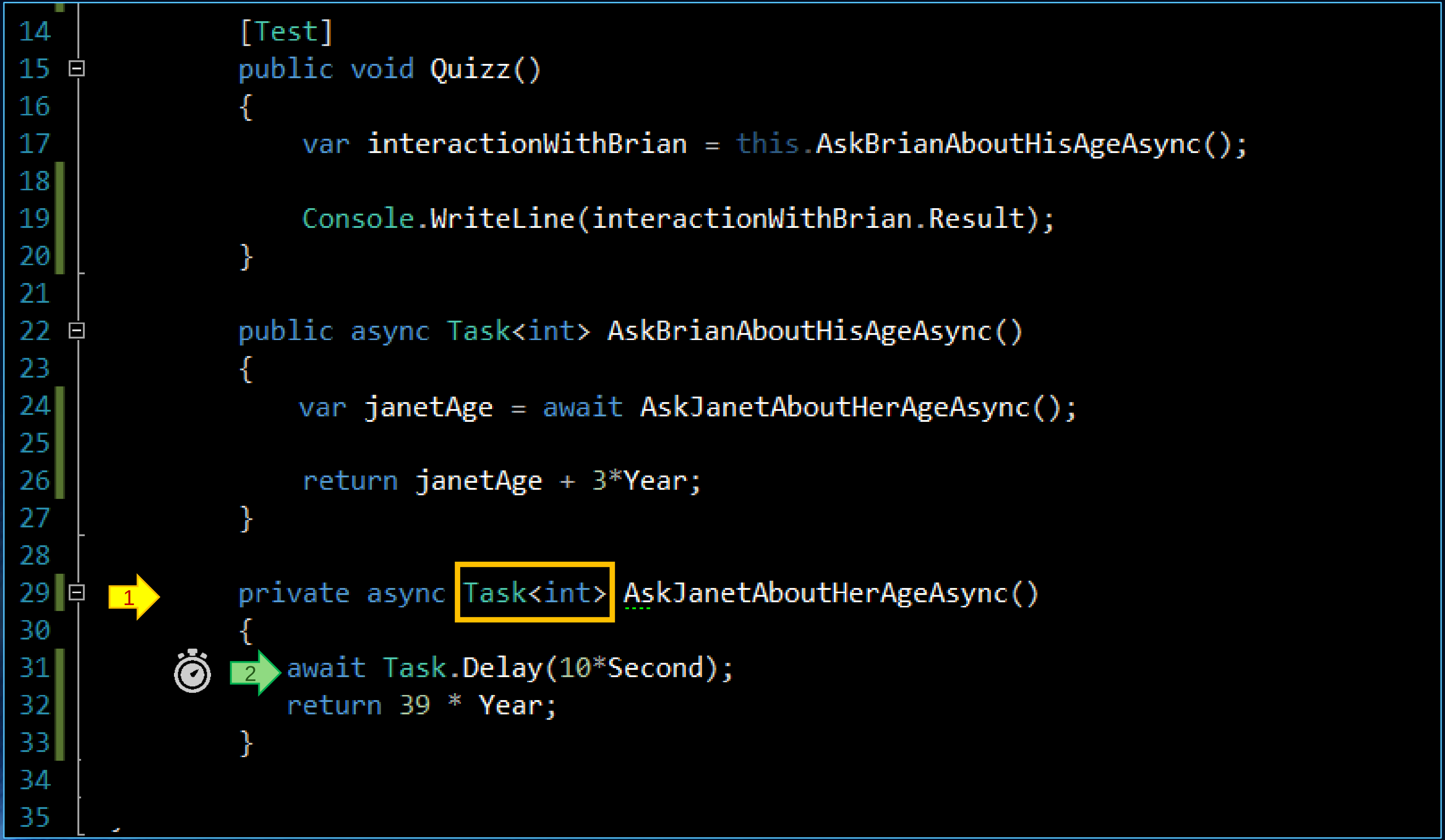
```
[Test]
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

LET'S START WITH A QUESTION…



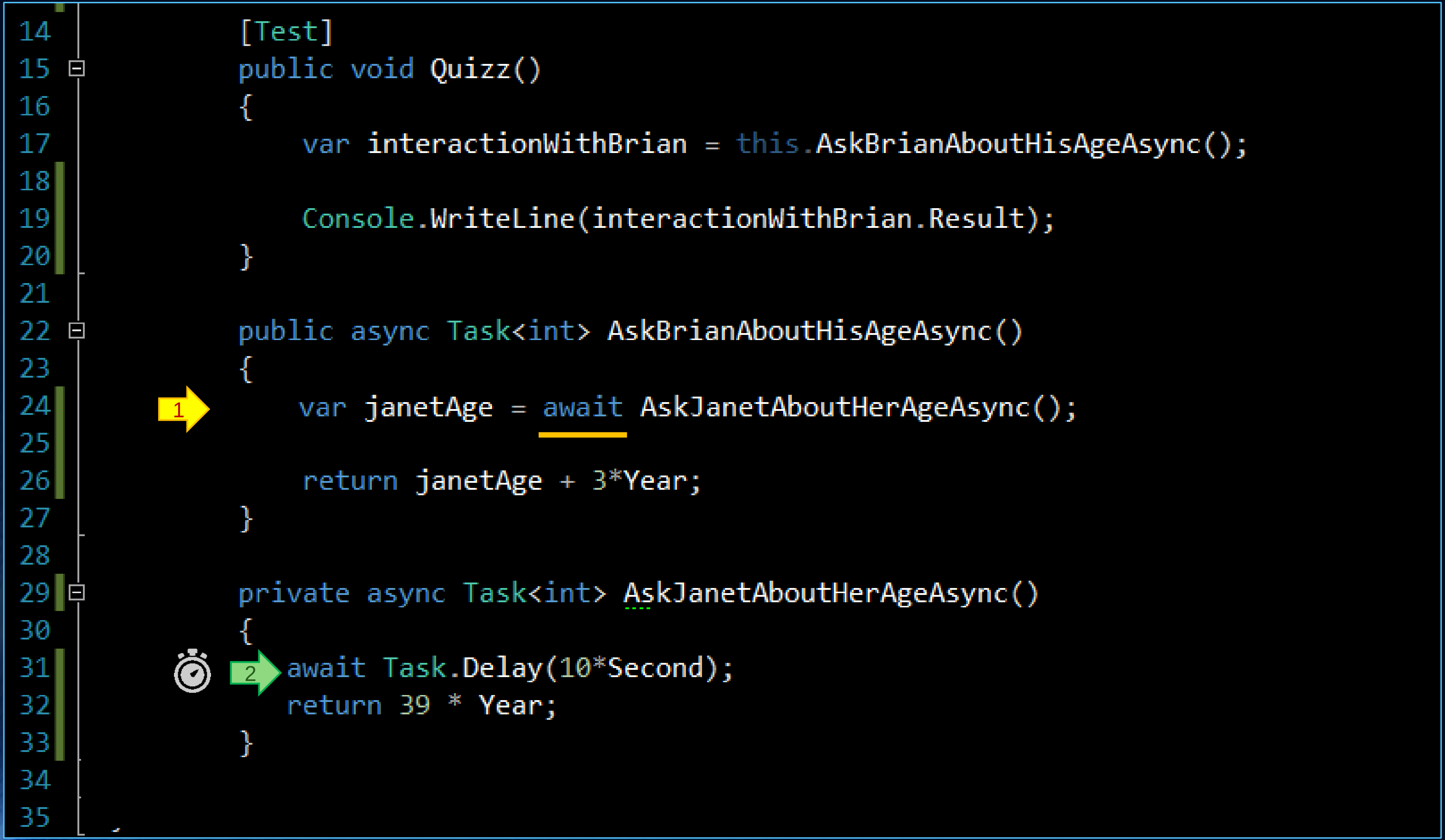
```
[Test]
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

LET'S START WITH A QUESTION...



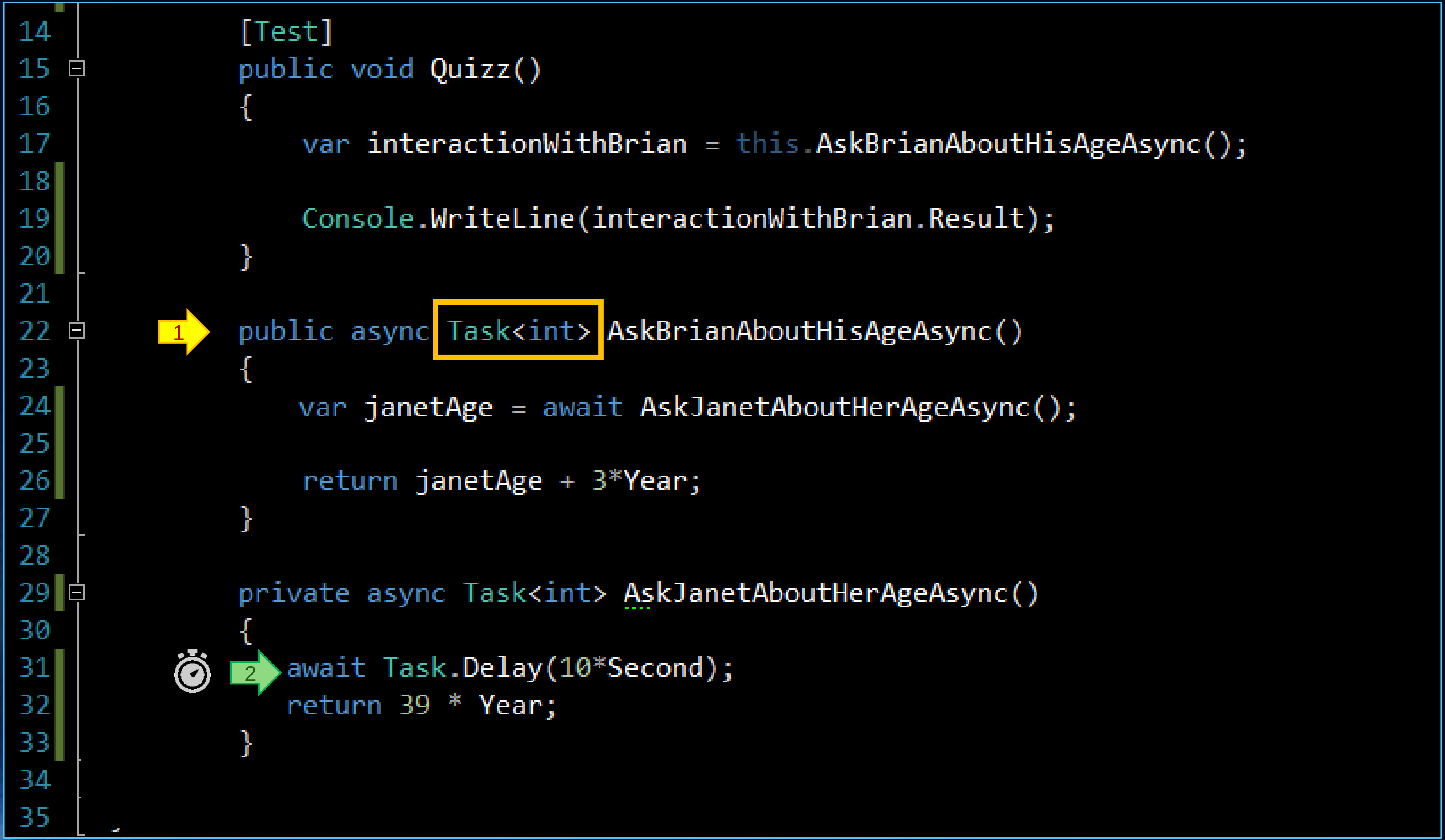
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



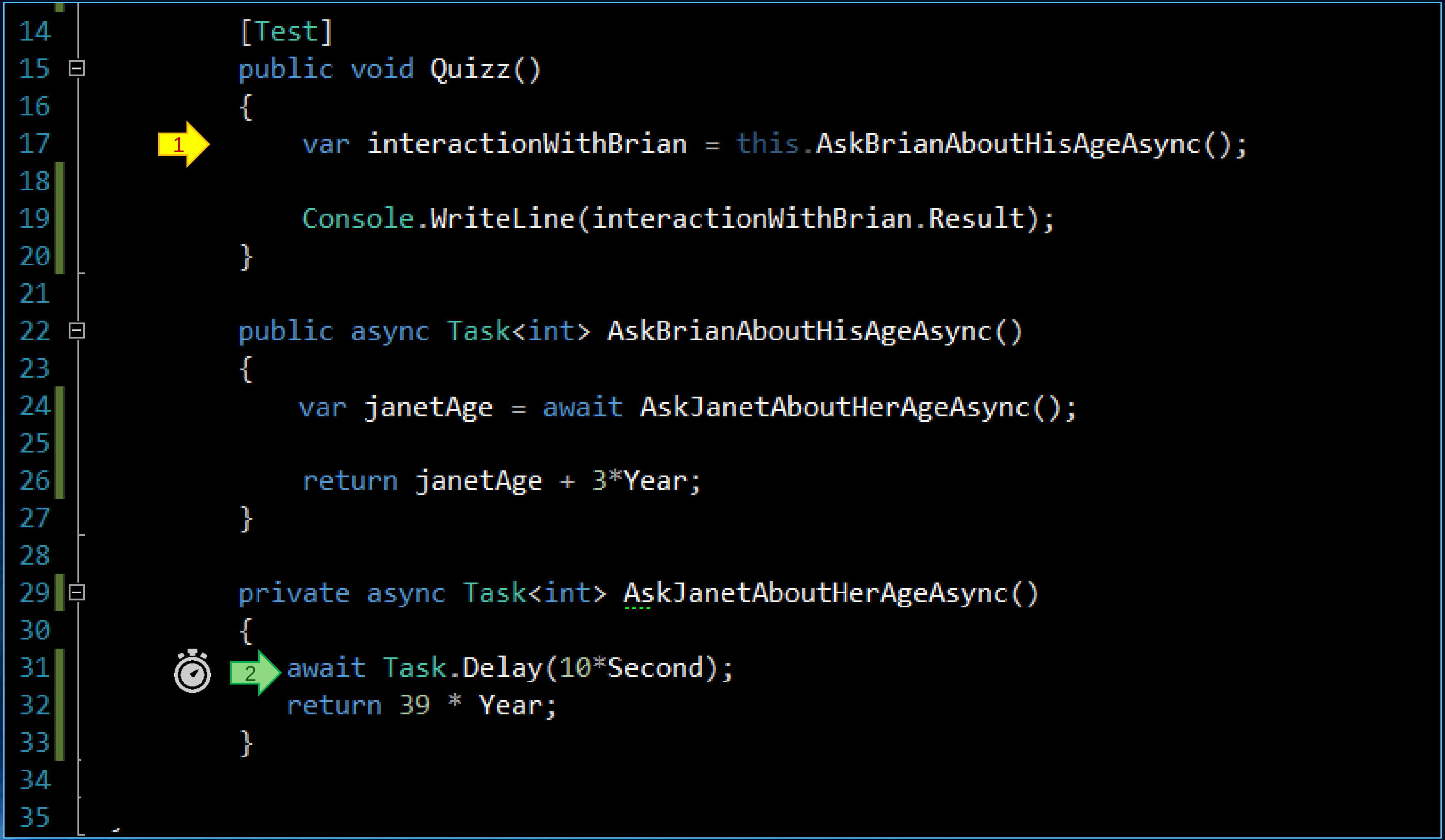
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
     1      var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
     2    await Task.Delay(10*Second);
          return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



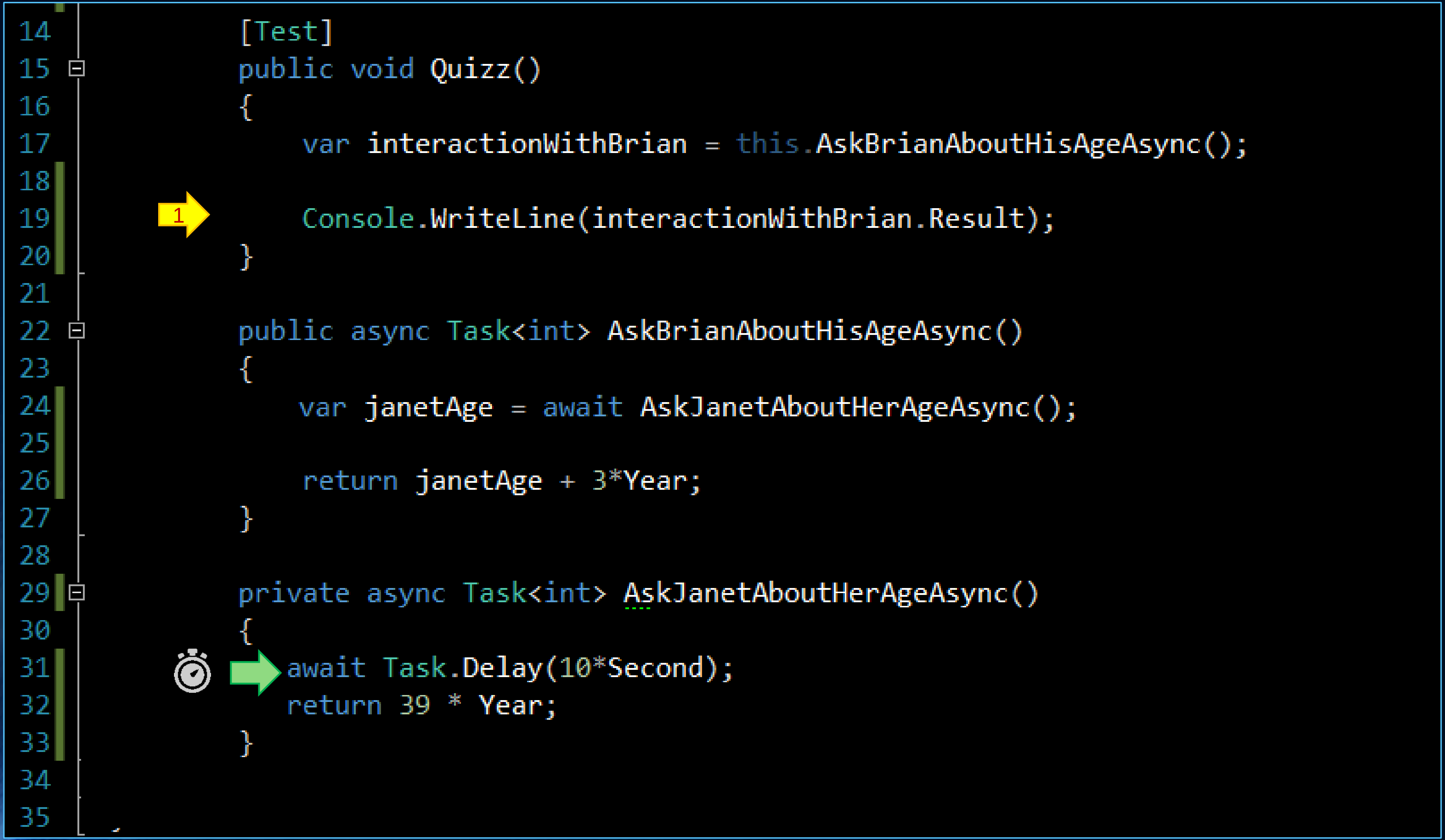
```
[Test]
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

LET'S START WITH A QUESTION…



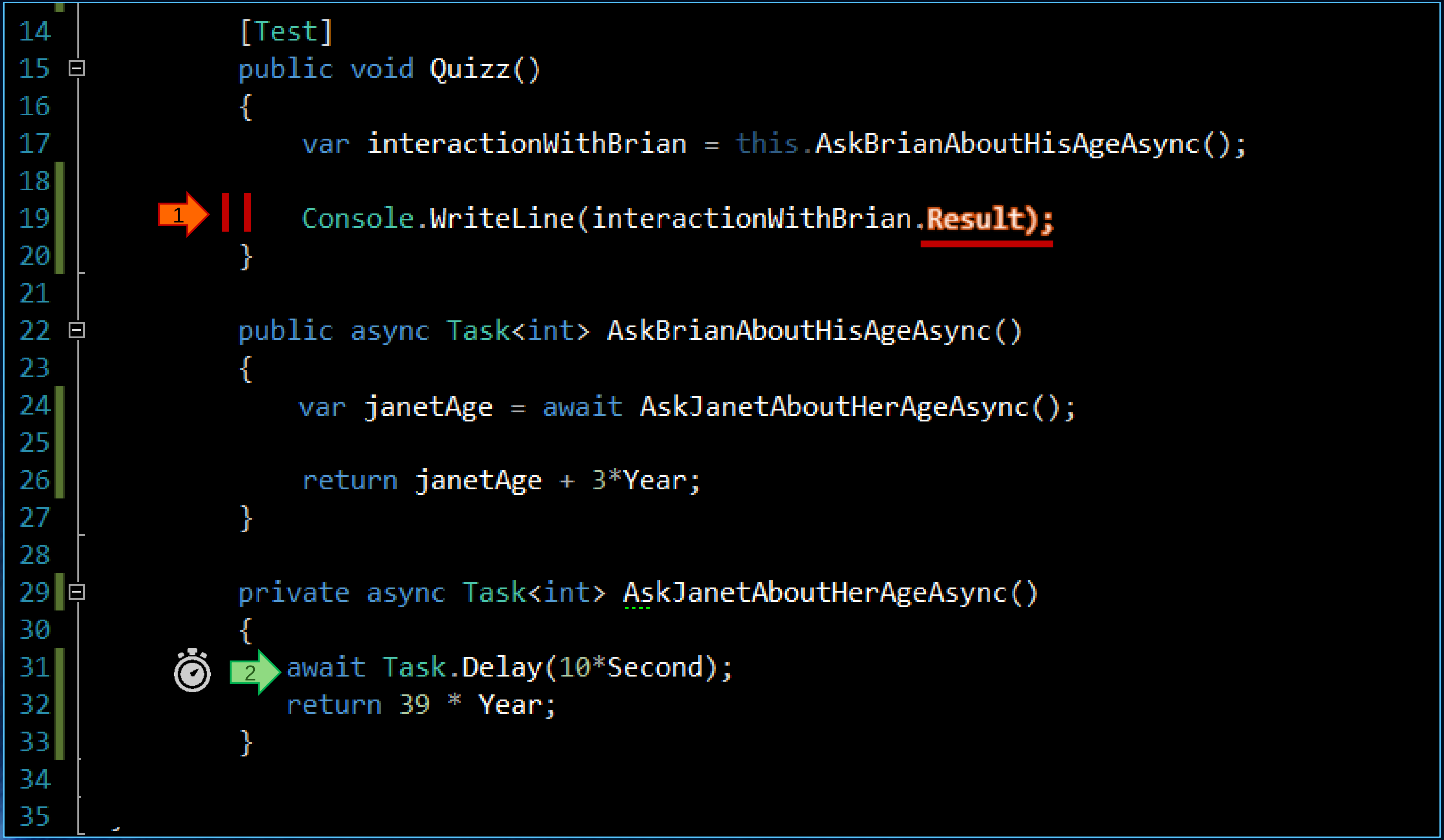
```
[Test]
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

LET'S START WITH A QUESTION...



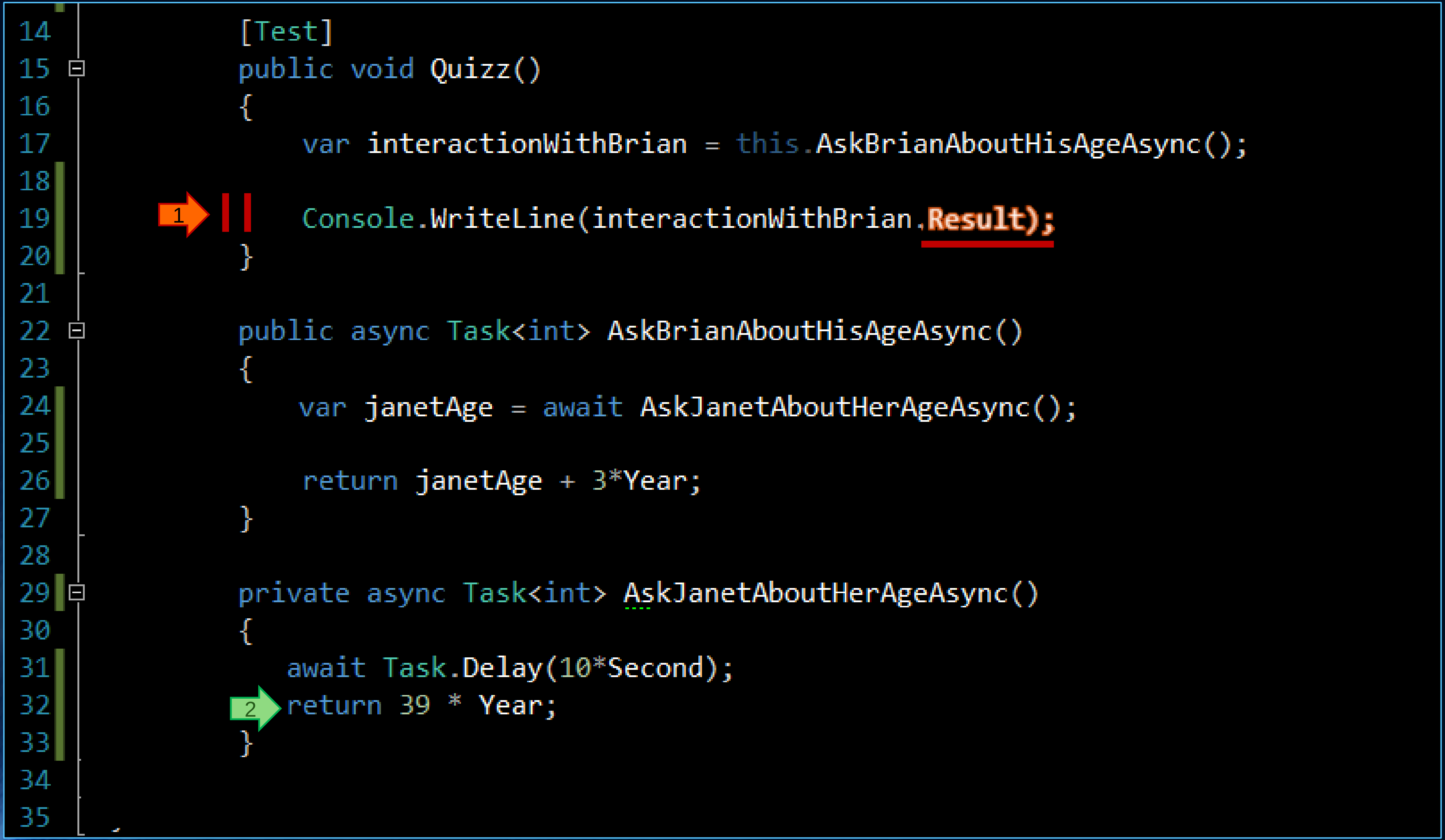
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



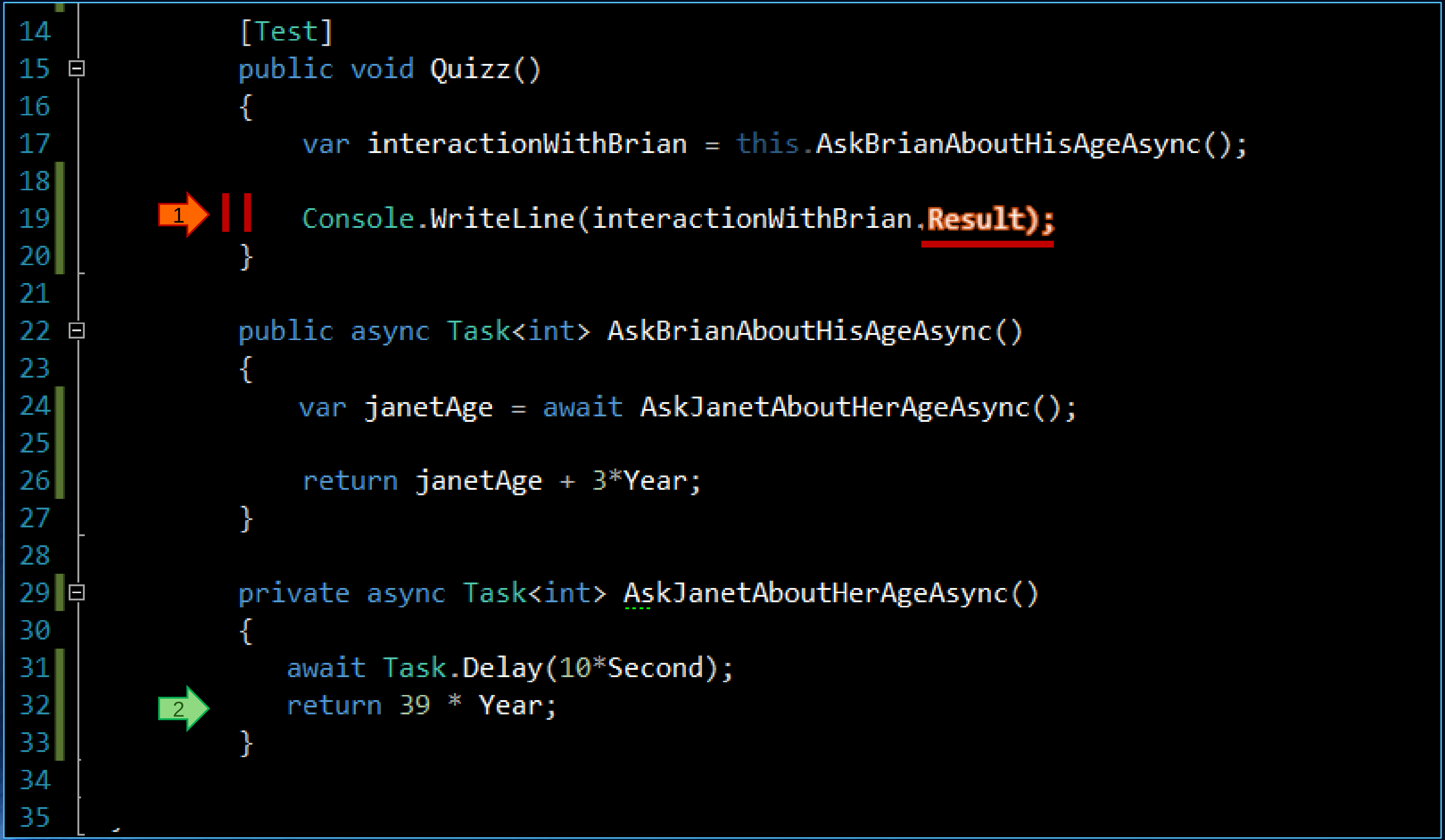
```
[Test]
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

LET'S START WITH A QUESTION…



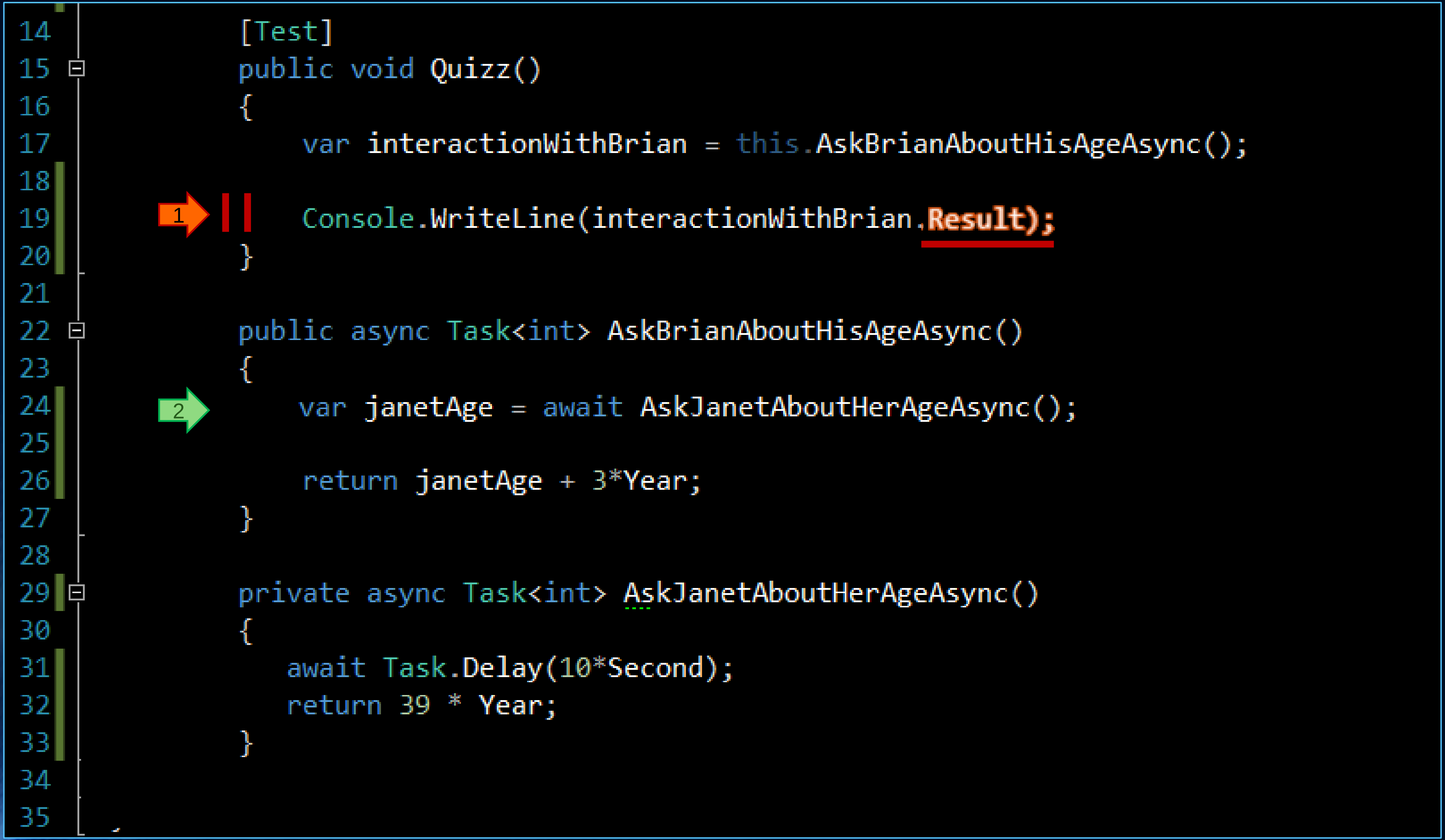
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

1 →         Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
2 →        return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



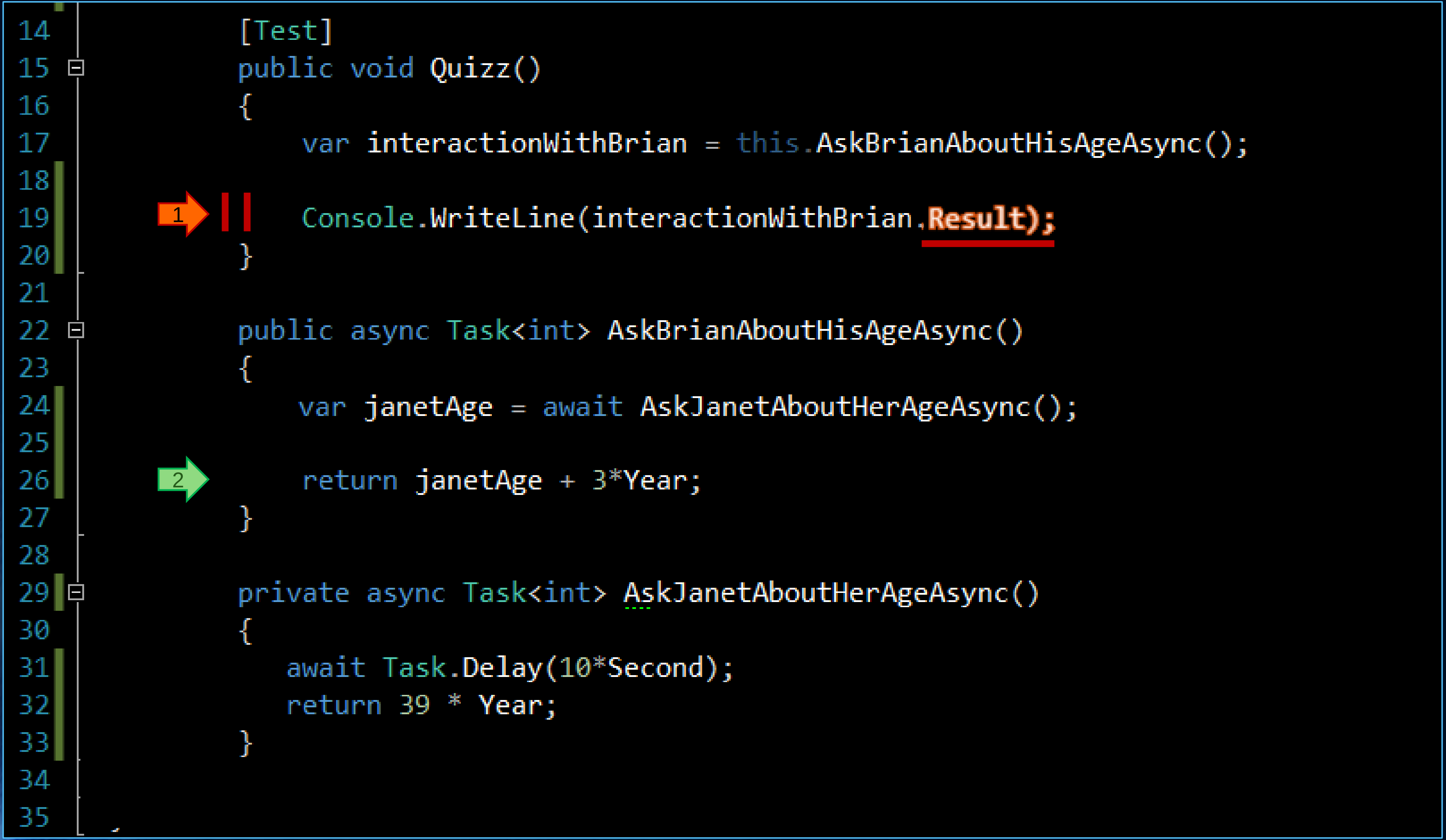
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



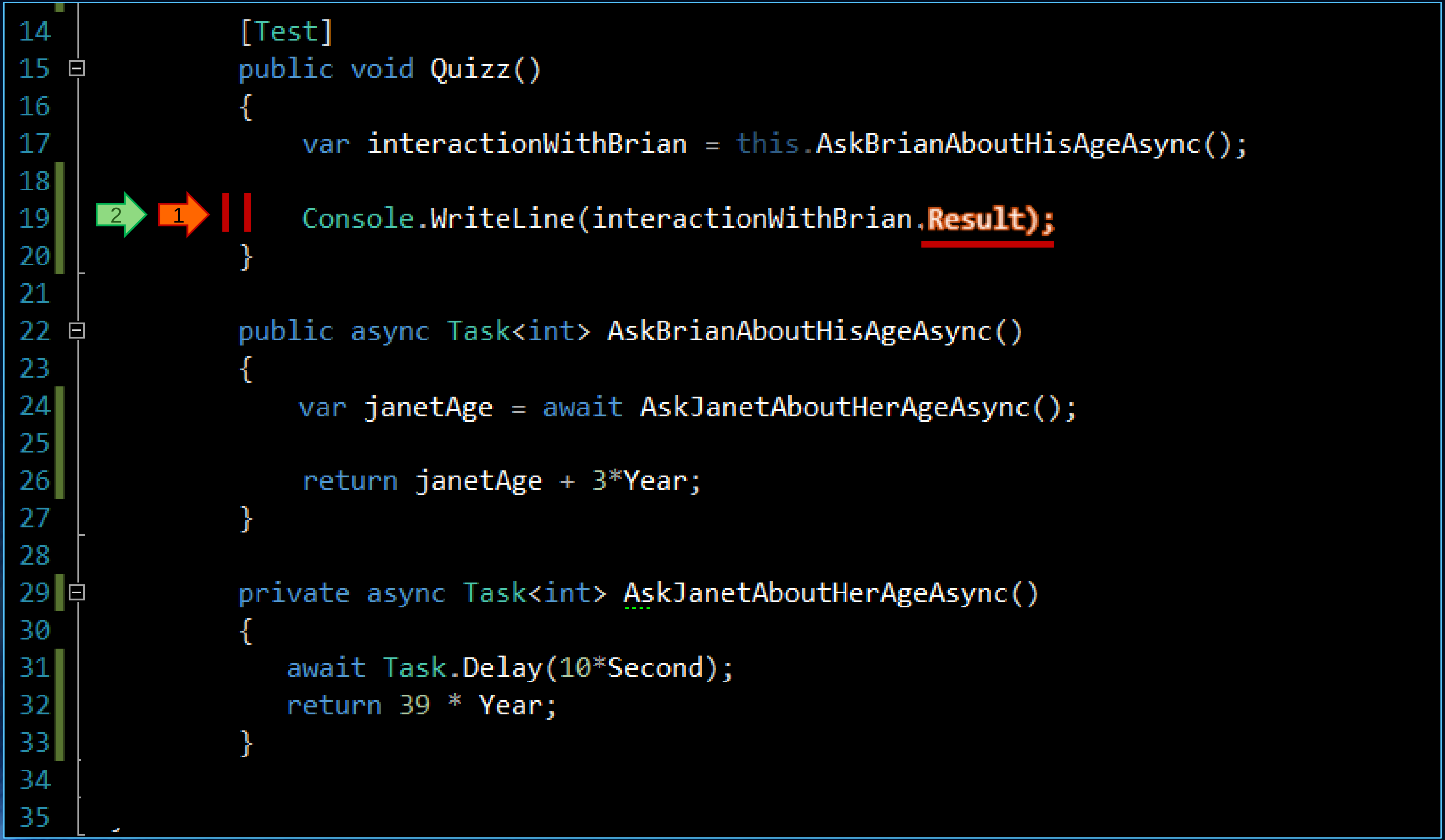
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



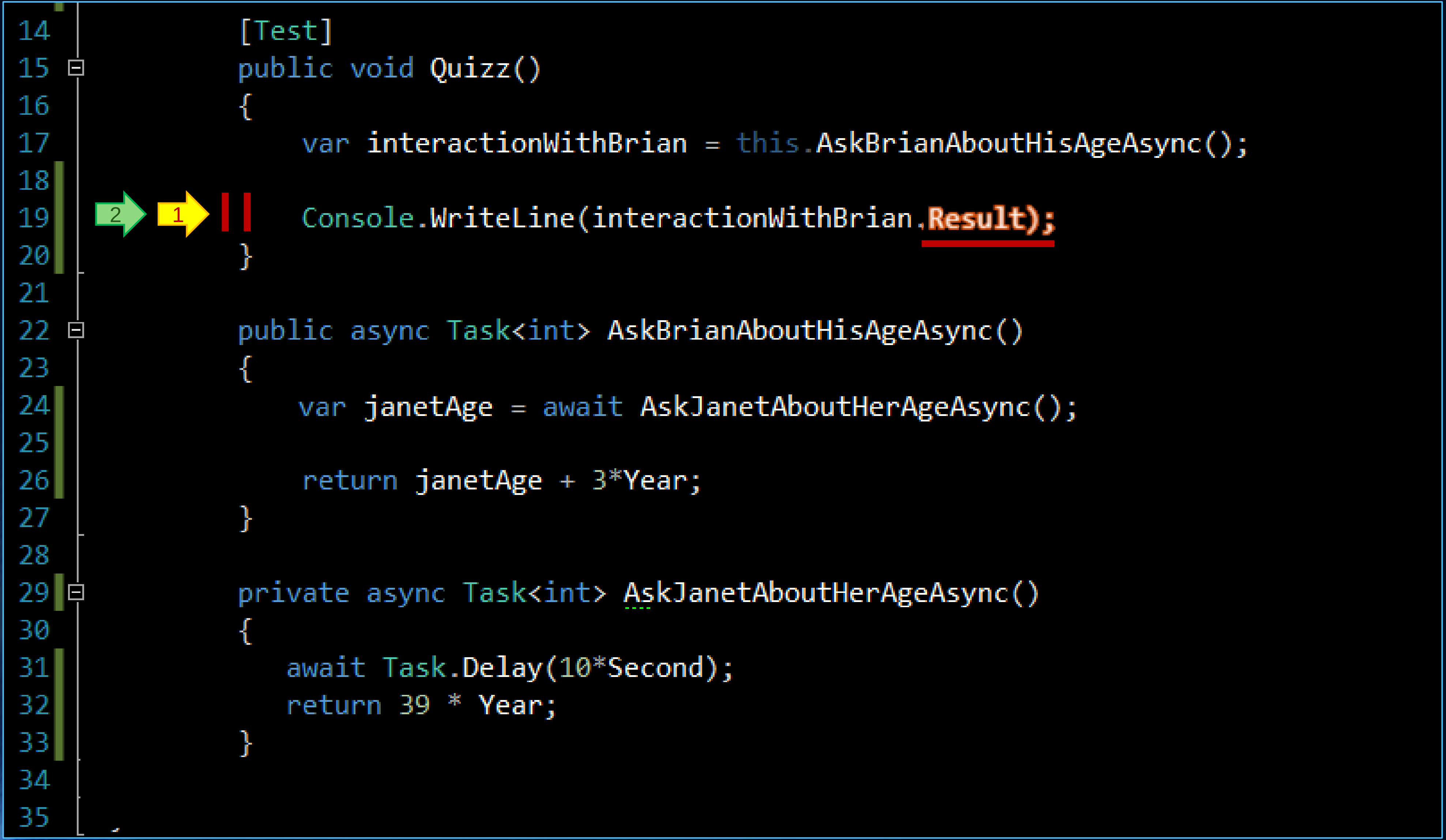
```csharp
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



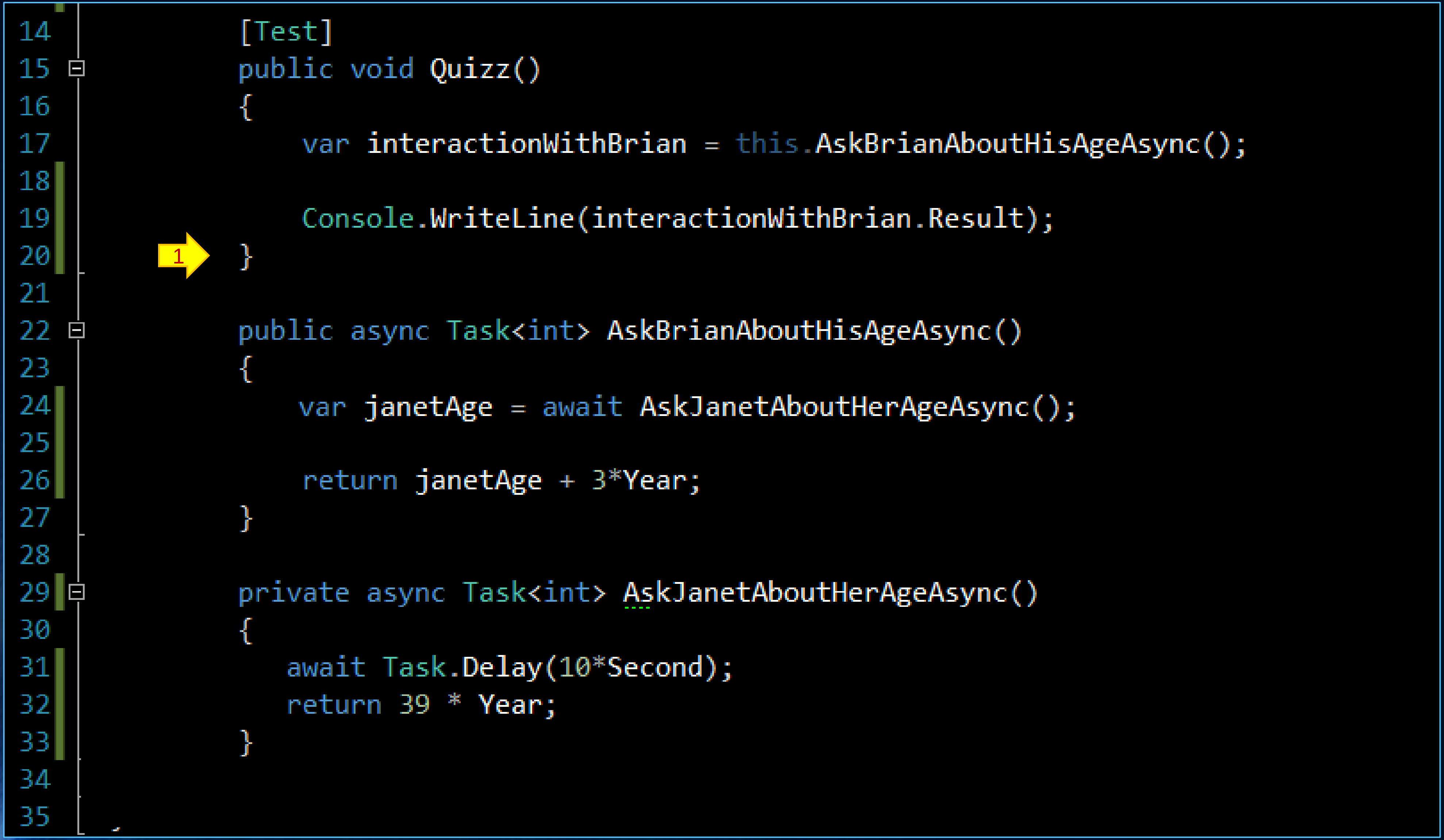
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



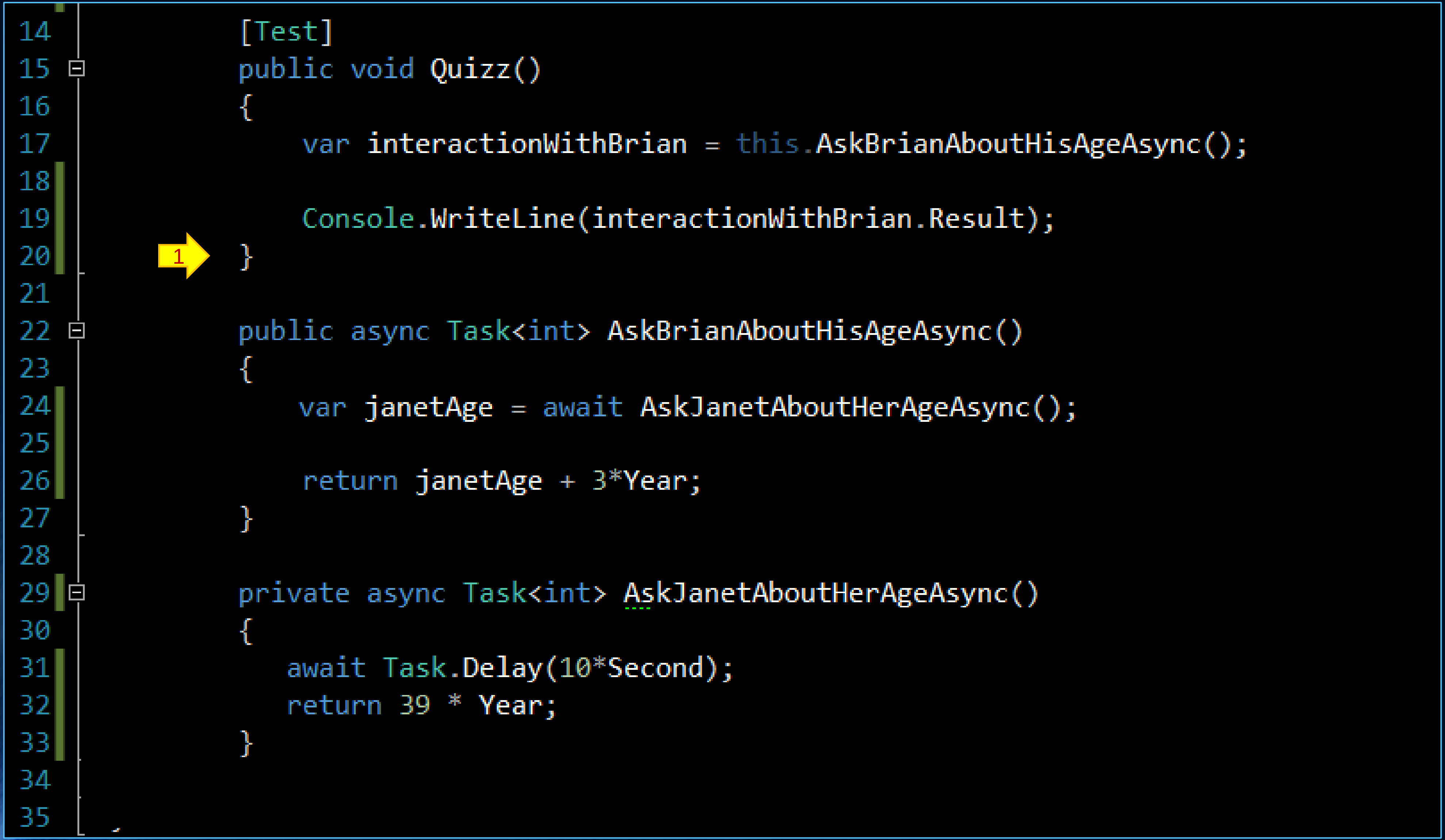
```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
           var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
          await Task.Delay(10*Second);
          return 39 * Year;
        }
```

LET'S START WITH A QUESTION…



```
        [Test]
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
 1      }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
           var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
          await Task.Delay(10*Second);
          return 39 * Year;
        }
```

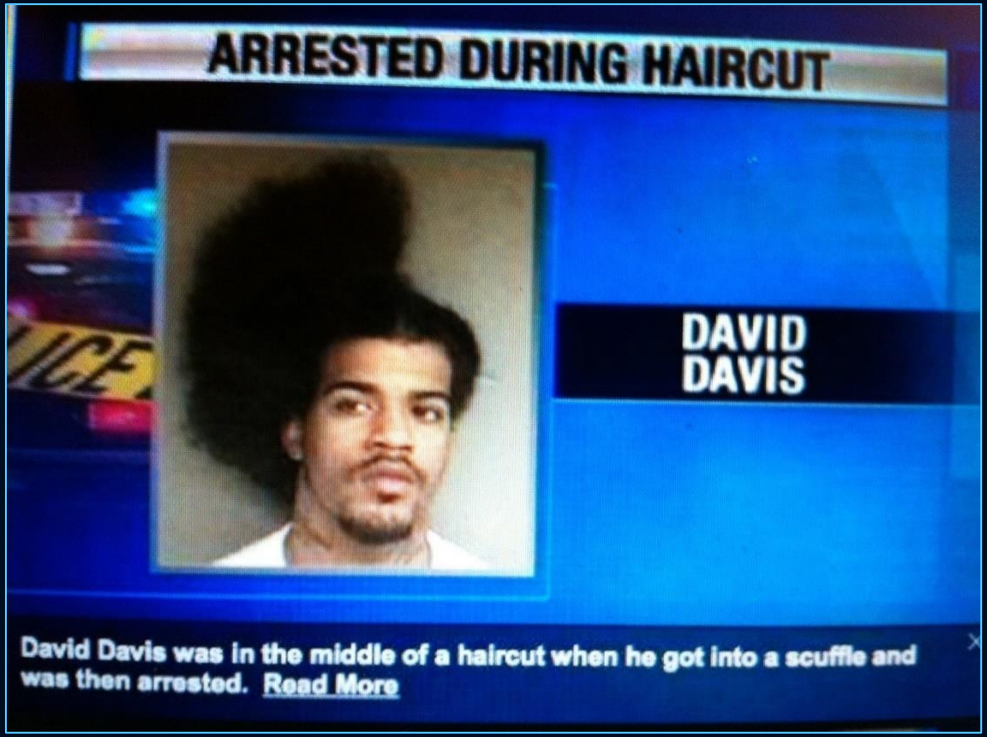
WAS IT YOUR FIRST ANSWER?

Replay?

GUESS WHAT...

SAME CODE IN ASP.NET OR UI CONTEXT

WILL DEADLOCK!


ARRESTED DURING HAIRCUT
DAVID DAVIS
David Davis was in the middle of a haircut when he got into a scuffle and was then arrested. Read More

OH… WAIT! AGENDA

1. Why
2. What
3. How
4. Pitfalls & Recommendations

CHAPTER 1: WHY

LIKE ME, YOU DON'T LIKE WASTE?

A SIMPLE RULE TO AVOID WASTE OF CPU

ASYNC-AWAIT INTENTION IS...

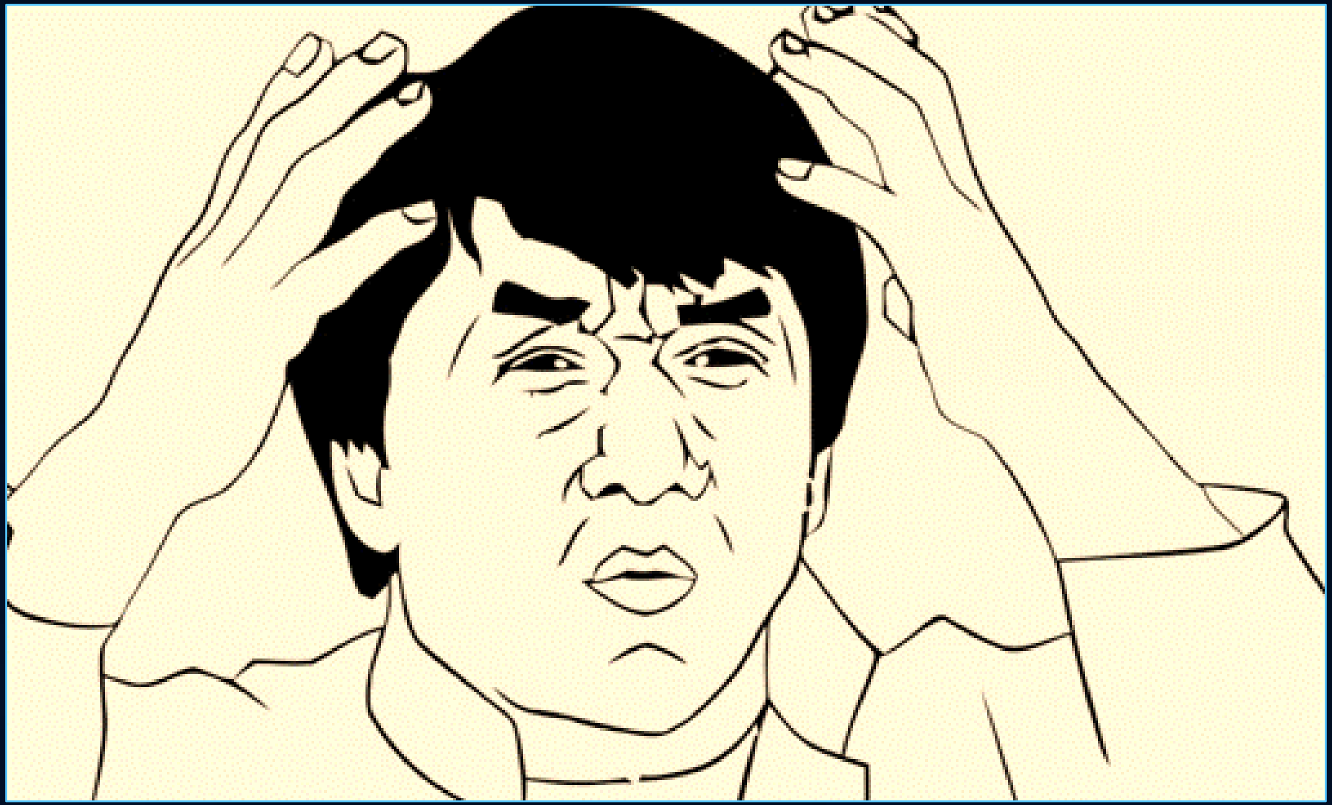
...TO EASILY TRANSFORM

SYNCHRONOUS CODE ➔ ASYNCHRONOUS CODE

(WITHOUT HURTING YOUR EXISTING CODE STRUCTURE)

WITHOUT HURTING YOUR EXISTING CODE STRUCTURE

WAIT A MINUTE… ASYNCHRONOUS, CONCURRENT, PARALLEL?



SYNCHRONOUS

- **PERFORM** something here and now.
- The caller thread will be blocked until it's done

ASYNCHRONOUS

- **INITIATE** something here and now (off-loading).
- The caller thread is released immediately
  - Free for something else
  - No waste of CPU resource ;-)

SYNCHRONOUS

- PERFORM

ASYNCHRONOUS

- INITIATE

THIS IS ABOUT INVOCATION!

(NOT ABOUT HOW THE GODDAMN THING IS EXECUTED)

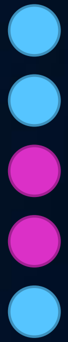
WHAT ABOUT EXECUTION

CONCURRENCY

- Multiple “*threads*” of execution
  - Independent logical segments

CONCURRENCY

- Multiple "*threads*" of execution
  - Independent logical segments

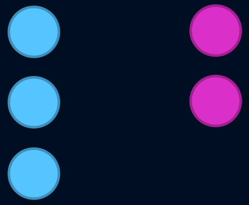


PARALLELISM

CONCURRENCY

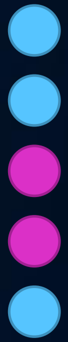
+

SIMULTANEOUS EXECUTION



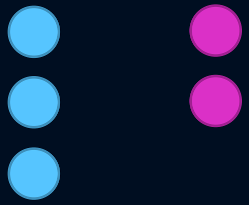
CONCURRENCY

- Multiple "*threads*" of execution
  - Independent logical segments



PARALLELISM

CONCURRENCY
+
SIMULTANEOUS EXECUTION



THIS IS ABOUT EXECUTION!

(OF THE GODDAMN THING ;-)

CHAPTER 2: WHAT

ASYNC-AWAIT IS NOT ABOUT ~~GOING ASYNC~~

ASYNC-AWAIT IS ABOUT COMPOSING THE ASYNC

COMPOSING THE ASYNC

WITH TASK CONTINUATIONS

CHAPTER 3: HOW

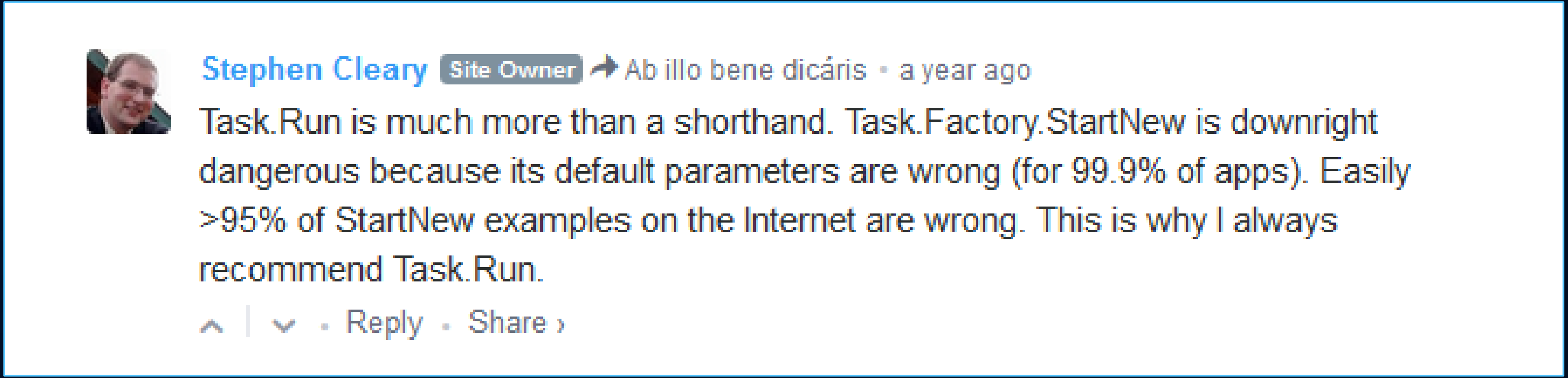
(PREREQUISITE - TPL)

TASK PARALLEL LIBRARY REMINDER (TPL)

- 3 ways to instantiate and run a TASK:
  - var task = Task.Run(lambda);
  - var task = new Task(lambda).Start();
  - Task.Factory.StartNew(lambda);

TASK PARALLEL LIBRARY REMINDER (TPL)

- 3 ways to instantiate and run a TASK:
  - var task = Task.Run(lambda); 🟢
  - var task = new Task(lambda).Start(); 🔴
  - Task.Factory.StartNew(lambda); 🔴

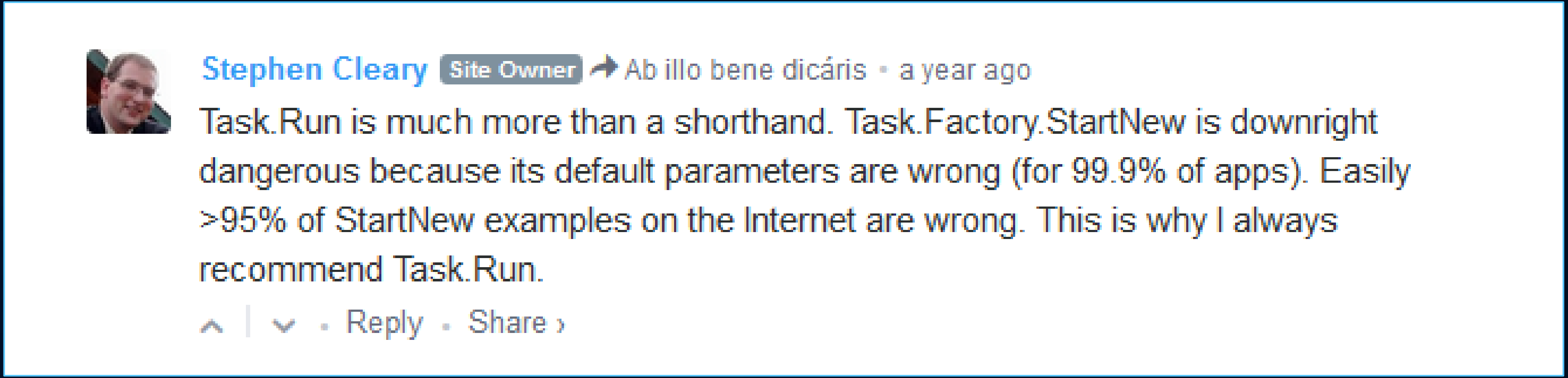


Stephen Cleary Site Owner ➜ Ab illo bene dicáris • a year ago

Task.Run is much more than a shorthand. Task.Factory.StartNew is downright dangerous because its default parameters are wrong (for 99.9% of apps). Easily >95% of StartNew examples on the Internet are wrong. This is why I always recommend Task.Run.

TASK PARALLEL LIBRARY REMINDER (TPL)

- 3 ways to instantiate and run a TASK:
  - var task = Task.Run(lambda); 🟢
  - var task = new Task(lambda).Start(); 🔴
  - Task.Factory.StartNew(lambda); 🔴



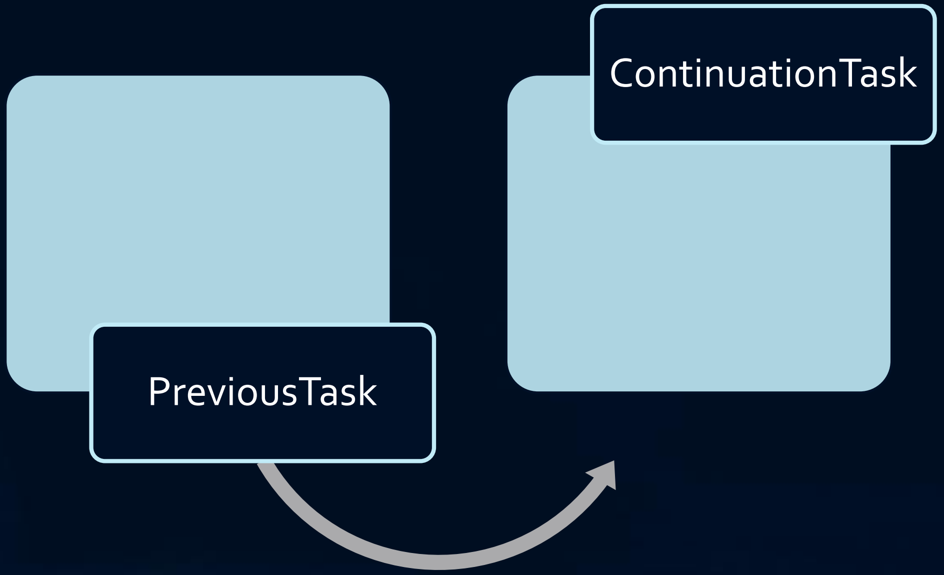
Stephen Cleary Site Owner ➦ Ab illo bene dicáris • a year ago

Task.Run is much more than a shorthand. Task.Factory.StartNew is downright dangerous because its default parameters are wrong (for 99.9% of apps). Easily >95% of StartNew examples on the Internet are wrong. This is why I always recommend Task.Run.

- task.Wait(), task.Result & task.Exception (all blocking ;-(🔴

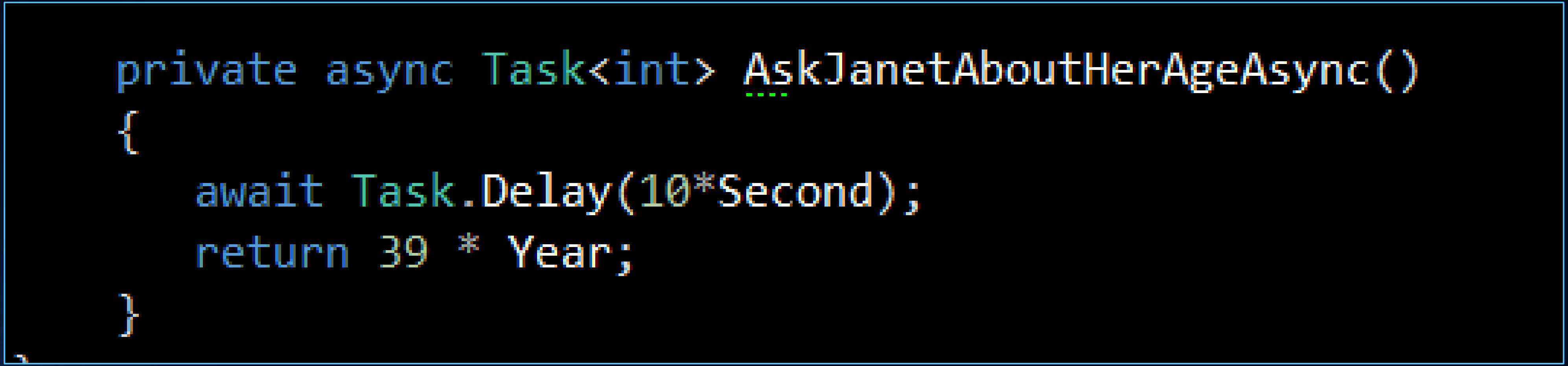
TASK PARALLEL LIBRARY REMINDER (TPL)

- CONTINUATION
  - A Task that will be achieve once a previous Task has finished
  - var continuationTask = previousTask.ContinueWith(lambda);



ASYNC-AWAIT

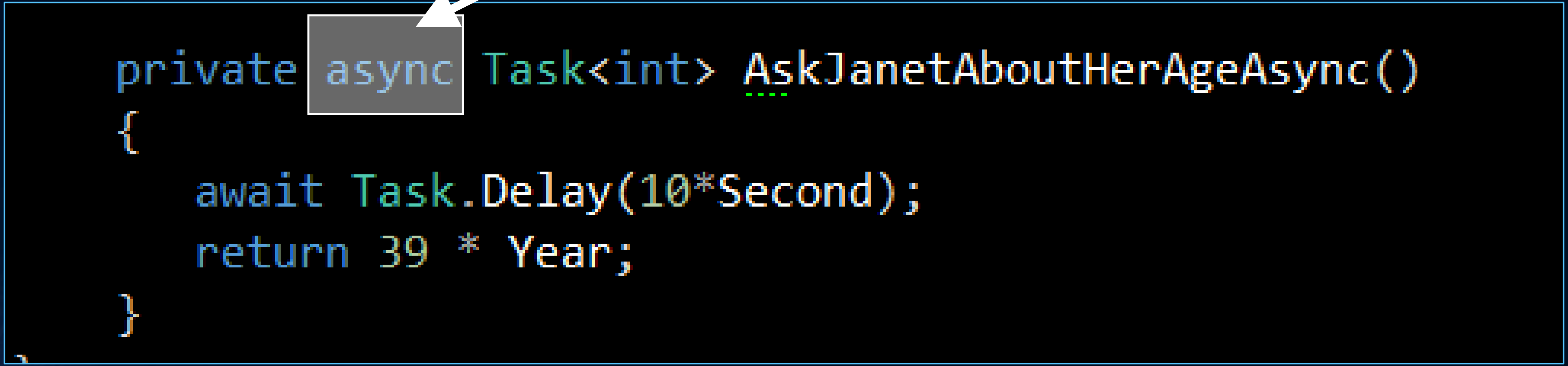
ASYNC-AWAIT



```
private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

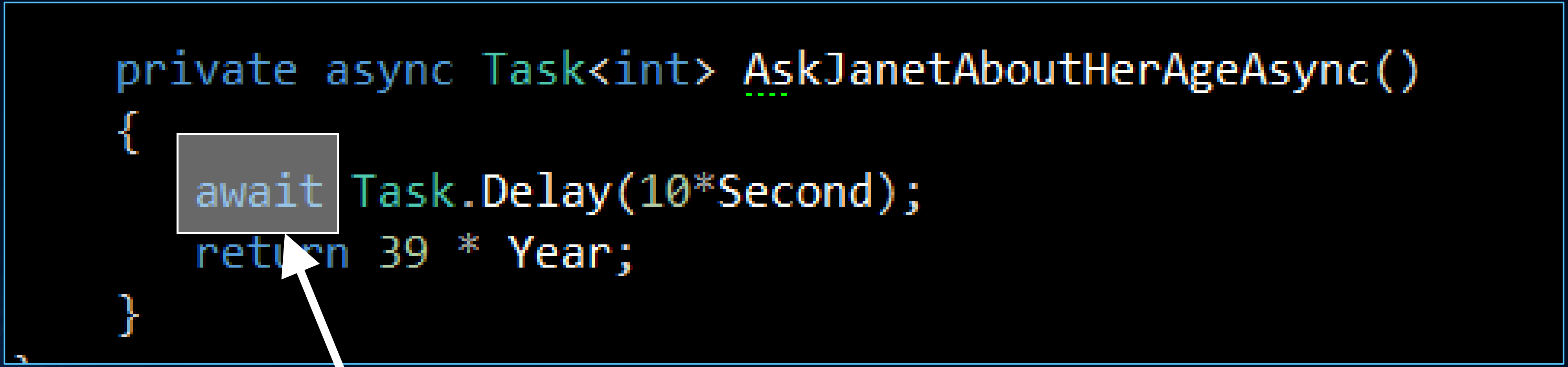
ASYNC

GENERATES STATE MACHINE



```
private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

AWAIT

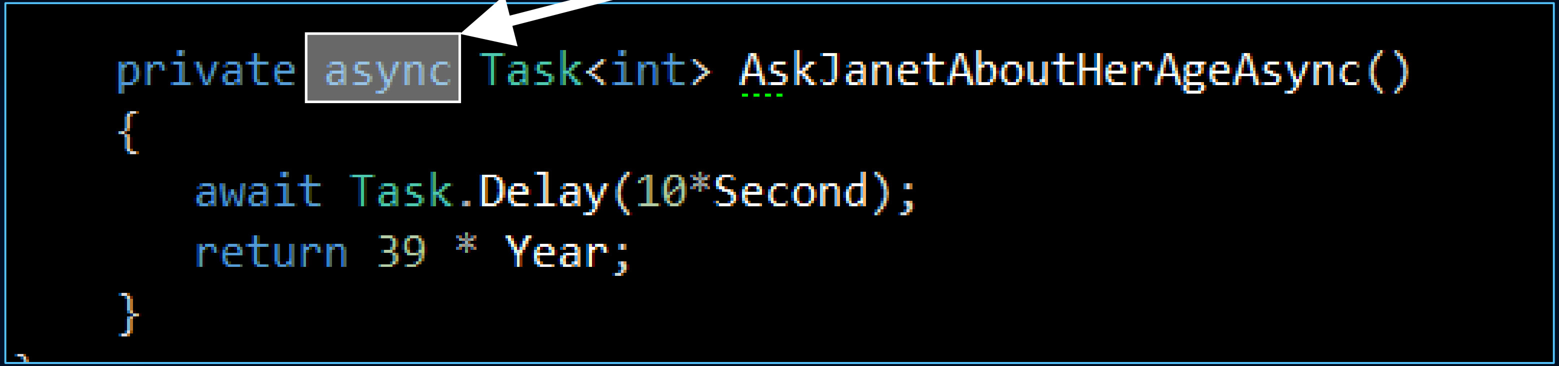


```
private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

MARKS A CONTINUATION

ASYNC

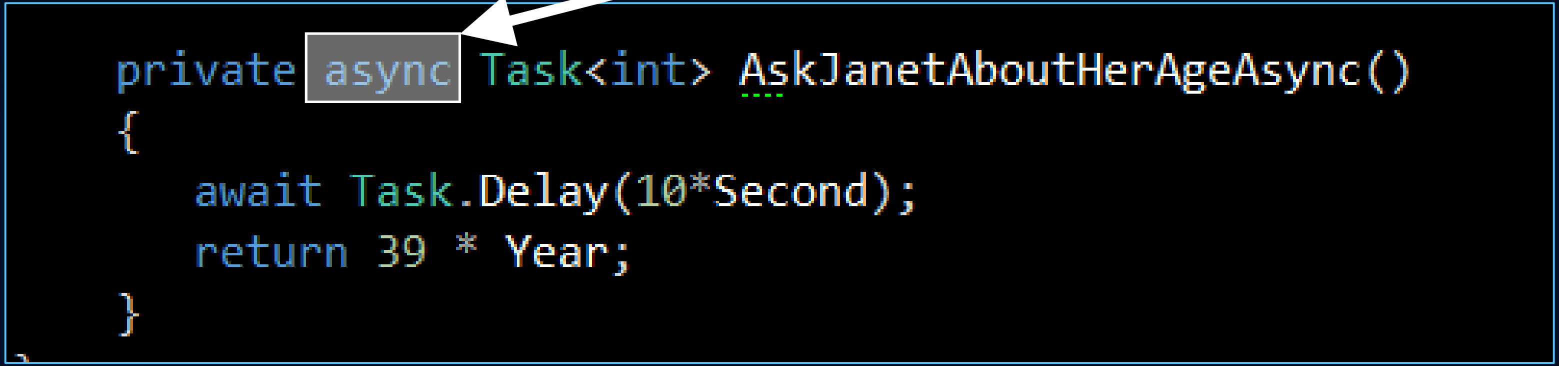
ASYNC



```
private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

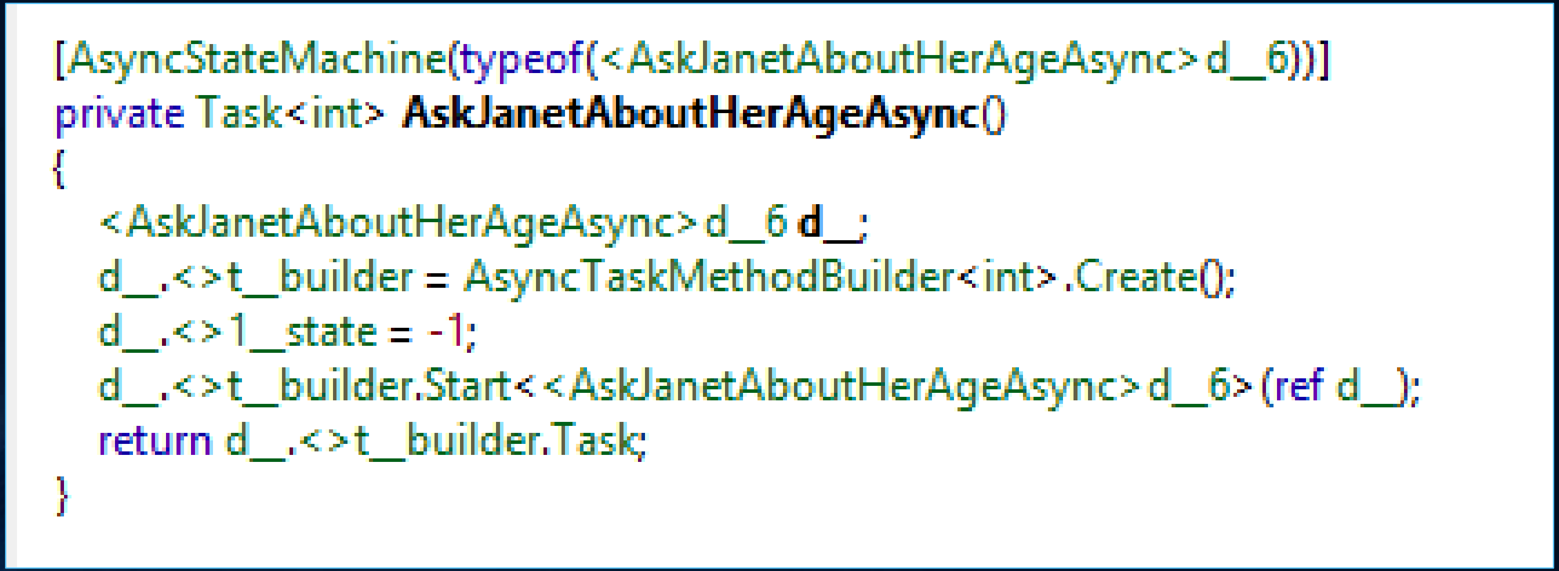
ASYNC

GENERATES STATE MACHINE



```
private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

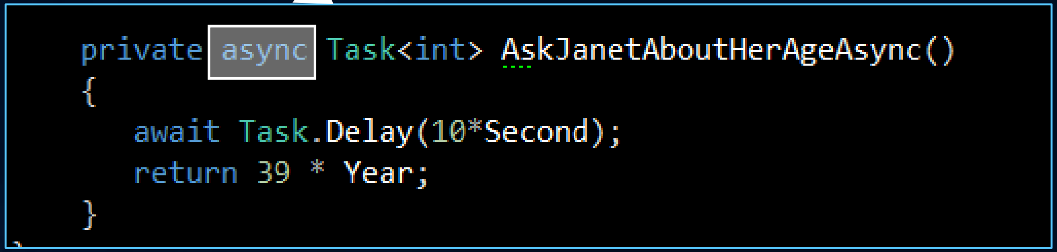
AT ↓ COMPILE TIME



```
[AsyncStateMachine(typeof(<AskJanetAboutHerAgeAsync>d__6))]
private Task<int> AskJanetAboutHerAgeAsync()
{
    <AskJanetAboutHerAgeAsync>d__6 d__;
    d__.<>t__builder = AsyncTaskMethodBuilder<int>.Create();
    d__.<>1__state = -1;
    d__.<>t__builder.Start<<AskJanetAboutHerAgeAsync>d__6>(ref d__);
    return d__.<>t__builder.Task;
}
```

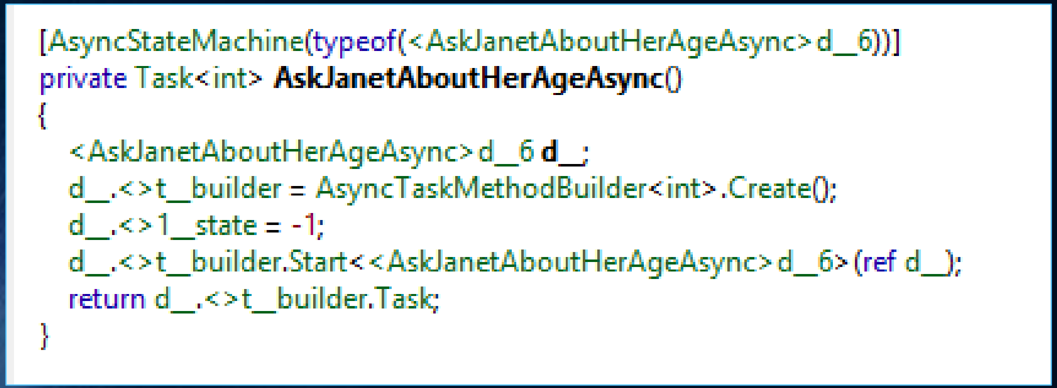
ASYNC

GENERATES STATE MACHINE



```
private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

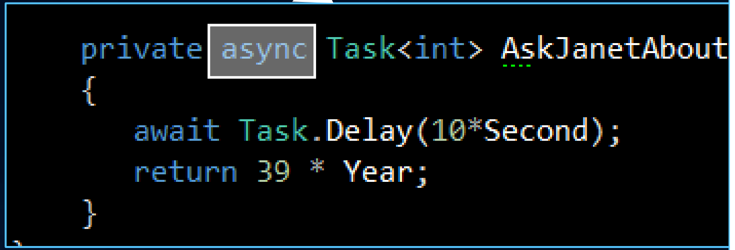
AT COMPILE TIME



```
[AsyncStateMachine(typeof(<AskJanetAboutHerAgeAsync>d__6))]
private Task<int> AskJanetAboutHerAgeAsync()
{
   <AskJanetAboutHerAgeAsync>d__6 d__;
   d__.<>t__builder = AsyncTaskMethodBuilder<int>.Create();
   d__.<>1__state = -1;
   d__.<>t__builder.Start<<AskJanetAboutHerAgeAsync>d__6>(ref d__);
   return d__.<>t__builder.Task;
}
```

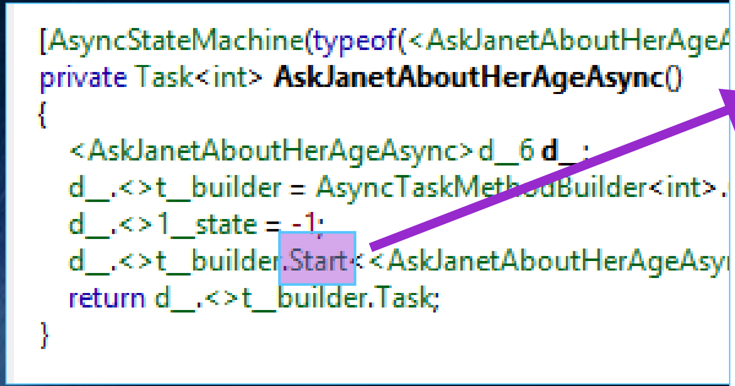
ASYNC

GENERATES STAT

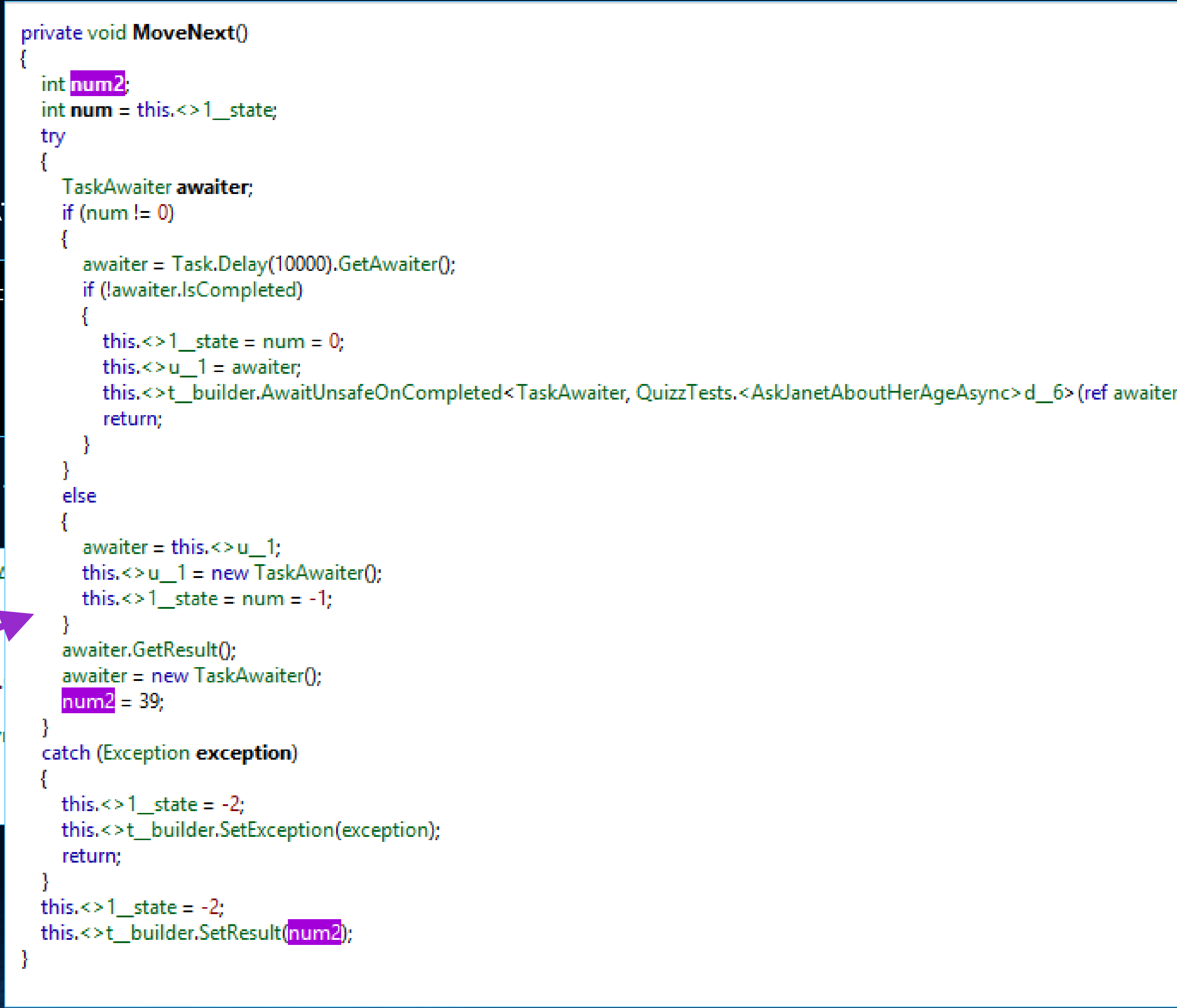


```
private async Task<int> AskJanetAbout
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

AT COMPILE



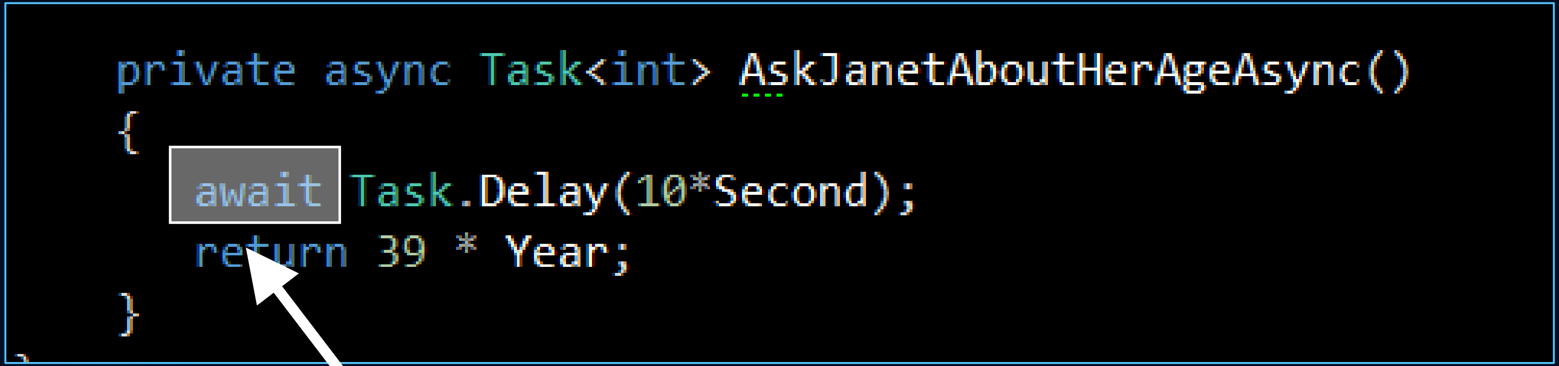
```
[AsyncStateMachine(typeof(<AskJanetAboutHerAgeA
private Task<int> AskJanetAboutHerAgeAsync()
{
   <AskJanetAboutHerAgeAsync>d__6 d__;
   d__.<>t__builder = AsyncTaskMethodBuilder<int>.
   d__.<>1__state = -1;
   d__.<>t__builder.Start<<AskJanetAboutHerAgeAsy
   return d__.<>t__builder.Task;
}
```



```
private void MoveNext()
{
   int num2;
   int num = this.<>1__state;
   try
   {
      TaskAwaiter awaiter;
      if (num != 0)
      {
         awaiter = Task.Delay(10000).GetAwaiter();
         if (!awaiter.IsCompleted)
         {
            this.<>1__state = num = 0;
            this.<>u__1 = awaiter;
            this.<>t__builder.AwaitUnsafeOnCompleted<TaskAwaiter, QuizzTests.<AskJanetAboutHerAgeAsync>d__6>(ref awaiter
            return;
         }
      }
      else
      {
         awaiter = this.<>u__1;
         this.<>u__1 = new TaskAwaiter();
         this.<>1__state = num = -1;
      }
      awaiter.GetResult();
      awaiter = new TaskAwaiter();
      num2 = 39;
   }
   catch (Exception exception)
   {
      this.<>1__state = -2;
      this.<>t__builder.SetException(exception);
      return;
   }
   this.<>1__state = -2;
   this.<>t__builder.SetResult(num2);
}
```

AWAIT

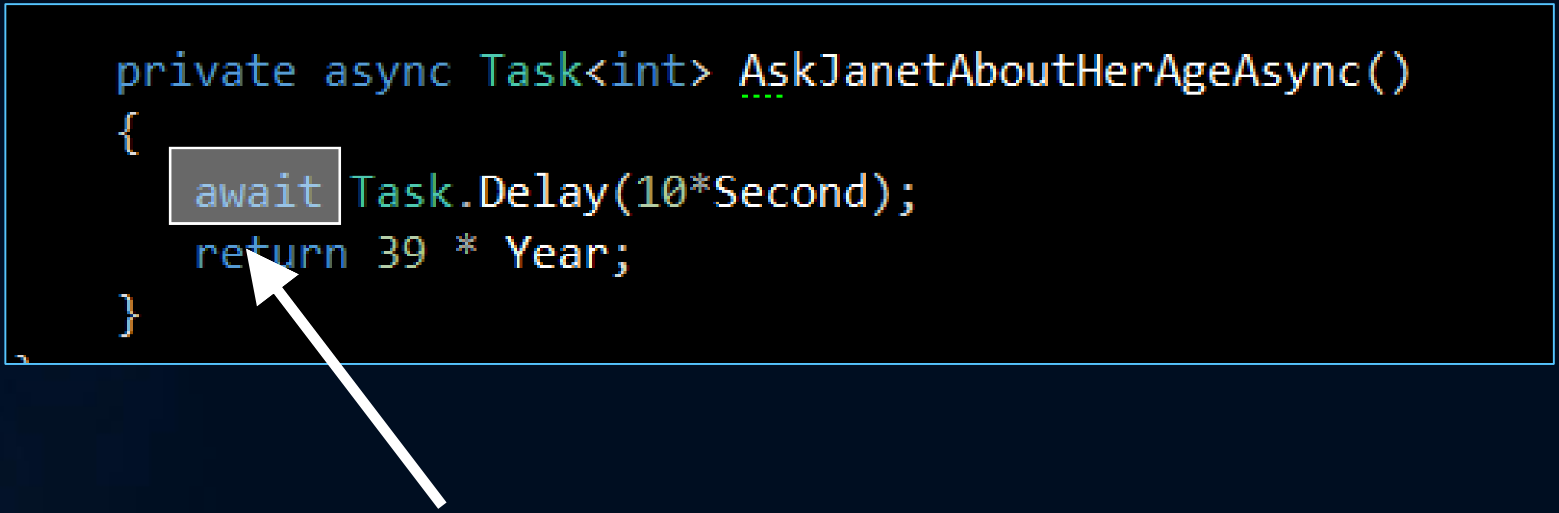
AWAIT



```
private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

MARKS A CONTINUATION

AWAIT



```
private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

RETURNS TO THE CALLER WITH A CONTINUATION TASK

WHO’S DOING THE CONTINUATION?

WELL… IT DEPENDS ;-)

IT DEPENDS ON THE AWAITER CONTEXT

The following code:

```
await FooAsync();
RestOfMethod();
```

IT DEPENDS ON THE AWAITER CONTEXT

The following code:

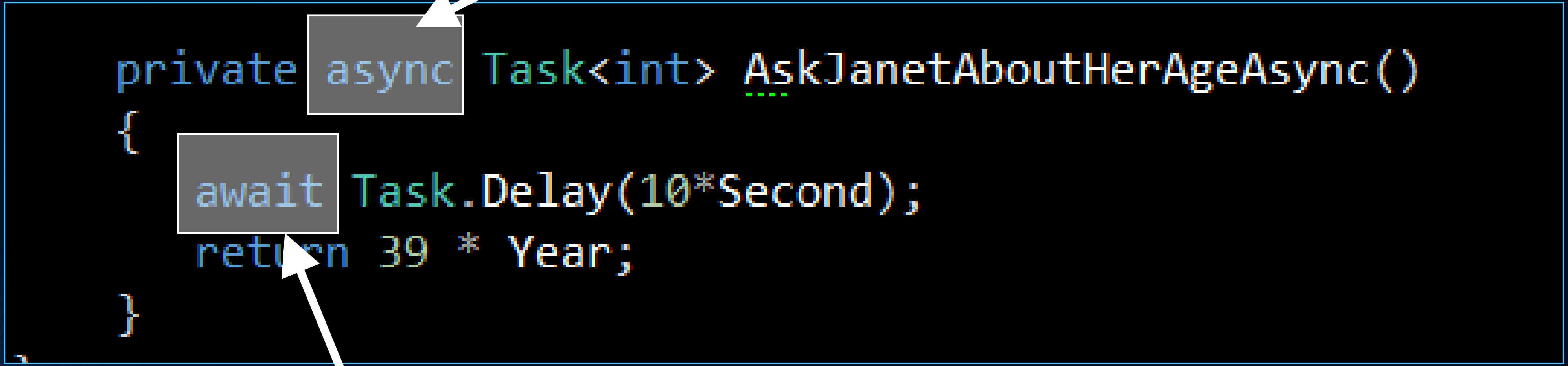
```
await FooAsync();
RestOfMethod();
```

Is equivalent to:

```
var initialTask = FooAsync();

var currentContext = SynchronizationContext.Current;
initialTask.ContinueWith(delegate
{
    if (currentContext == null)
        RestOfMethod();
    else
        currentContext.Post(delegate { RestOfMethod(); }, null);

}, null, CancellationToken.None, TaskContinuationOptions.ExecuteSynchronously, TaskScheduler.Current);
```

**WinForms,
WPF,
ASP.NET…**

ASYNC-AWAIT

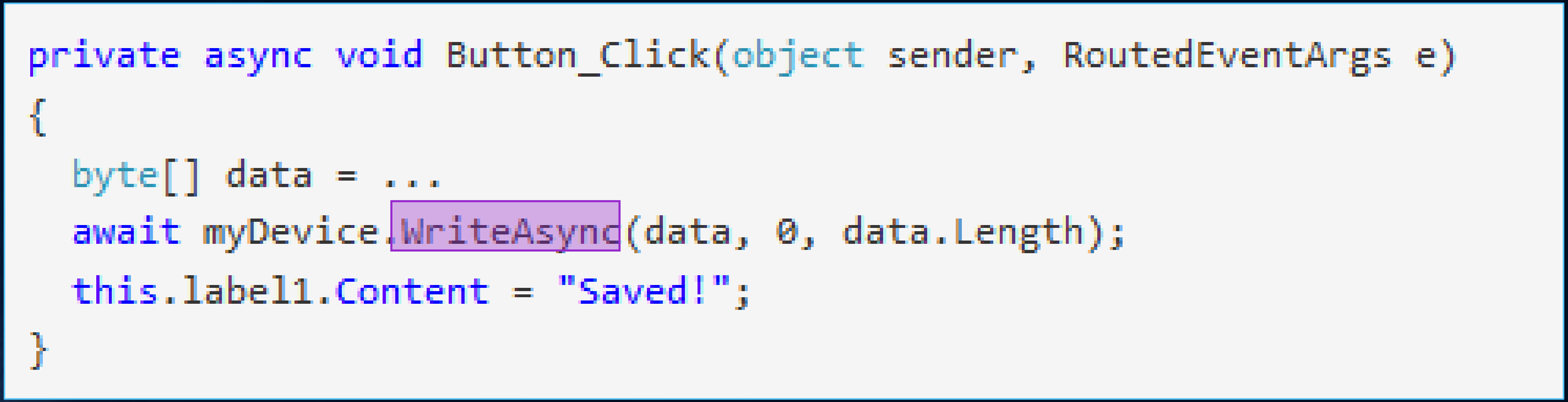


```
private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

THREAD(S) OR NO THREAD?

WINDOWS I/O: UNDER THE HOOD

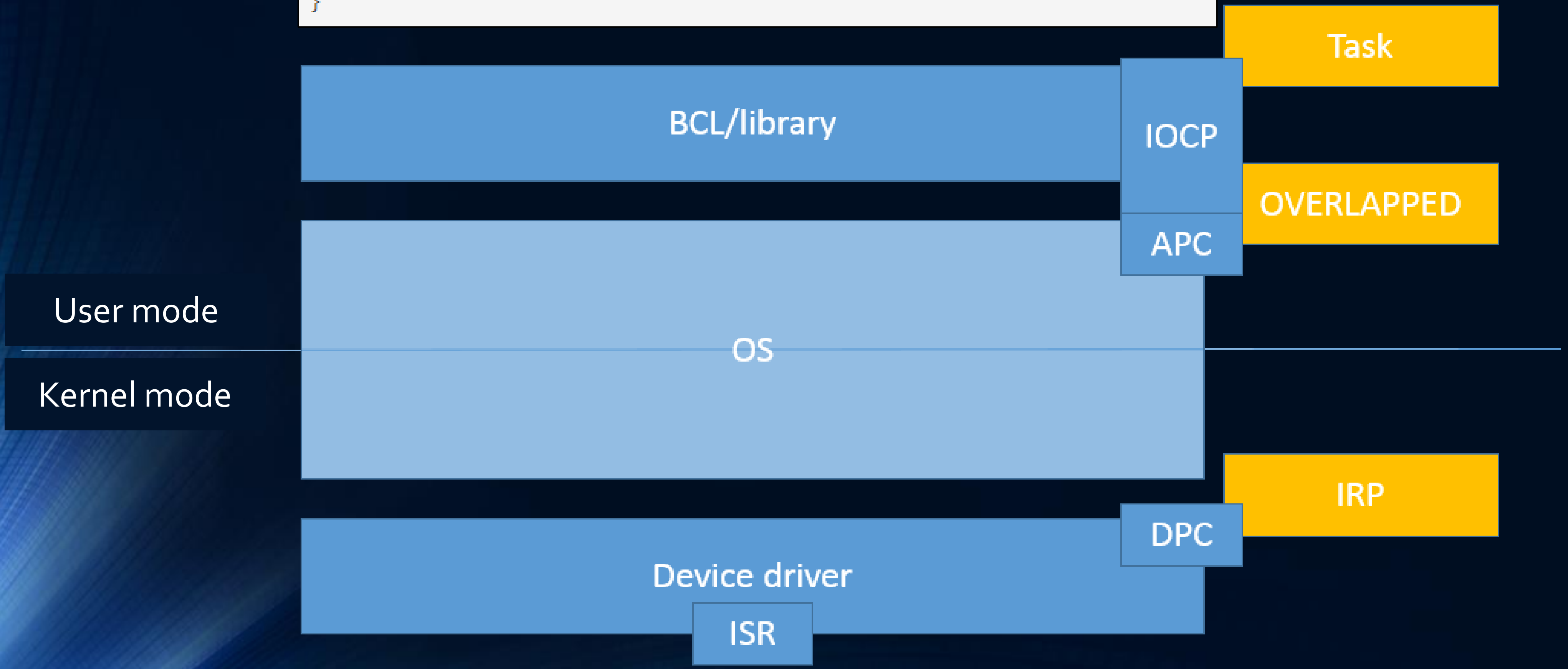
WINDOWS I/O: UNDER THE HOOD



```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```

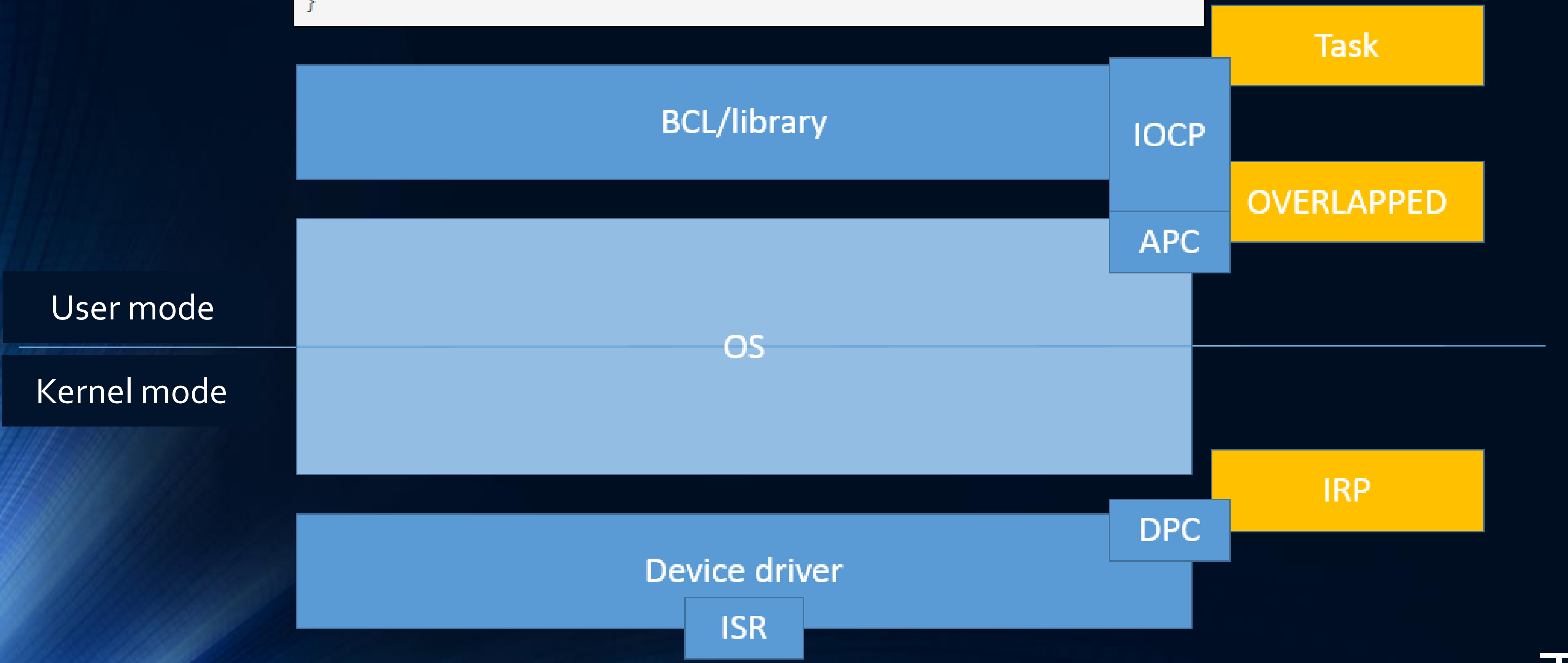
WINDOWS I/O: UNDER THE HOOD

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



WINDOWS I/O: UNDER THE HOOD

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```

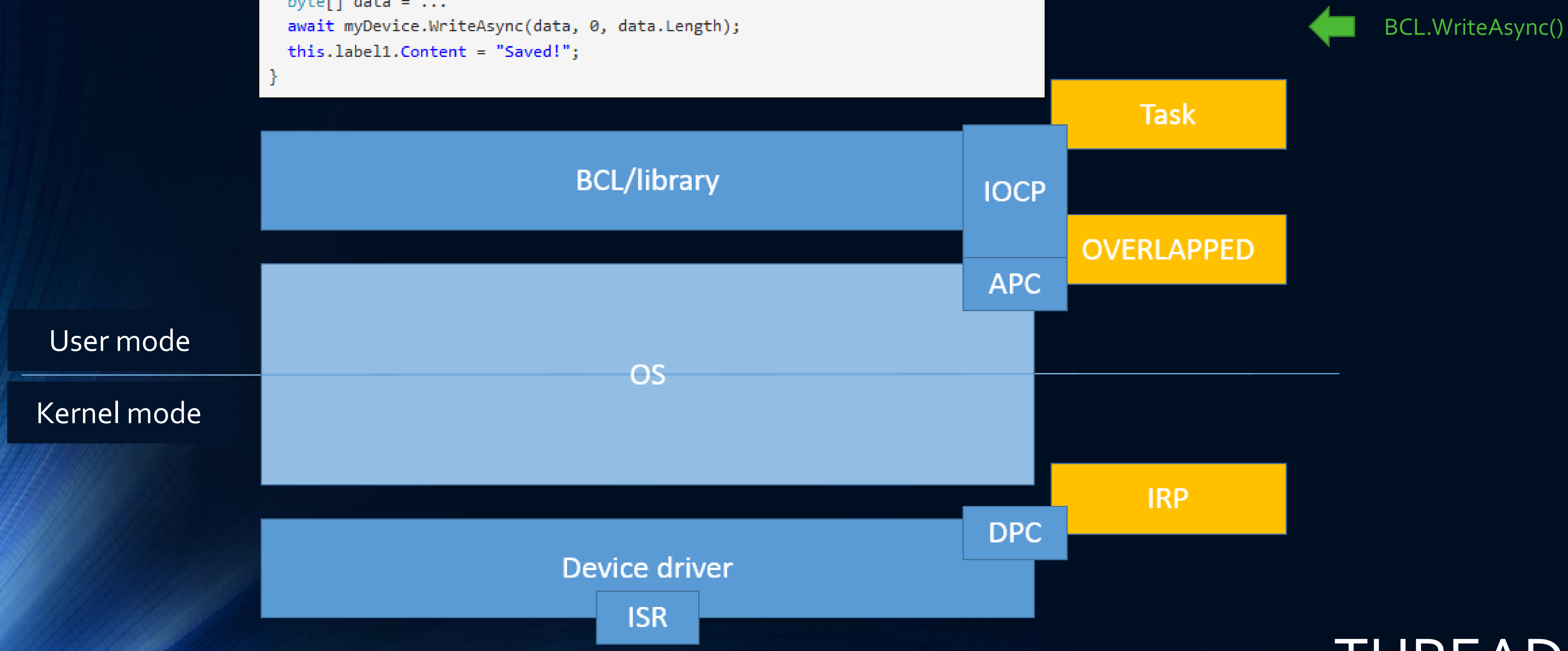


UI thread

THREAD

WINDOWS I/O: UNDER THE HOOD

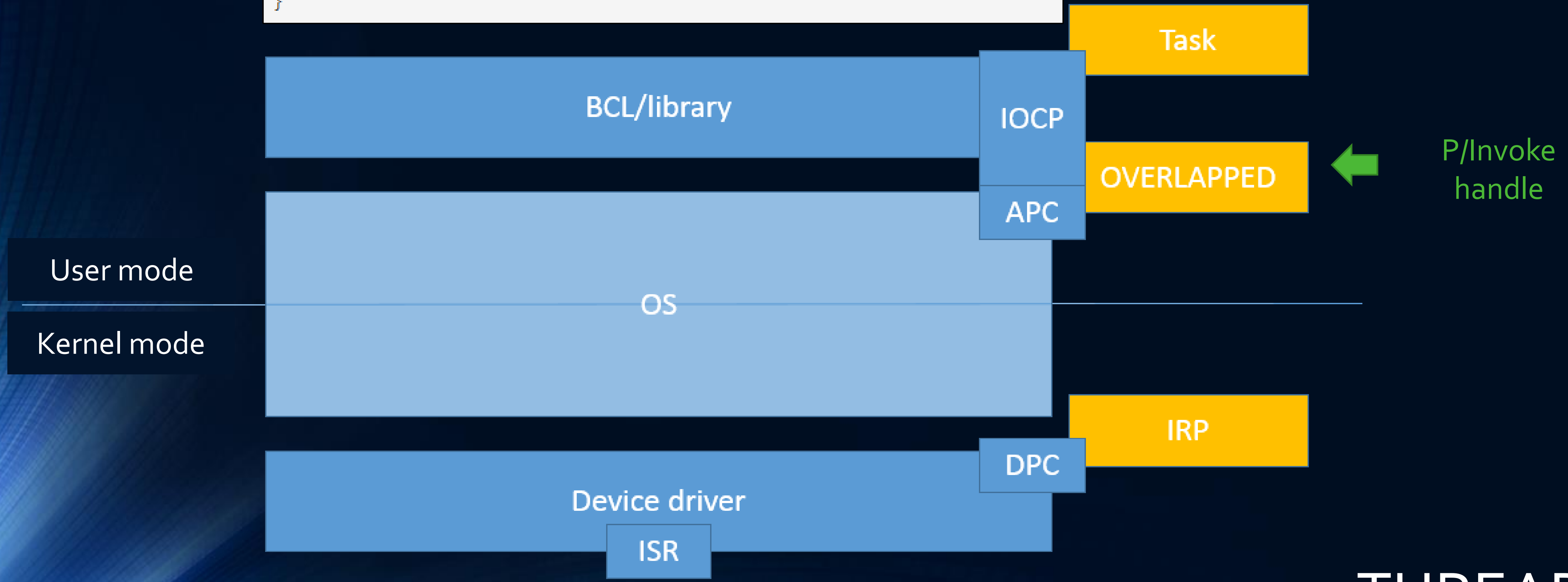
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



THREAD

WINDOWS I/O: UNDER THE HOOD

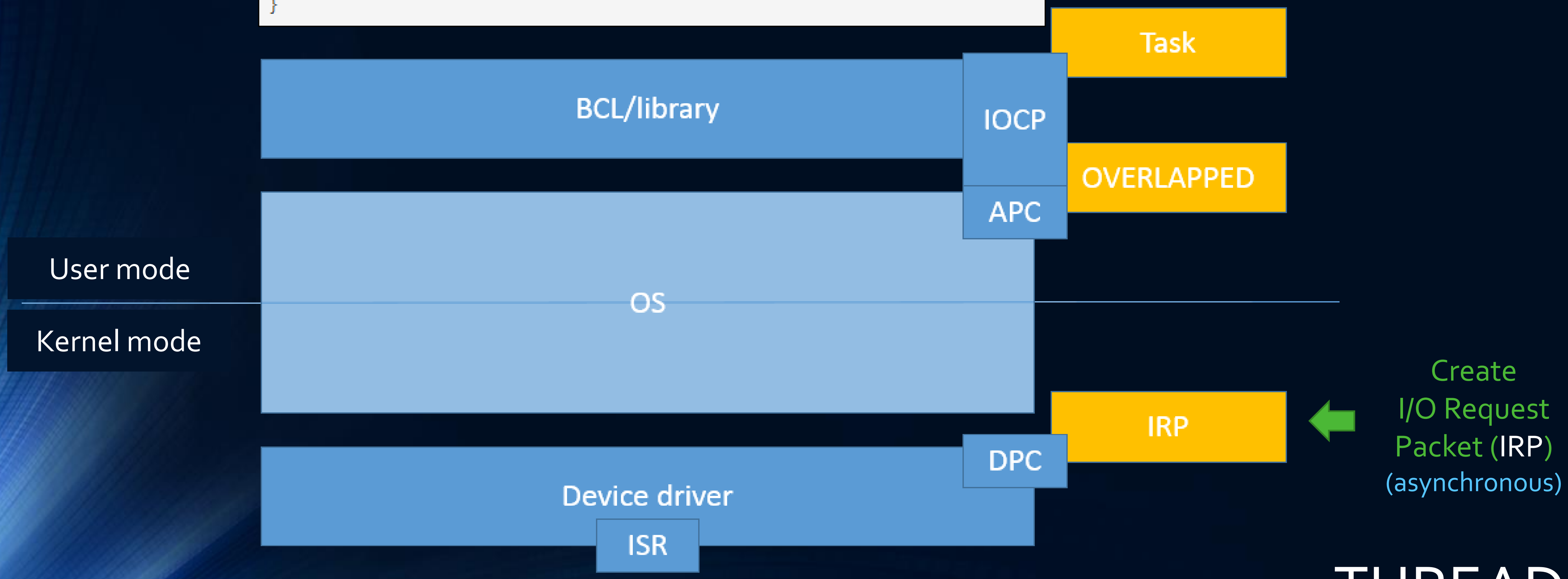
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



THREAD

WINDOWS I/O: UNDER THE HOOD

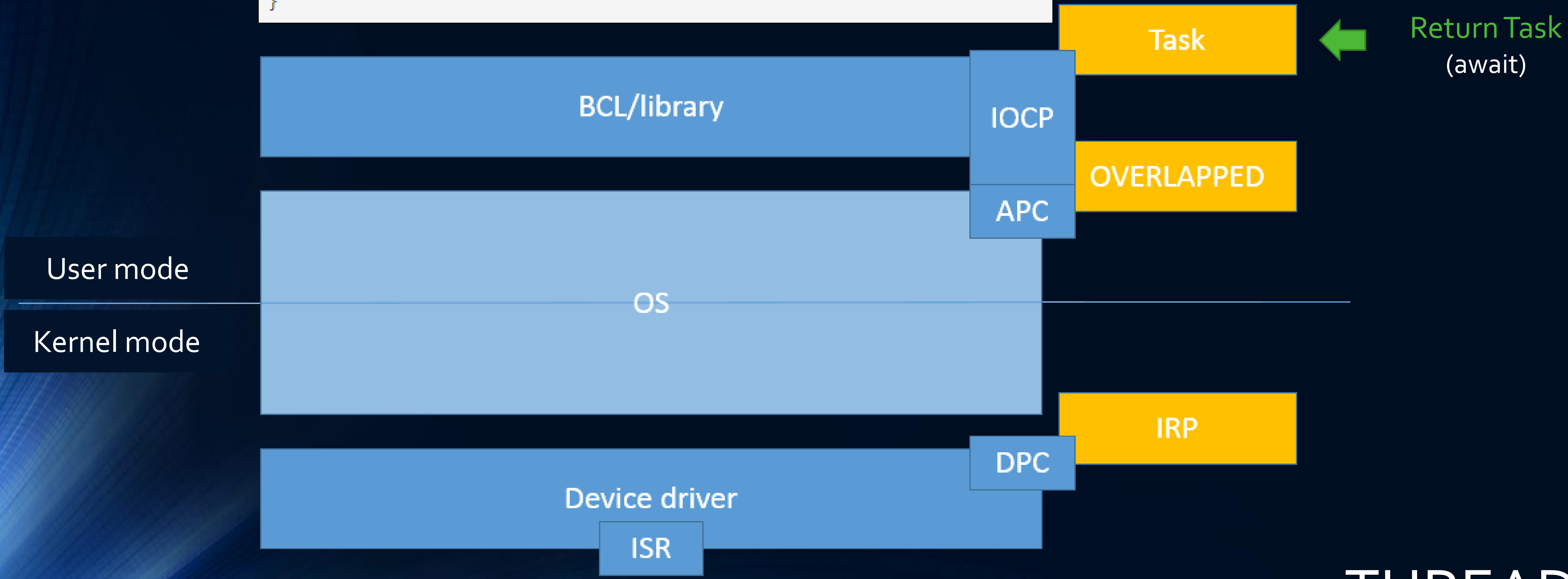
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



THREAD

WINDOWS I/O: UNDER THE HOOD

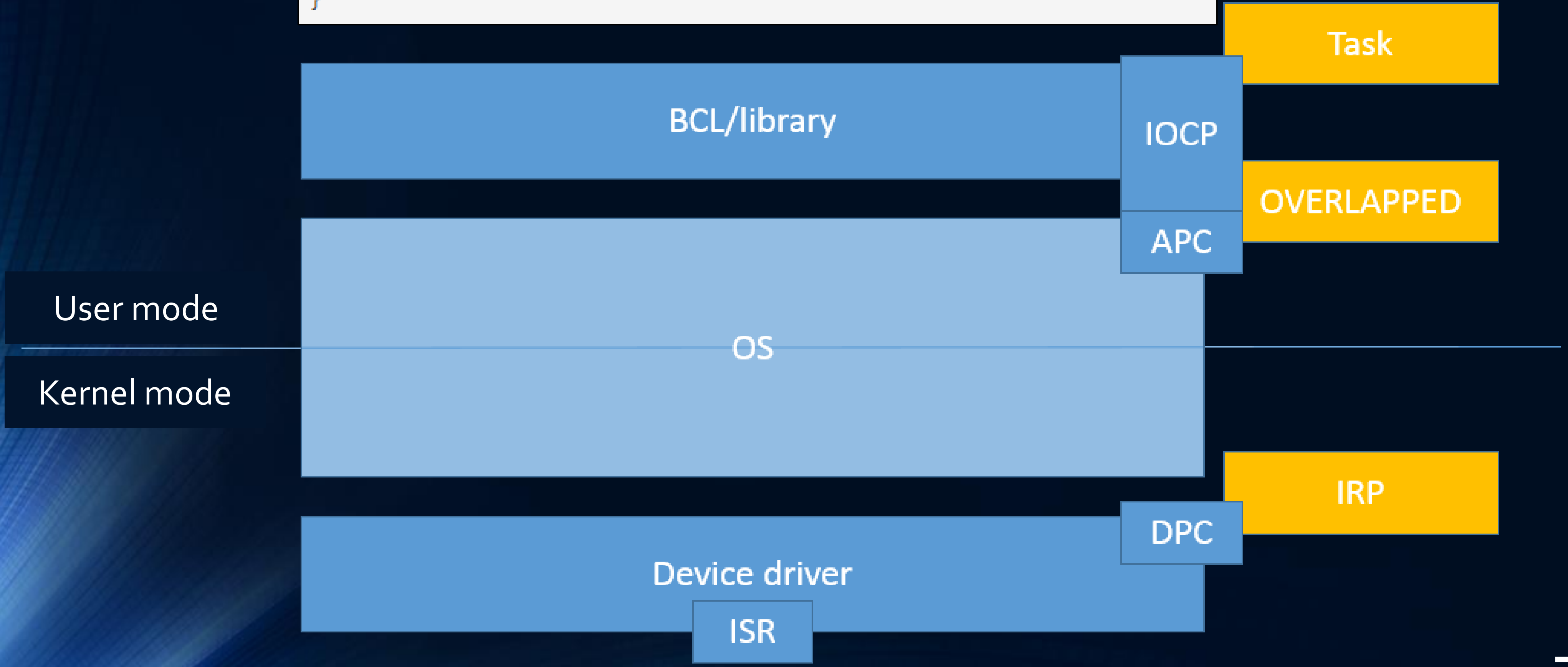
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



THREAD

WINDOWS I/O: UNDER THE HOOD

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```

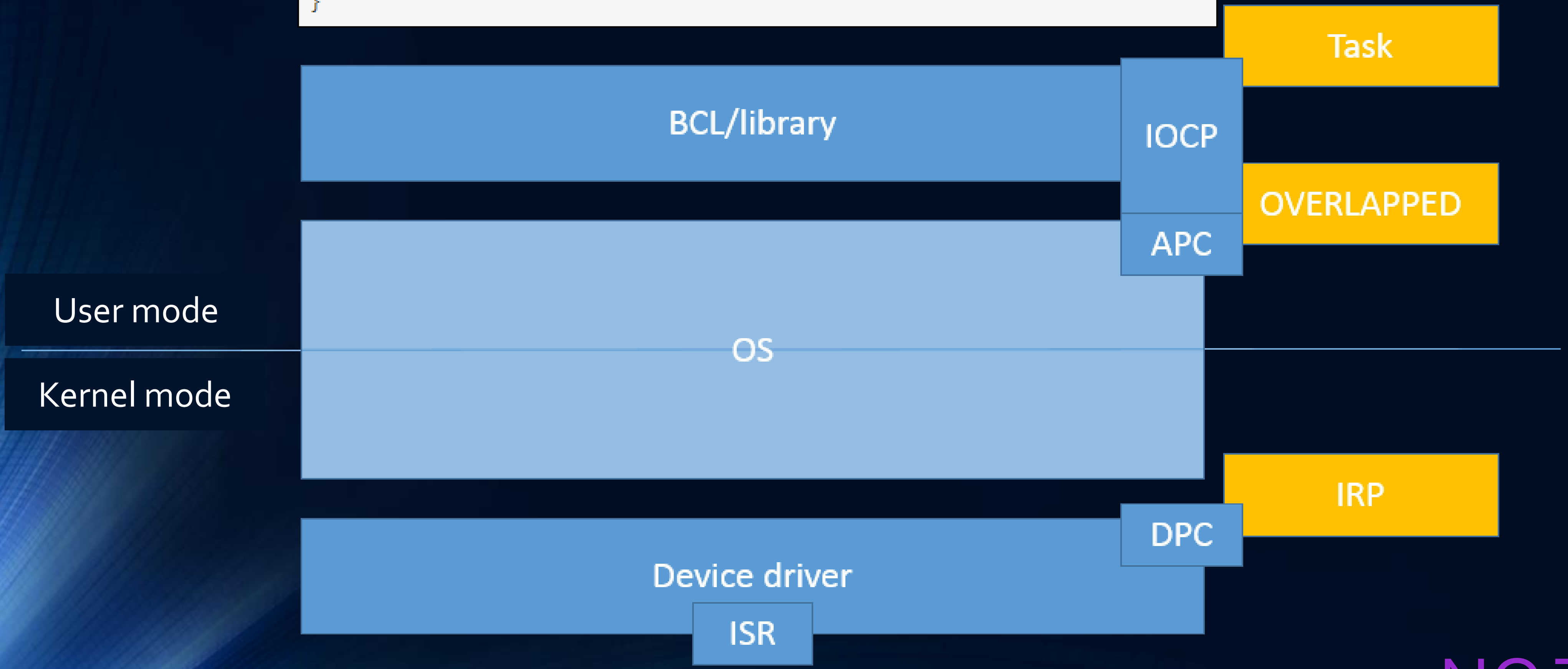


Release UI thread

THREAD

WINDOWS I/O: UNDER THE HOOD

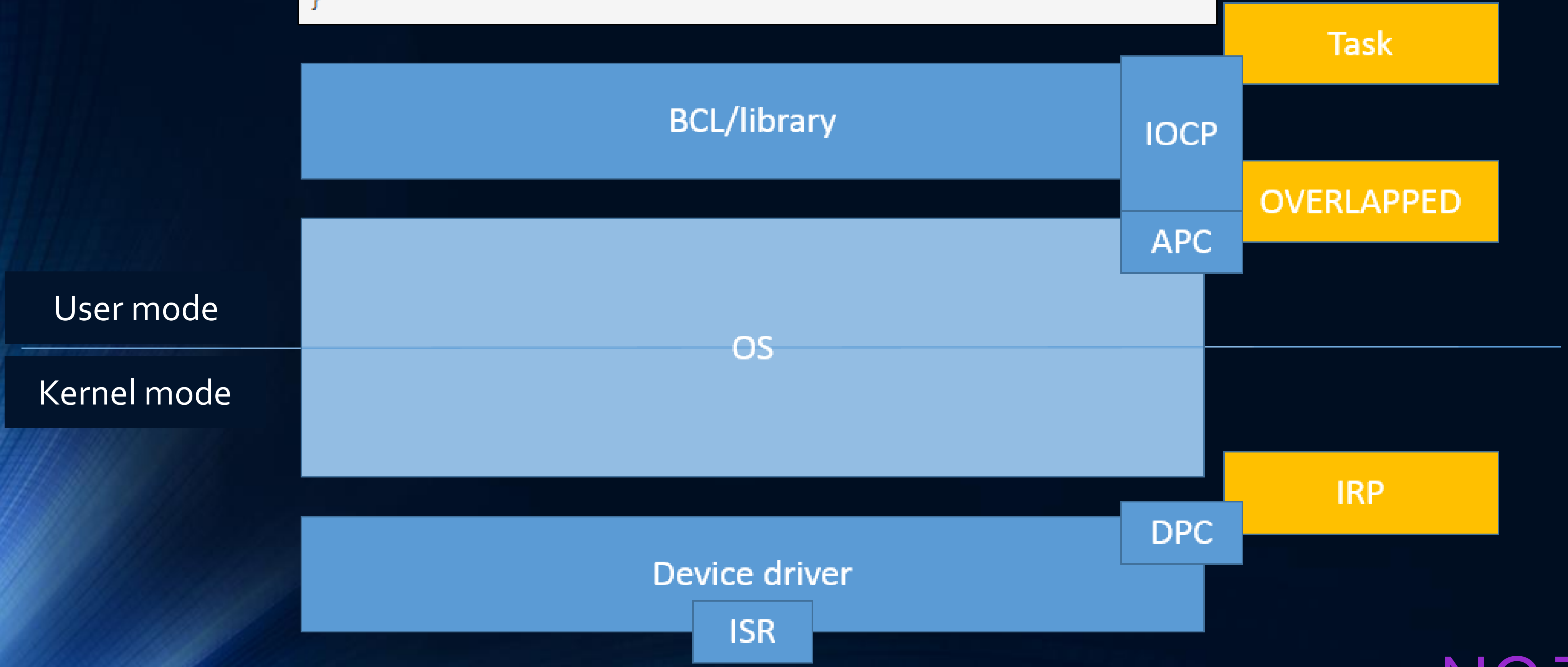
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



NO THREAD

WINDOWS I/O: UNDER THE HOOD

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```

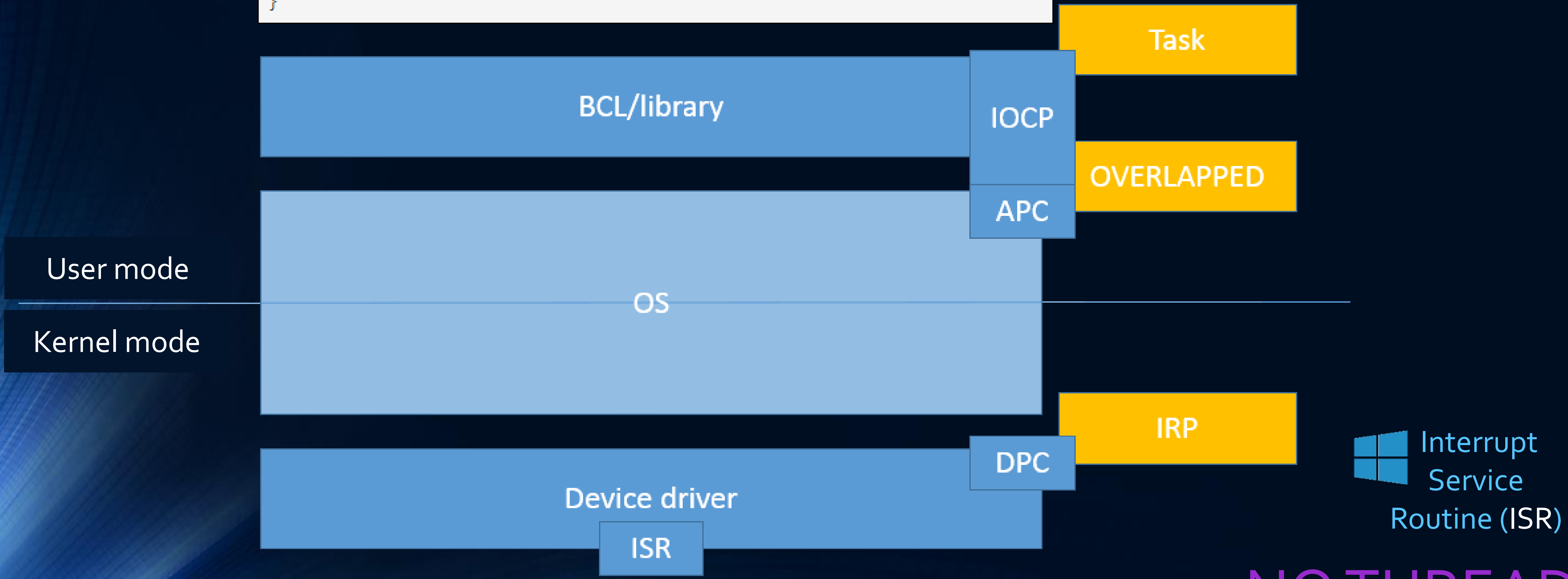


Interrupt!

NO THREAD

WINDOWS I/O: UNDER THE HOOD

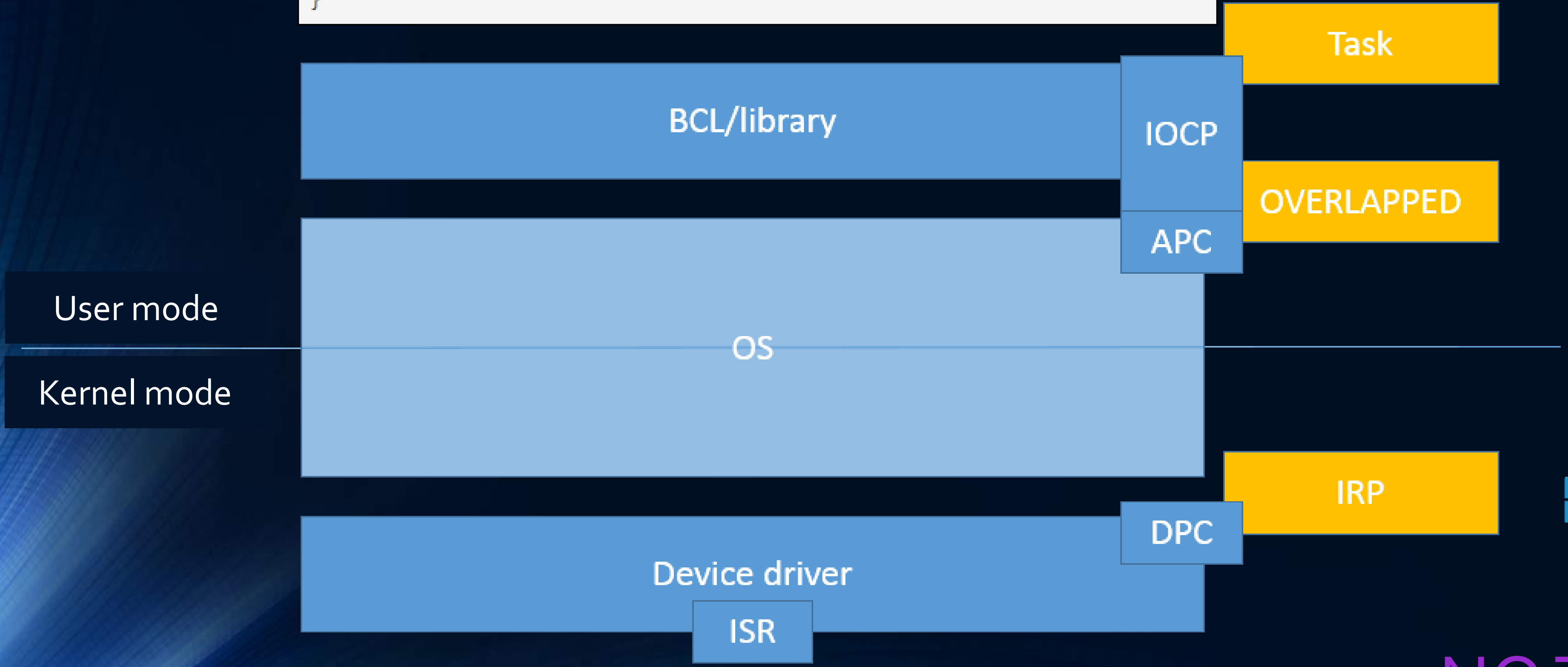
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



NO THREAD

WINDOWS I/O: UNDER THE HOOD

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```

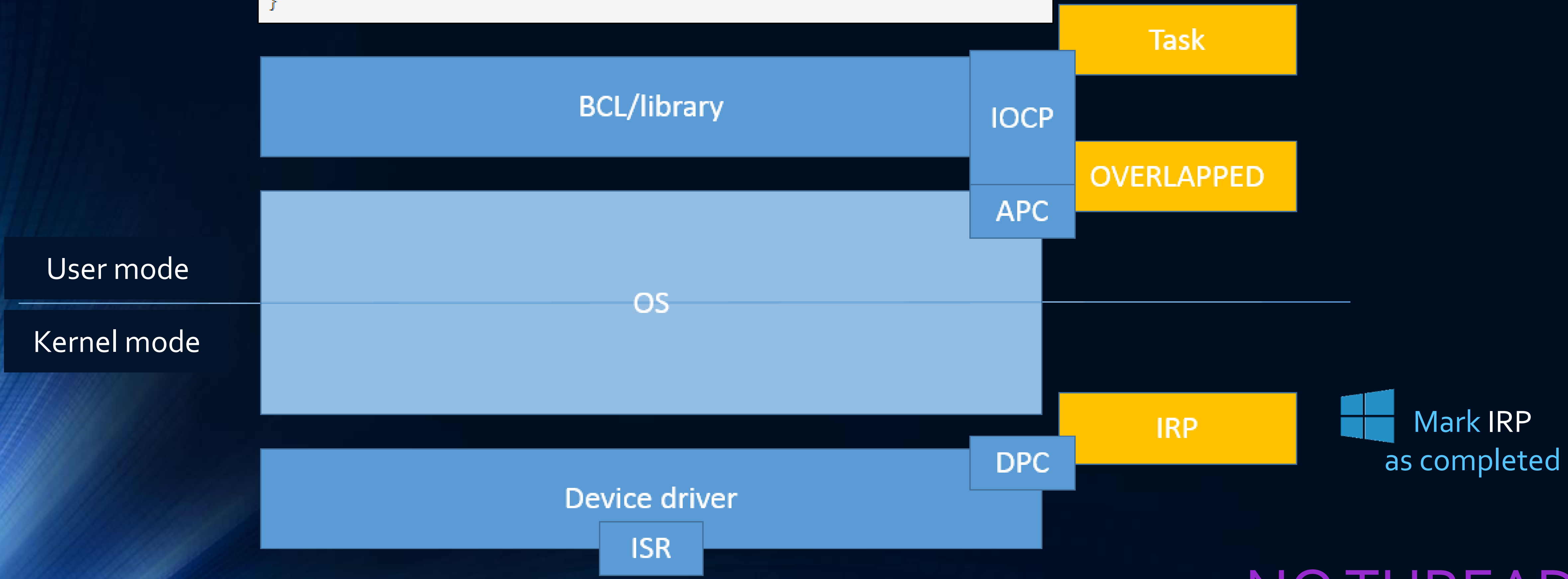


Queue Deferred Procedure Call (DPC).

NO THREAD

WINDOWS I/O: UNDER THE HOOD

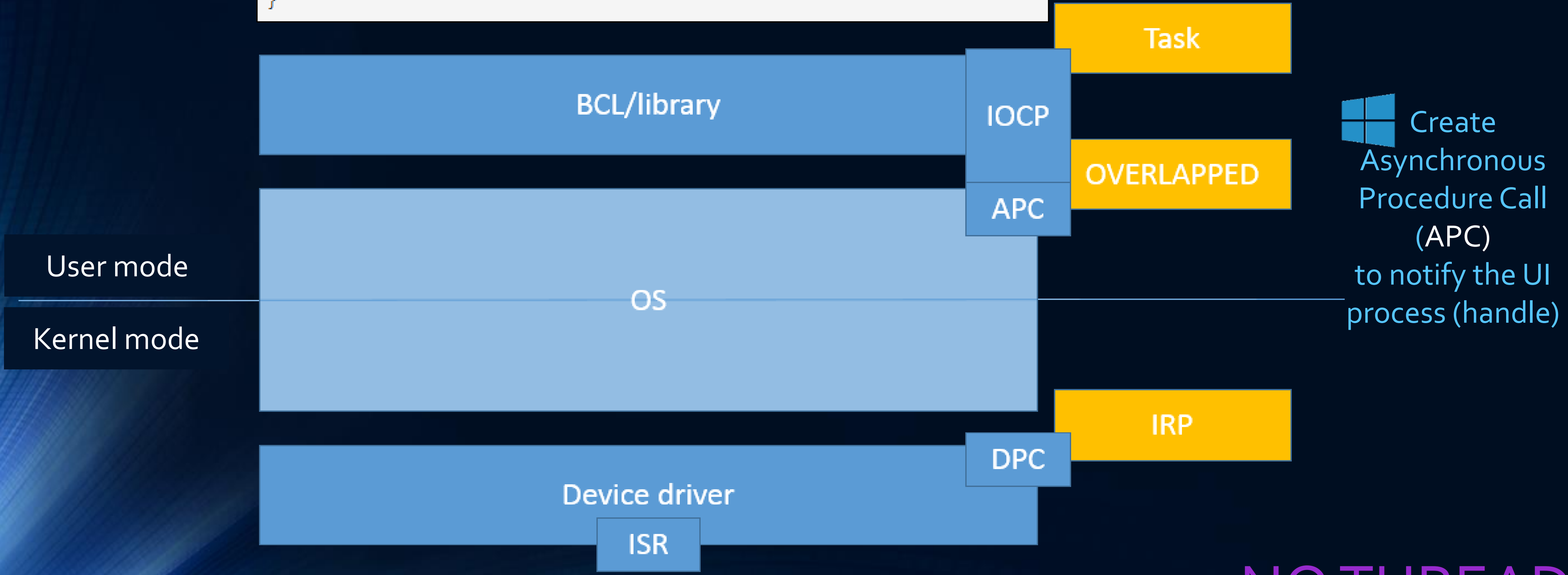
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



NO THREAD

WINDOWS I/O: UNDER THE HOOD

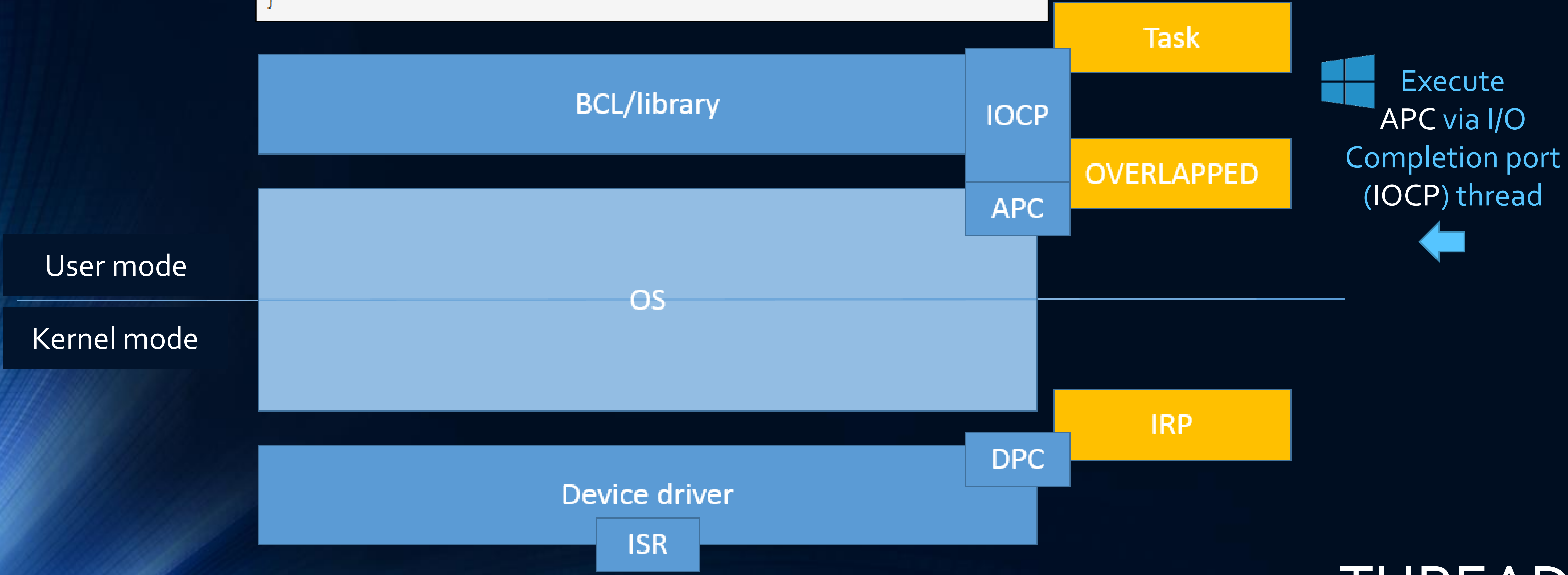
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



NO THREAD

WINDOWS I/O: UNDER THE HOOD

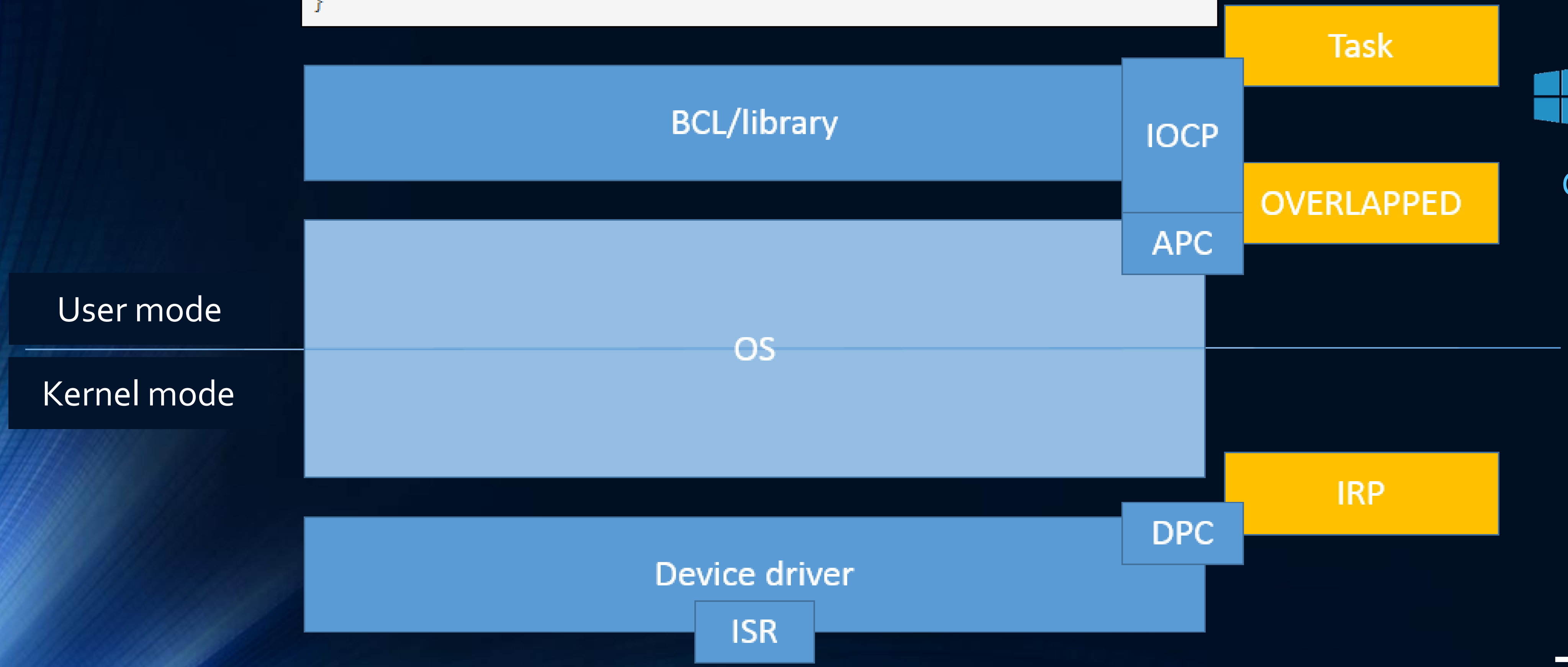
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



THREAD

WINDOWS I/O: UNDER THE HOOD

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```

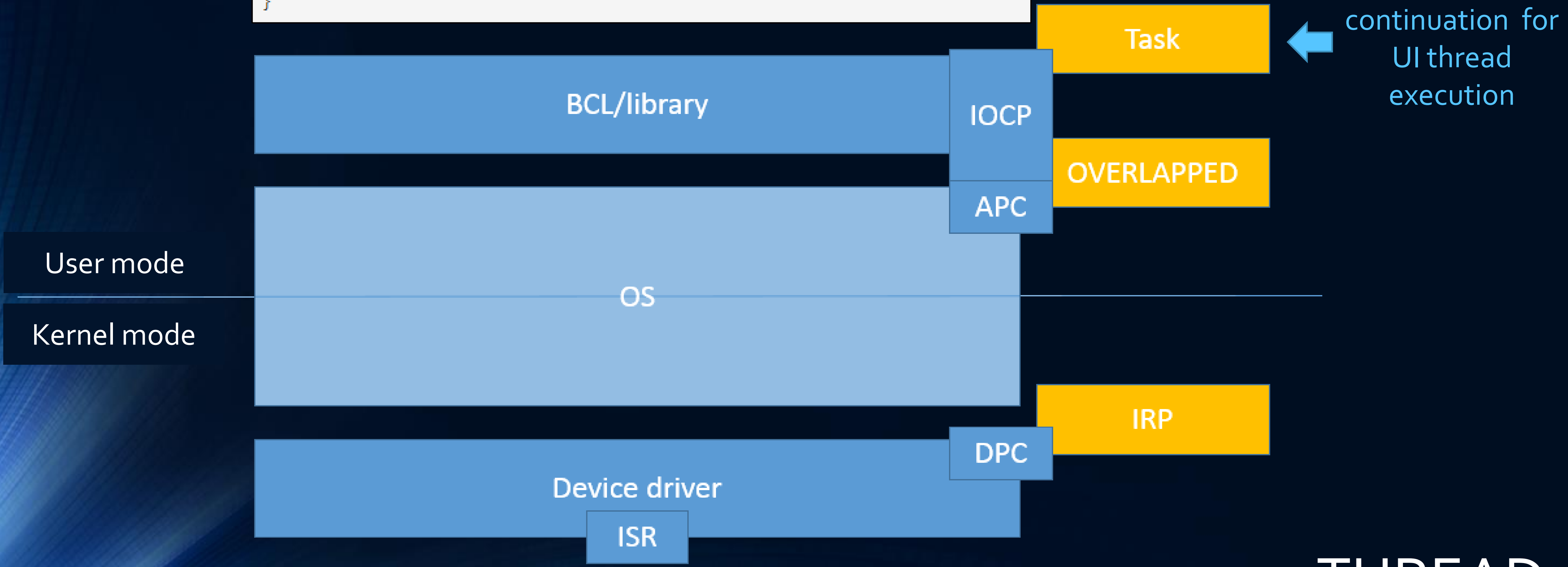


Execute APC via I/O Completion port (IOCP) thread

THREAD

WINDOWS I/O: UNDER THE HOOD

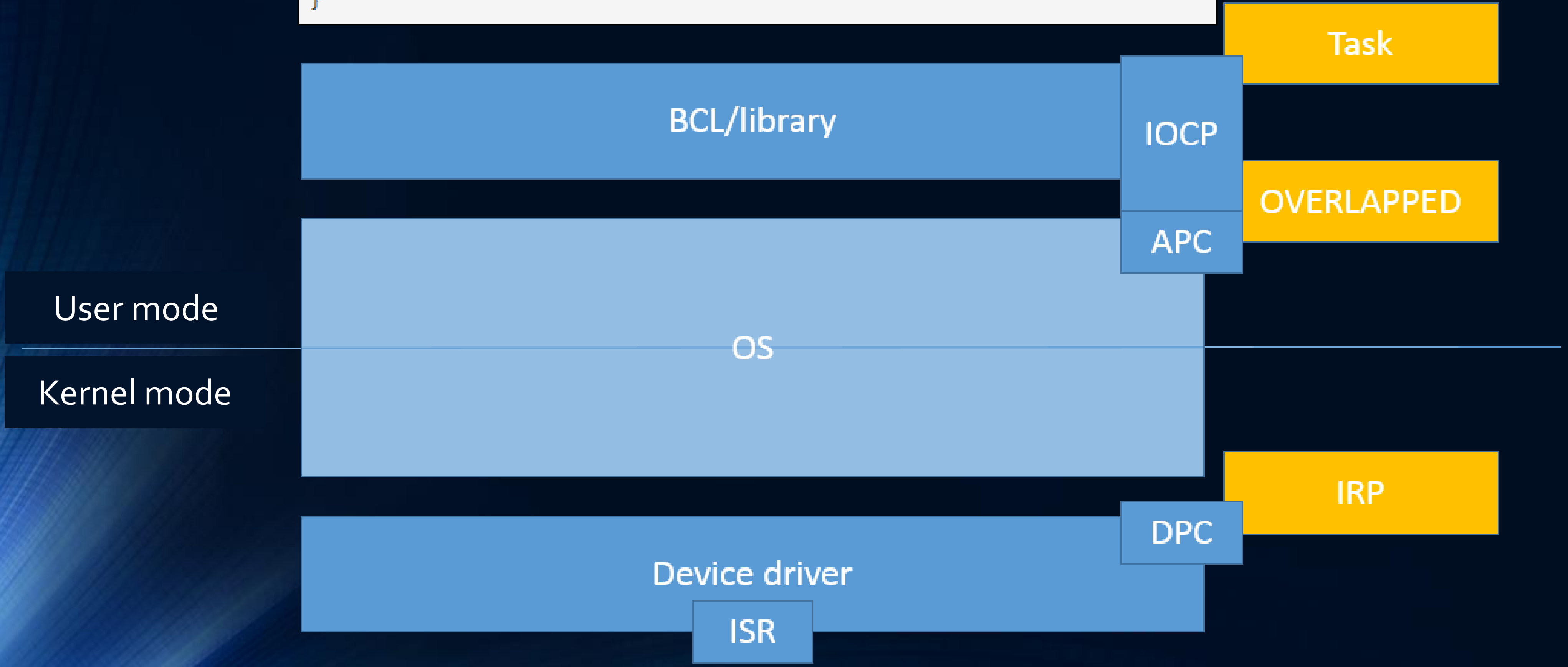
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



THREAD

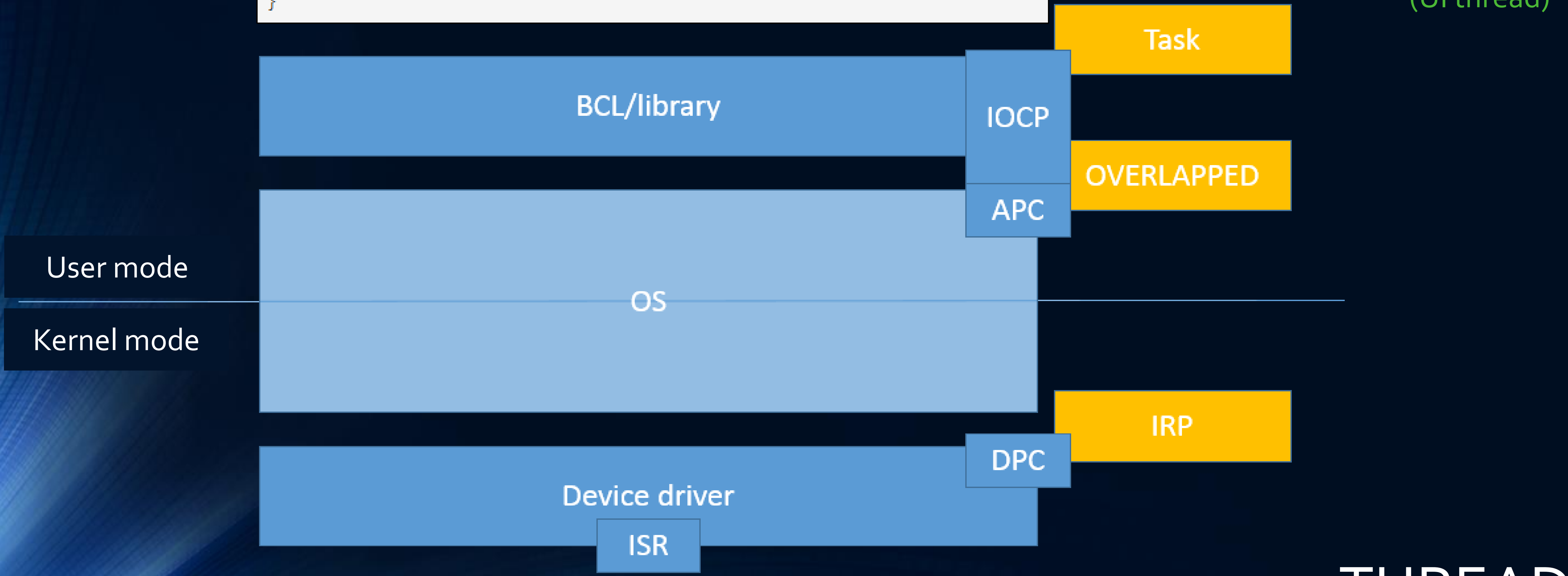
WINDOWS I/O: UNDER THE HOOD

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



WINDOWS I/O: UNDER THE HOOD

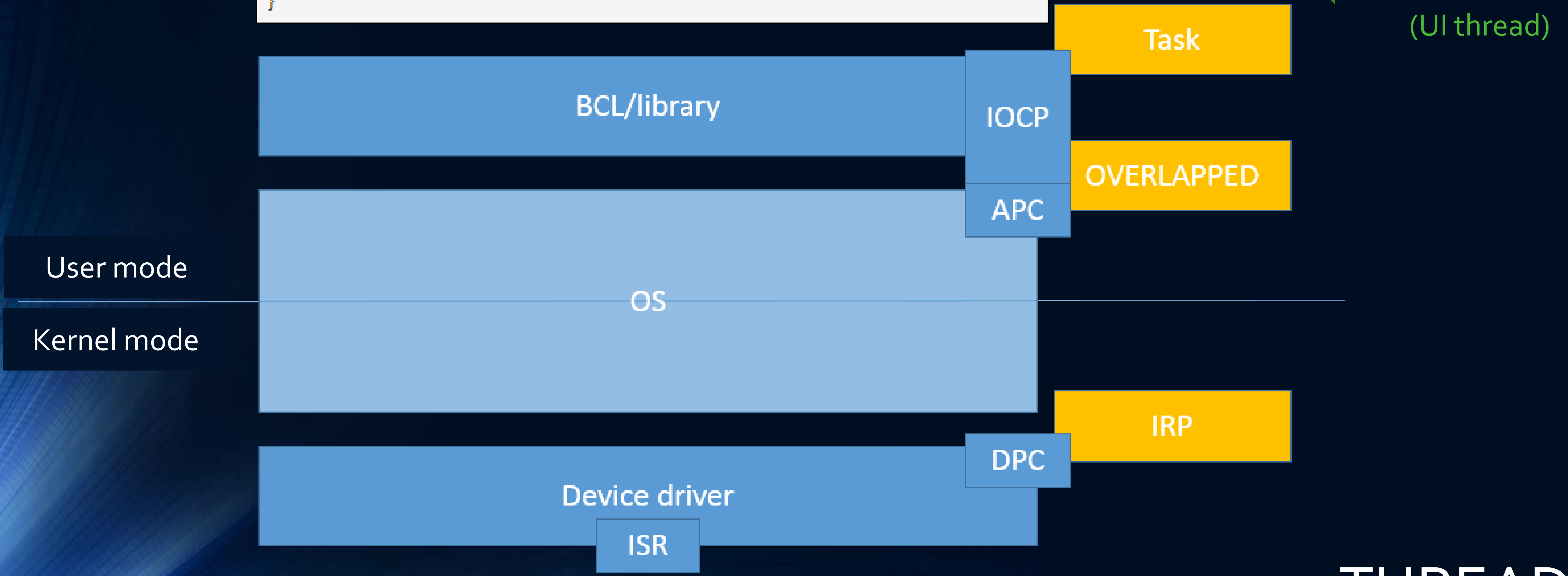
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



THREAD

WINDOWS I/O: UNDER THE HOOD

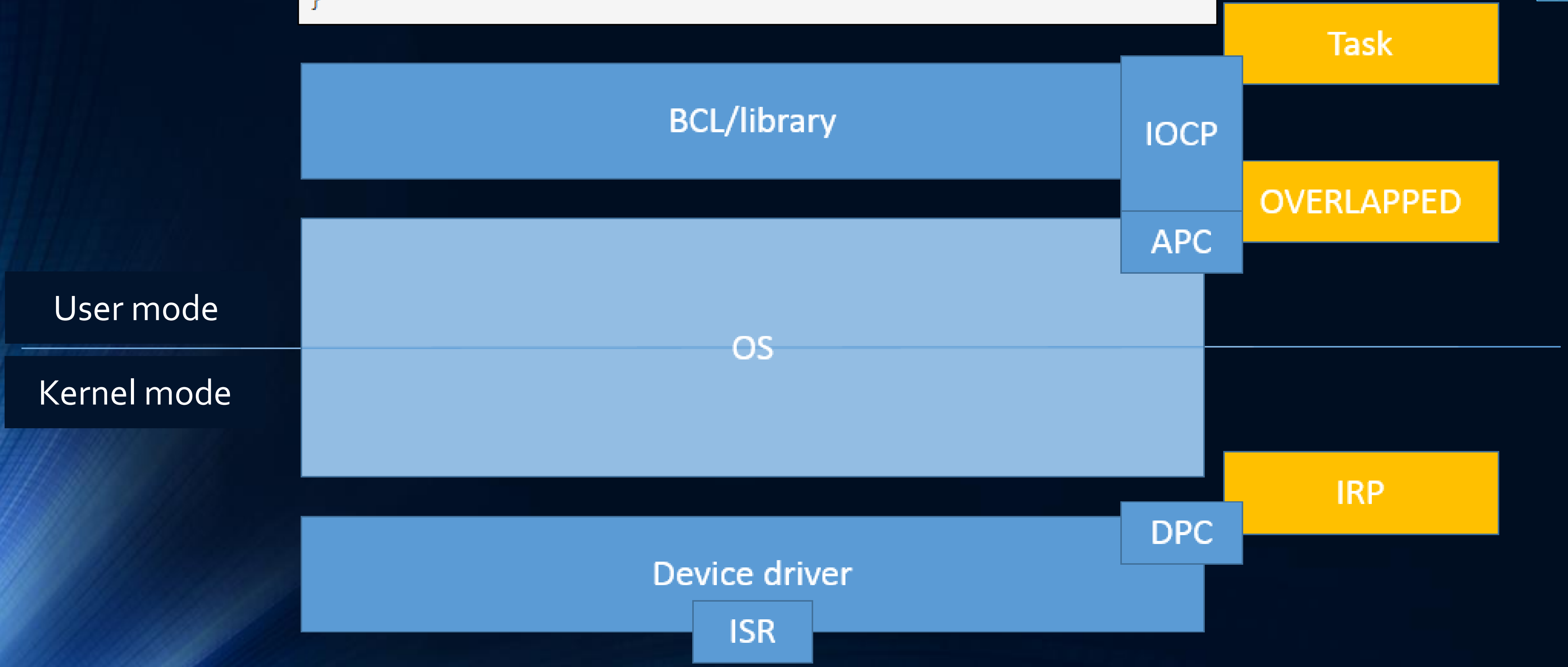
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



THREAD

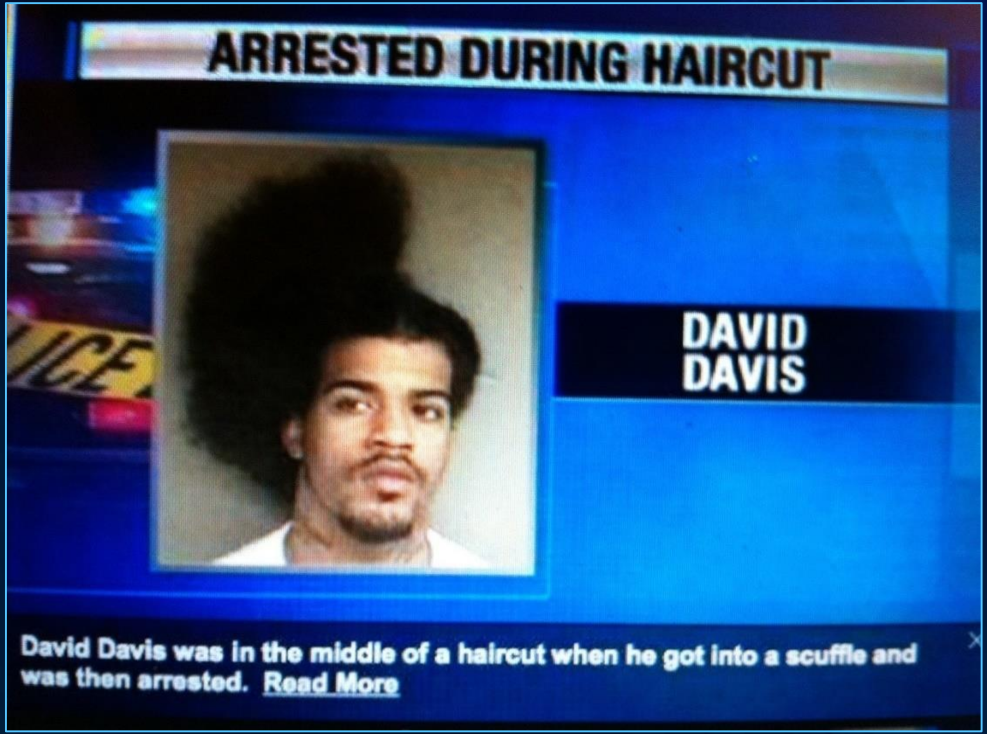
WINDOWS I/O: UNDER THE HOOD

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
  byte[] data = ...
  await myDevice.WriteAsync(data, 0, data.Length);
  this.label1.Content = "Saved!";
}
```



Replay?

CHAPTER 4: PITFALLS & RECOS

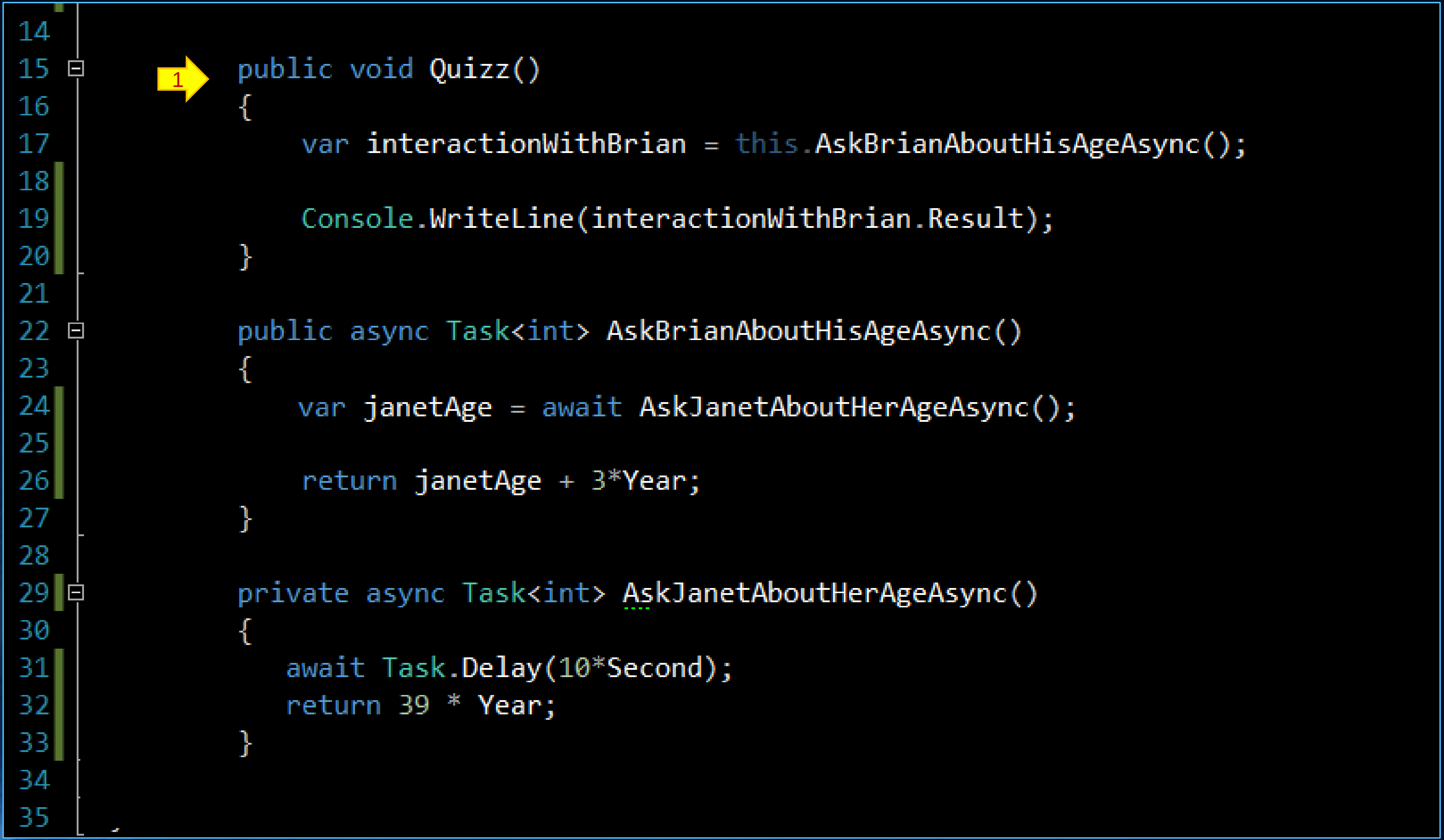

ARRESTED DURING HAIRCUT

DAVID DAVIS

David Davis was in the middle of a haircut when he got into a scuffle and was then arrested. Read More

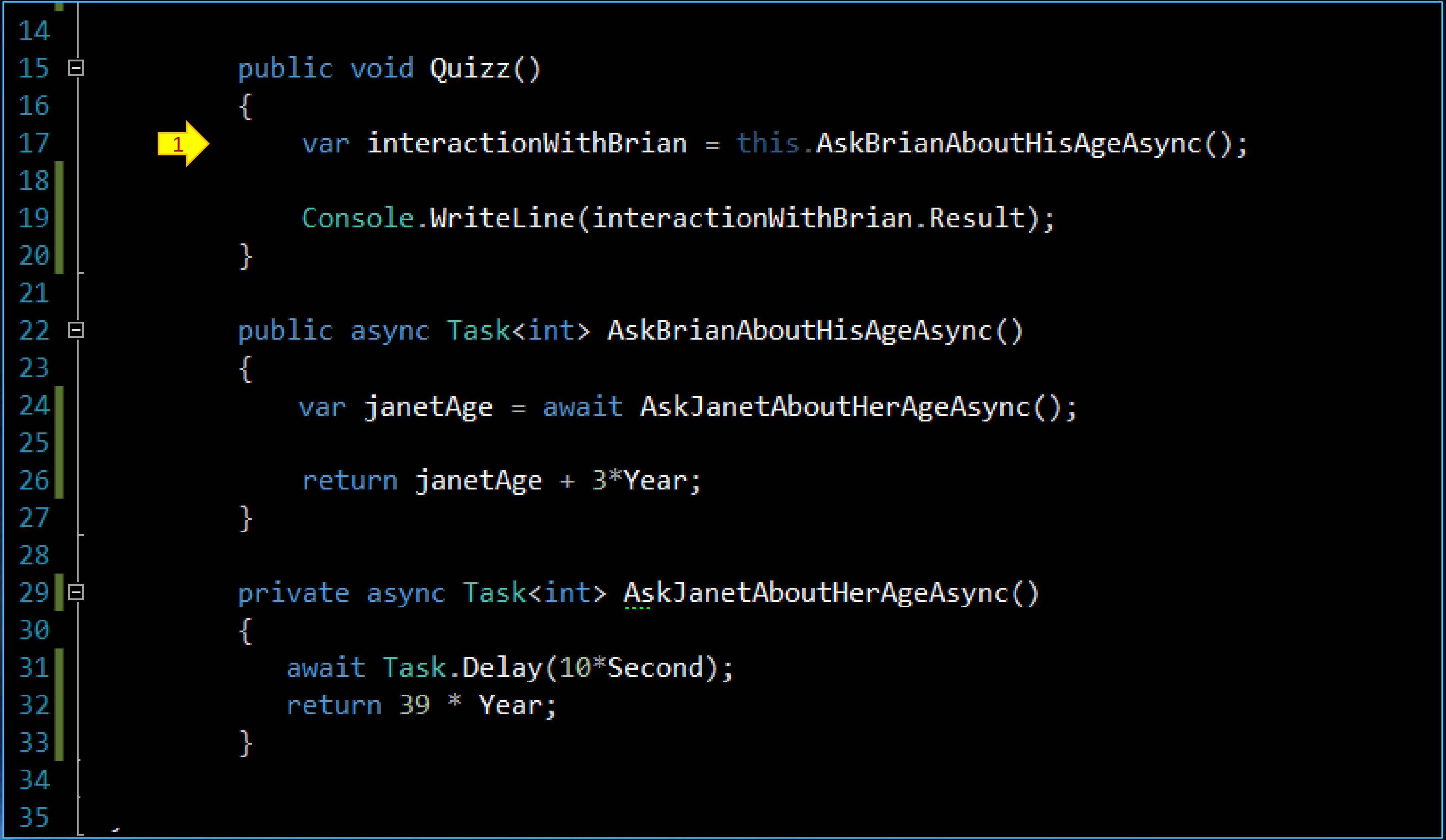
#1: DEADLOCK

INITIAL QUESTION, BUT IN A GUI CONTEXT



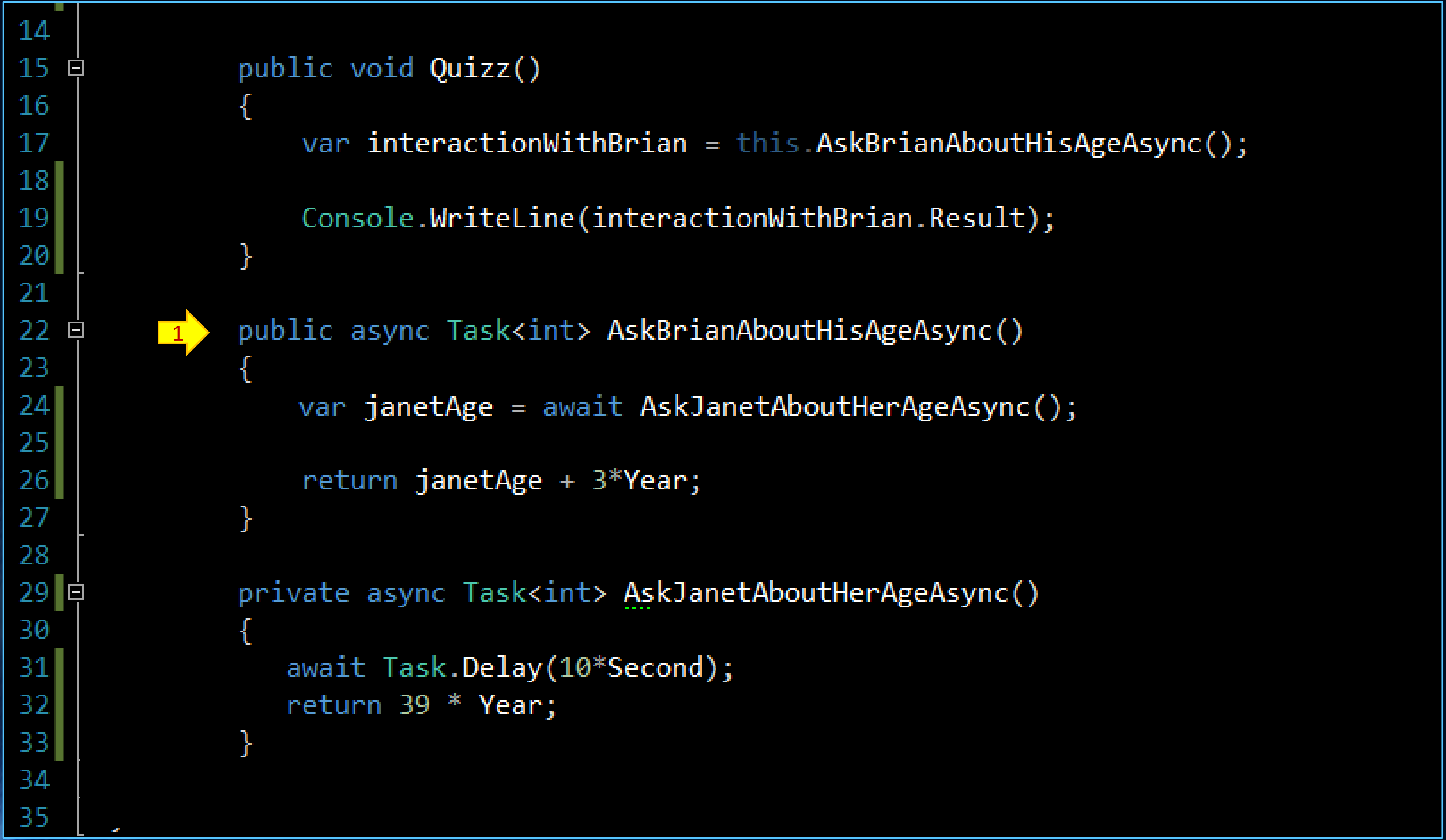
```
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



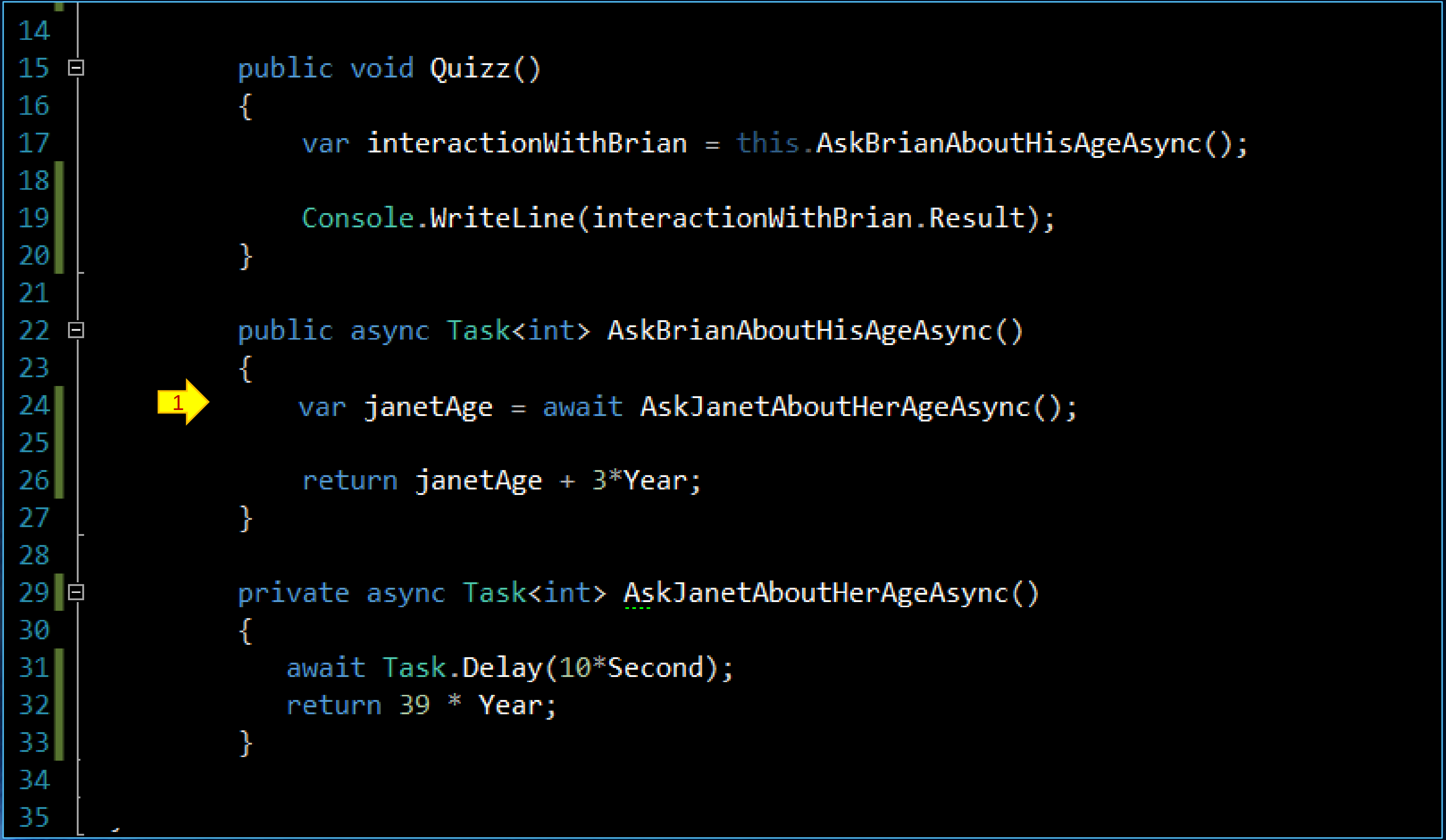
```
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



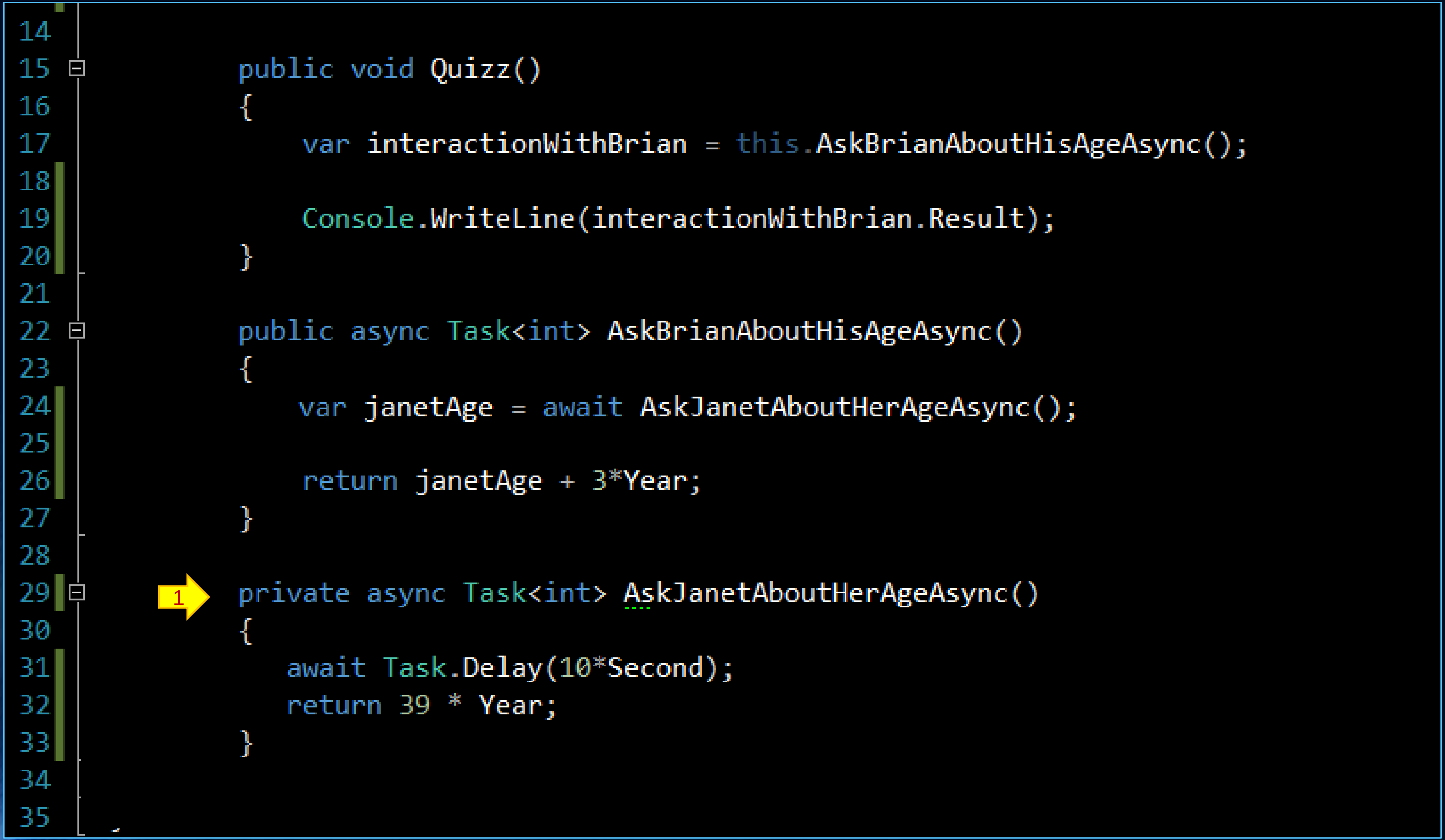
```
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



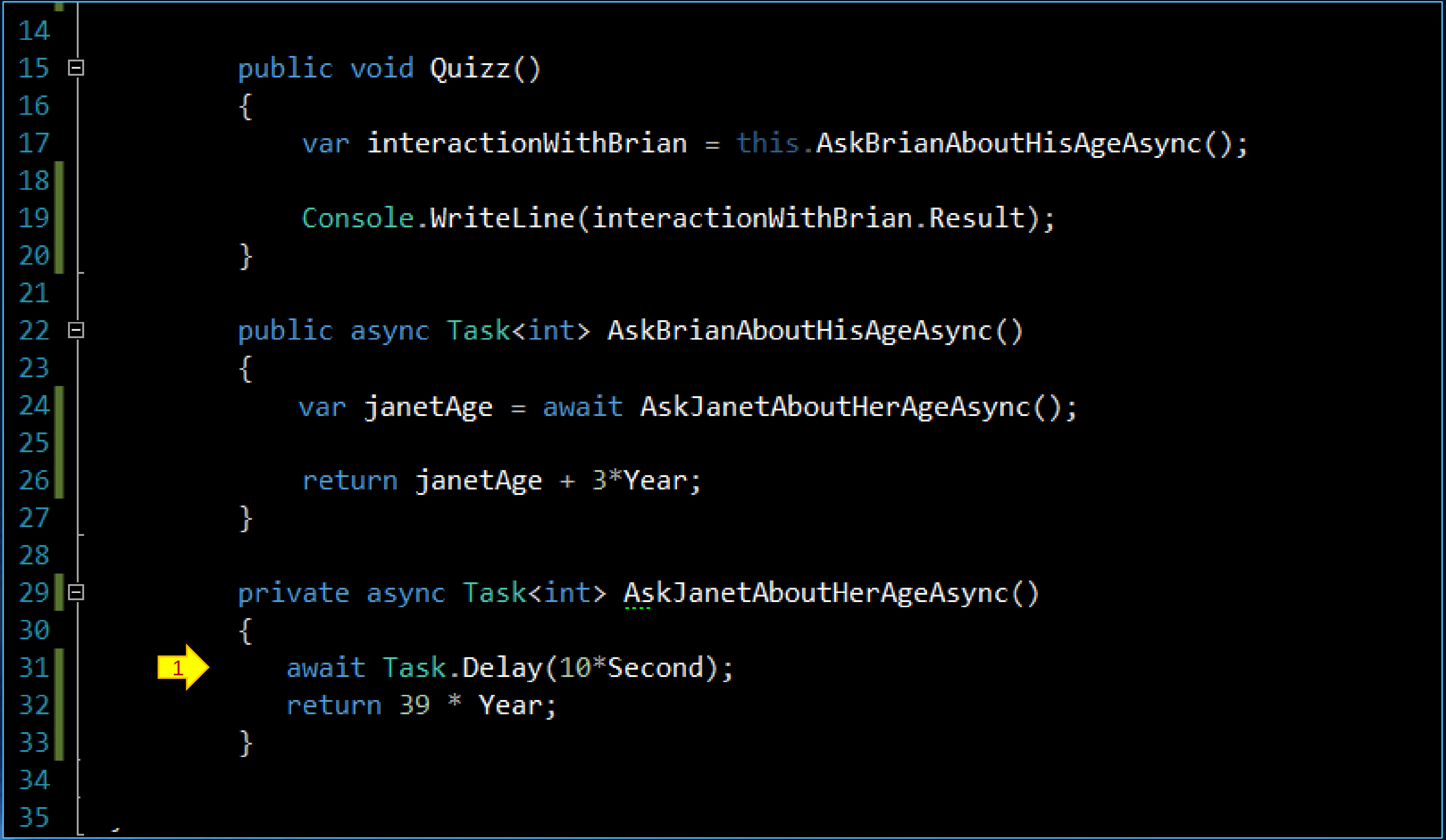
```
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



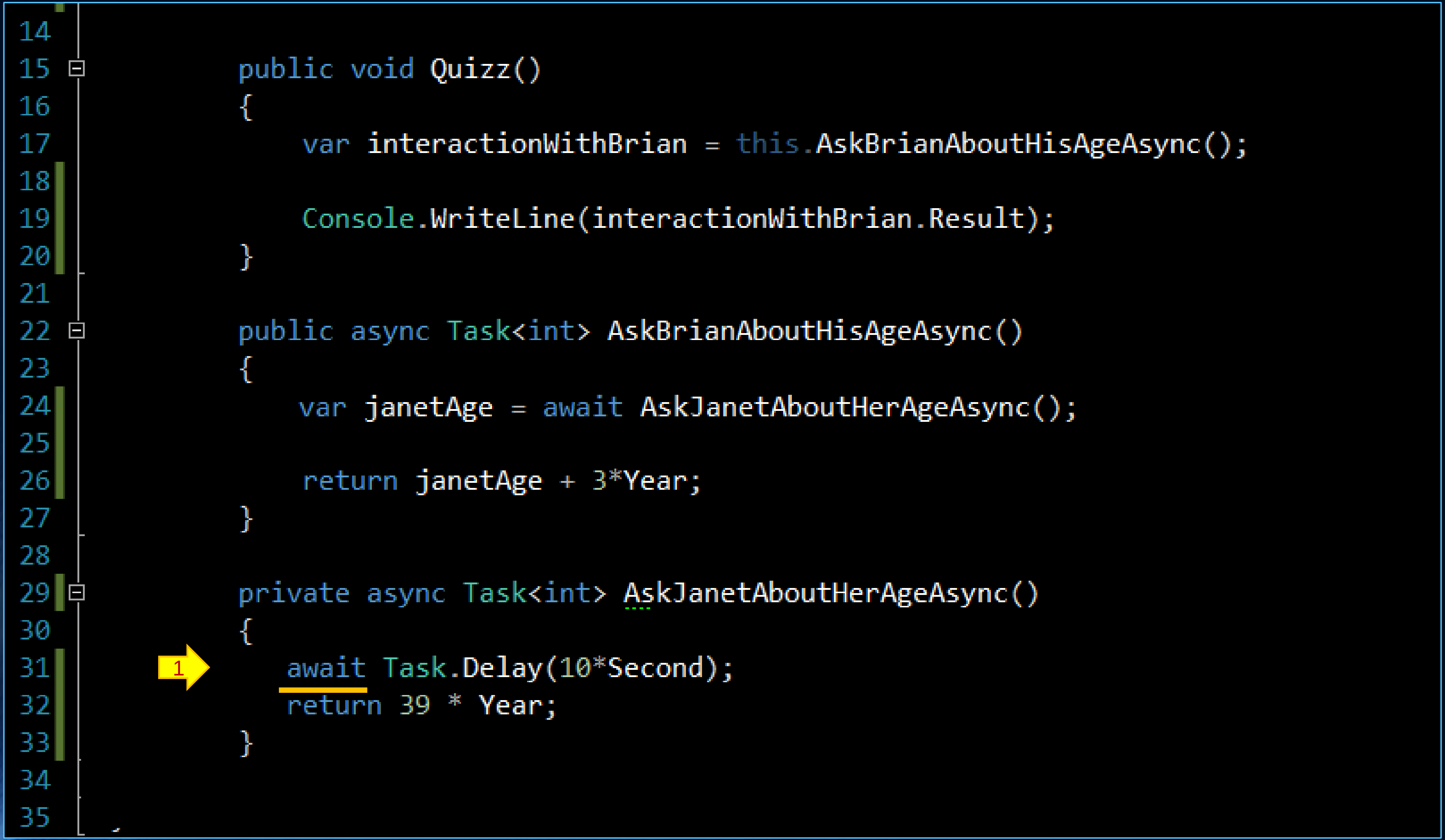
```
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



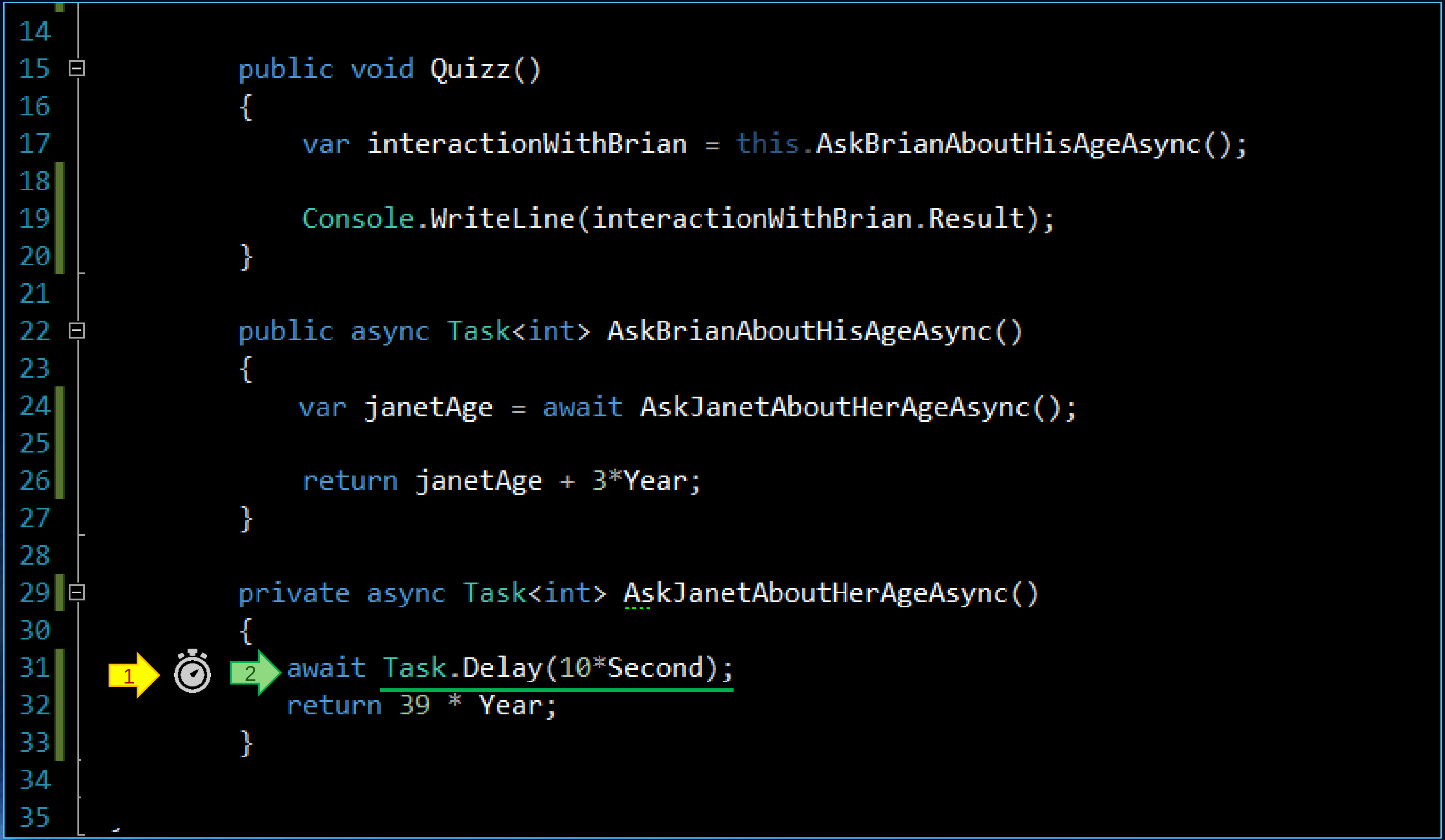
```csharp
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



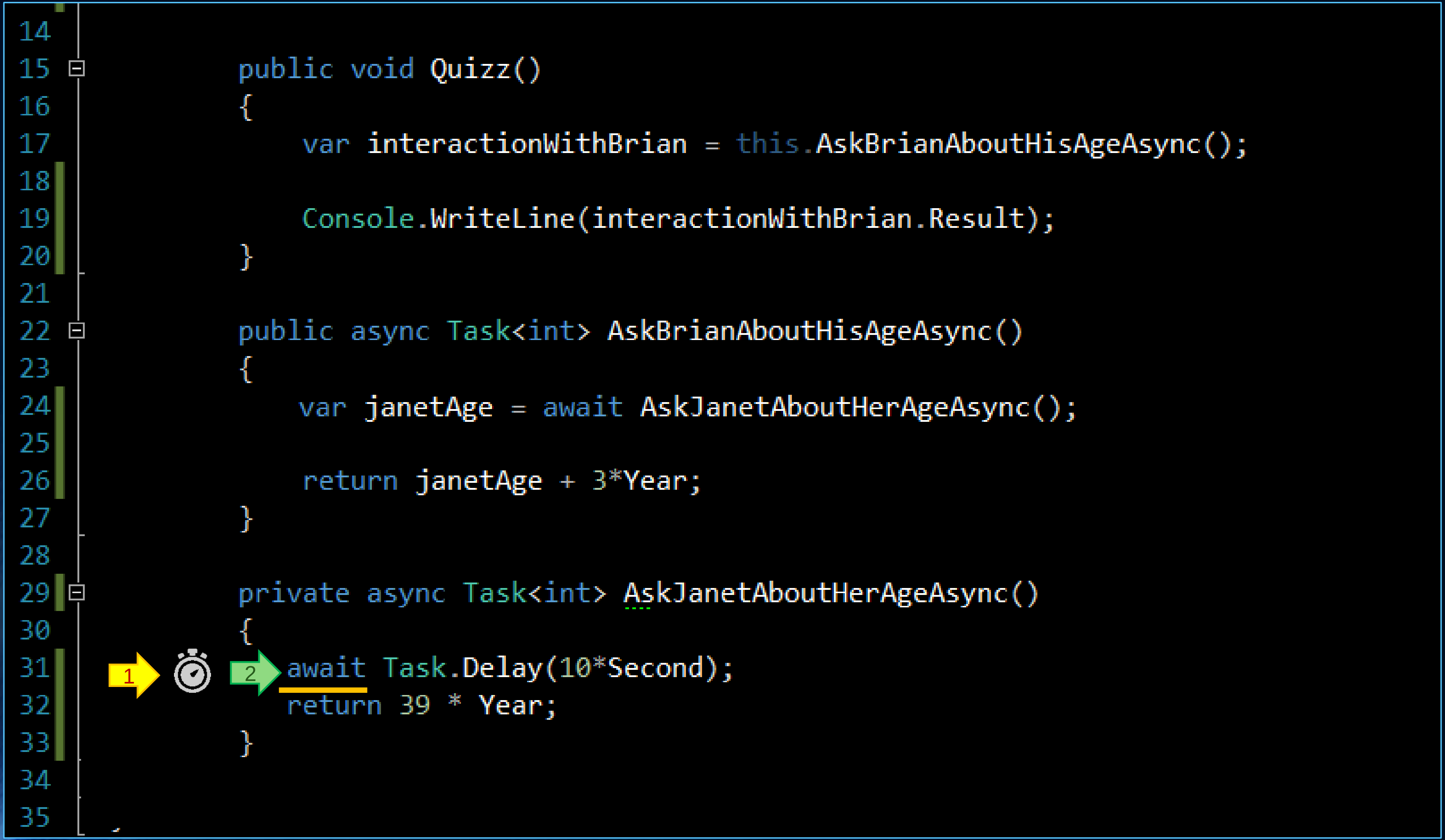
```
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



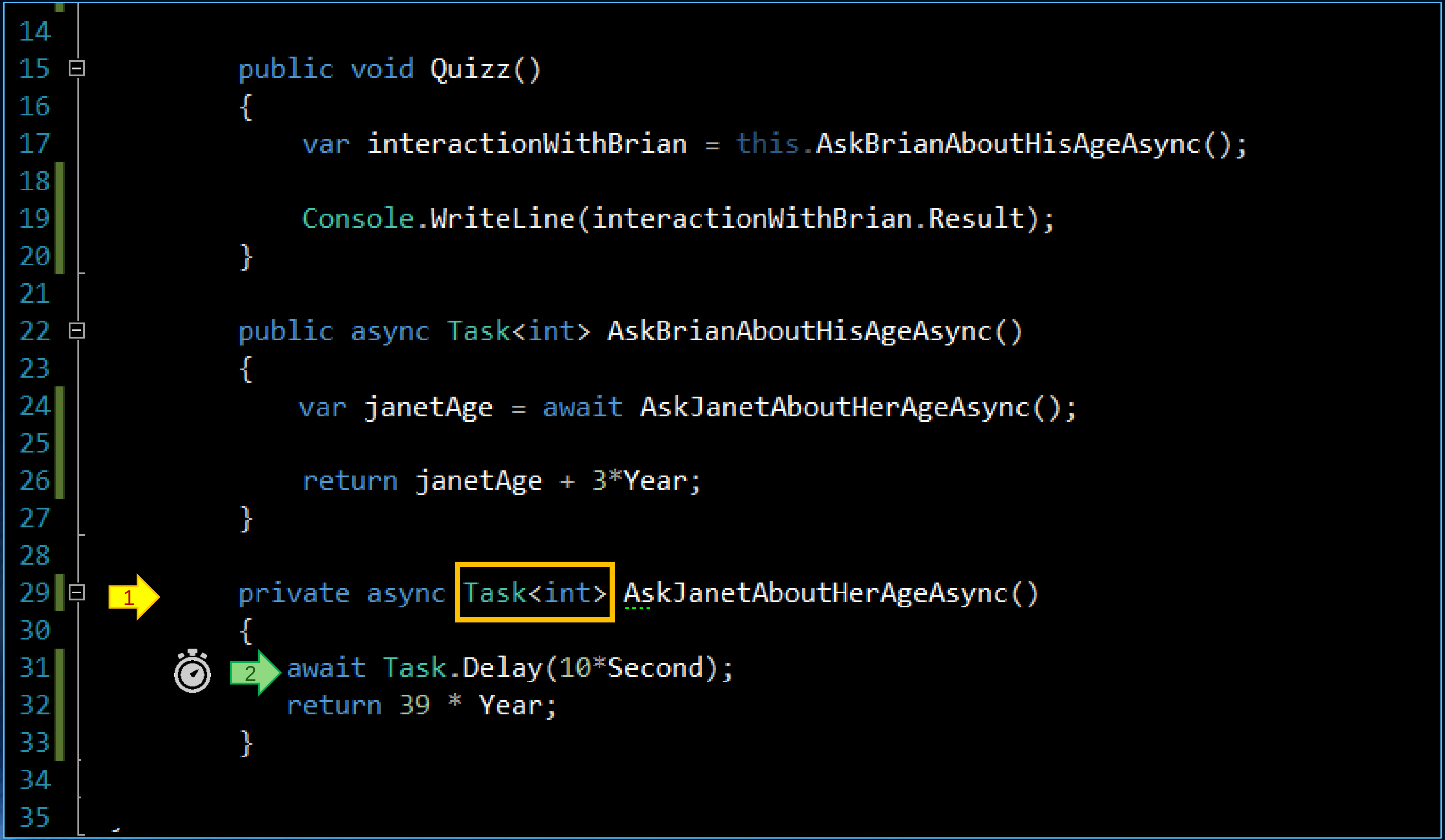
```
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



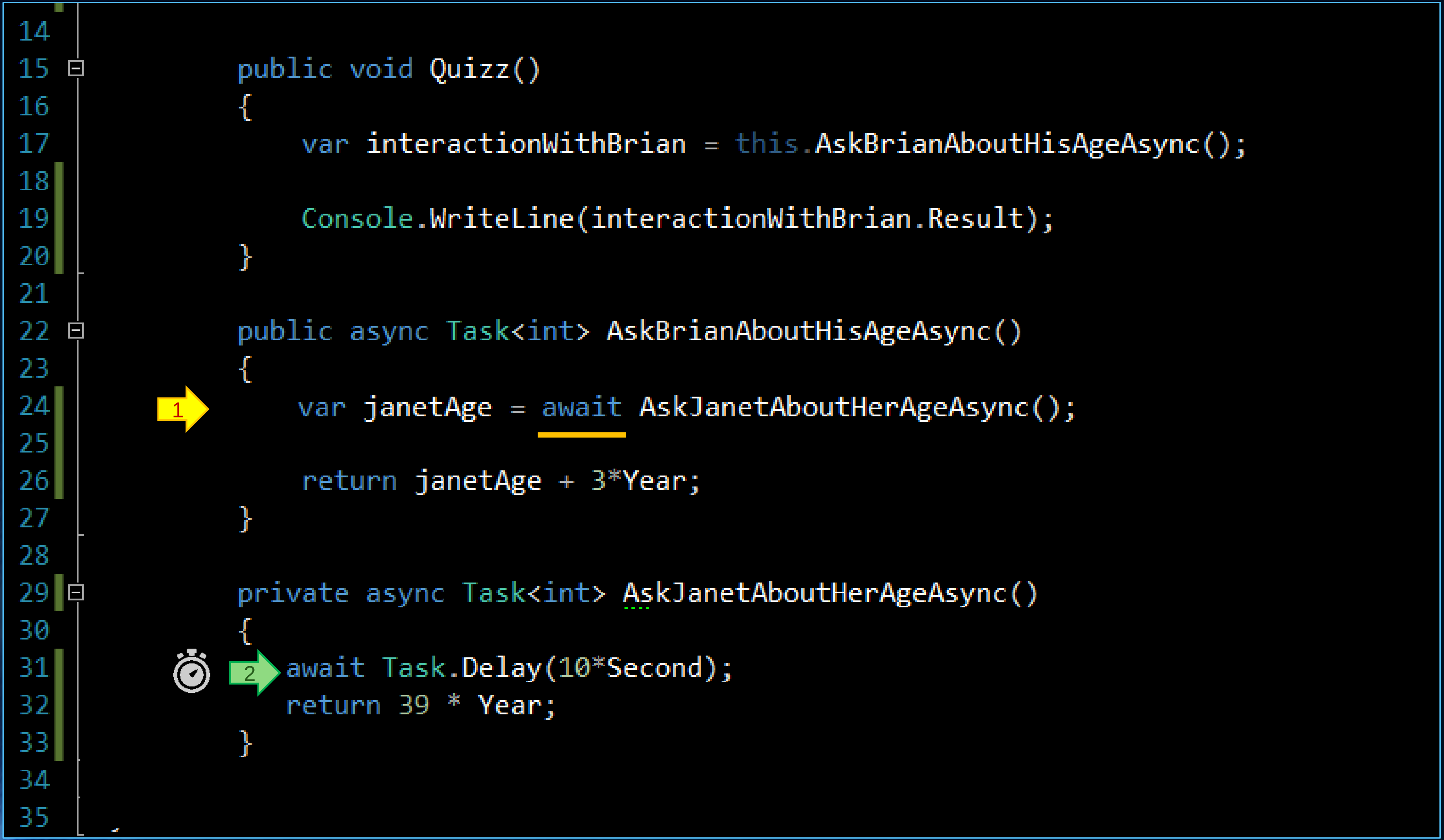
```
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



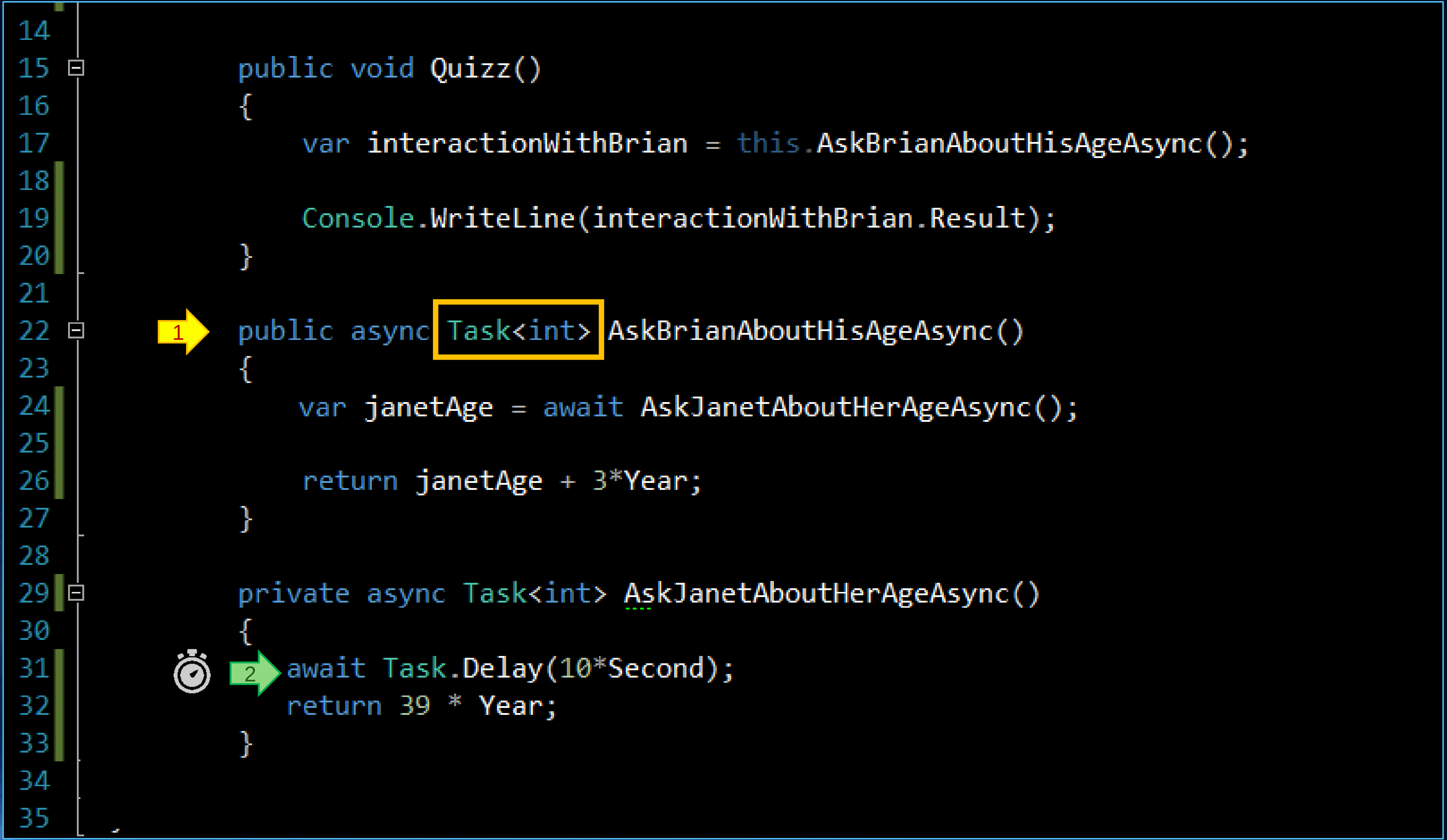
```
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



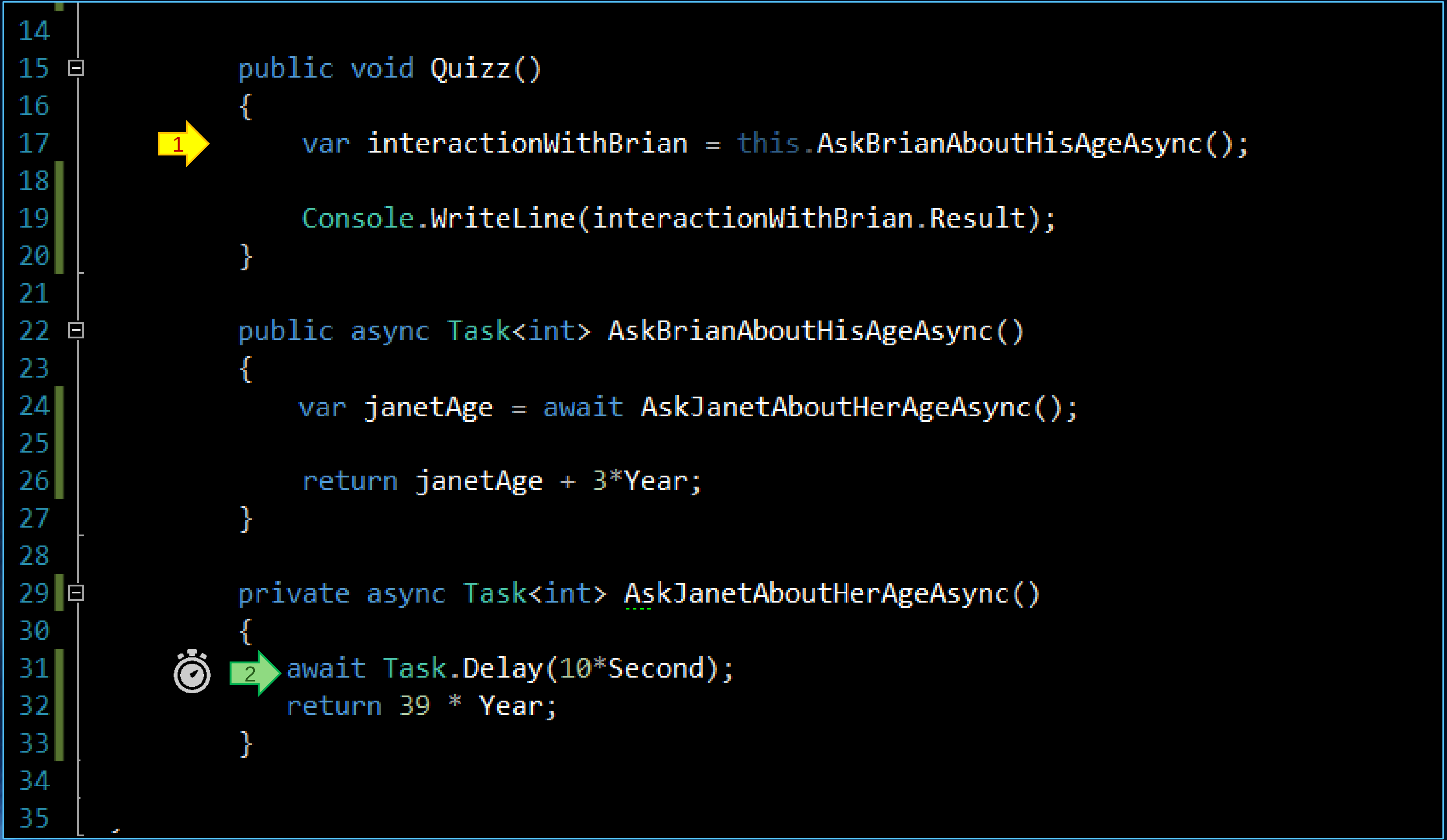
```
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



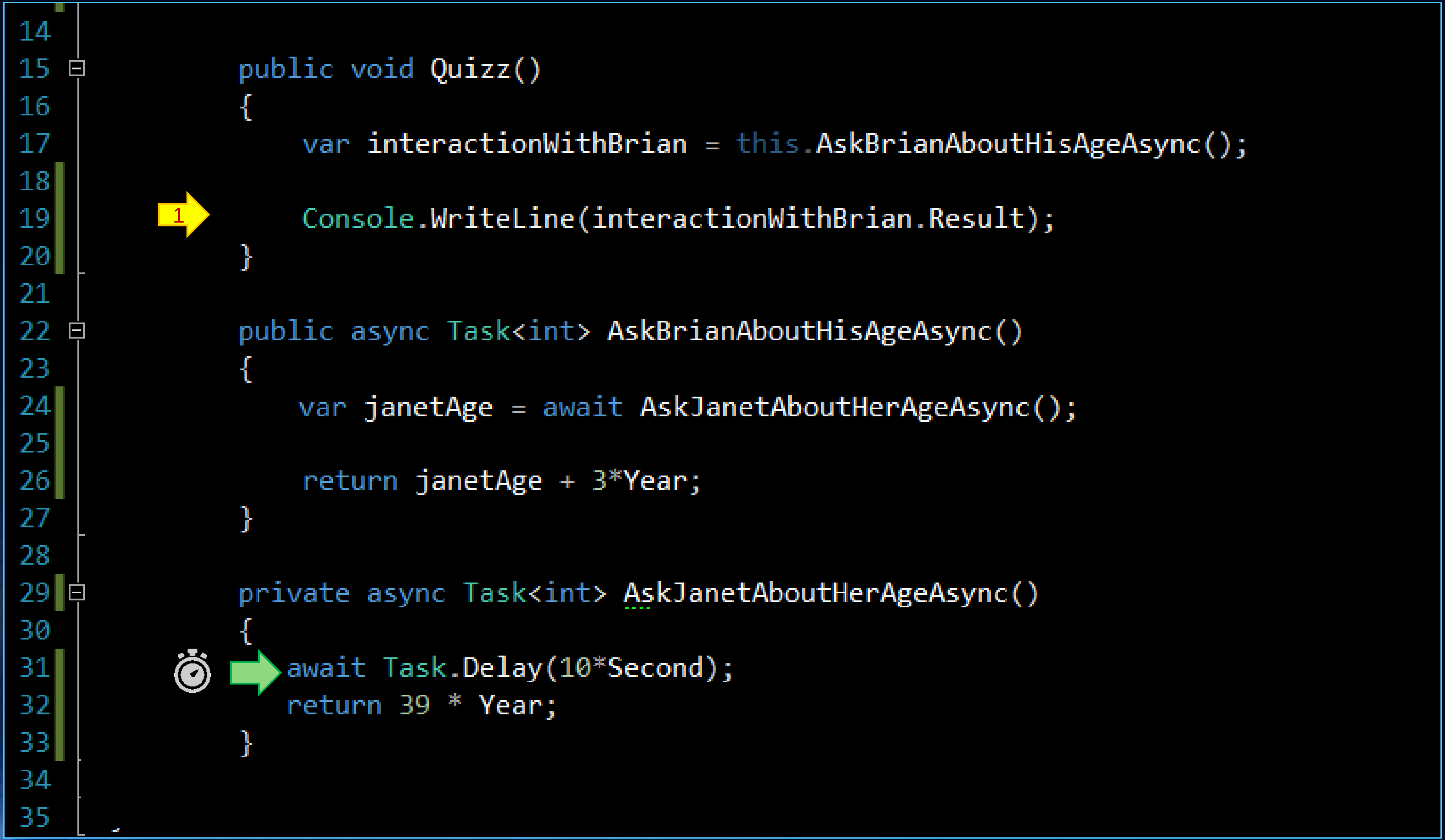
```
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private async Task<int> AskJanetAboutHerAgeAsync()
        {
           await Task.Delay(10*Second);
           return 39 * Year;
        }
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



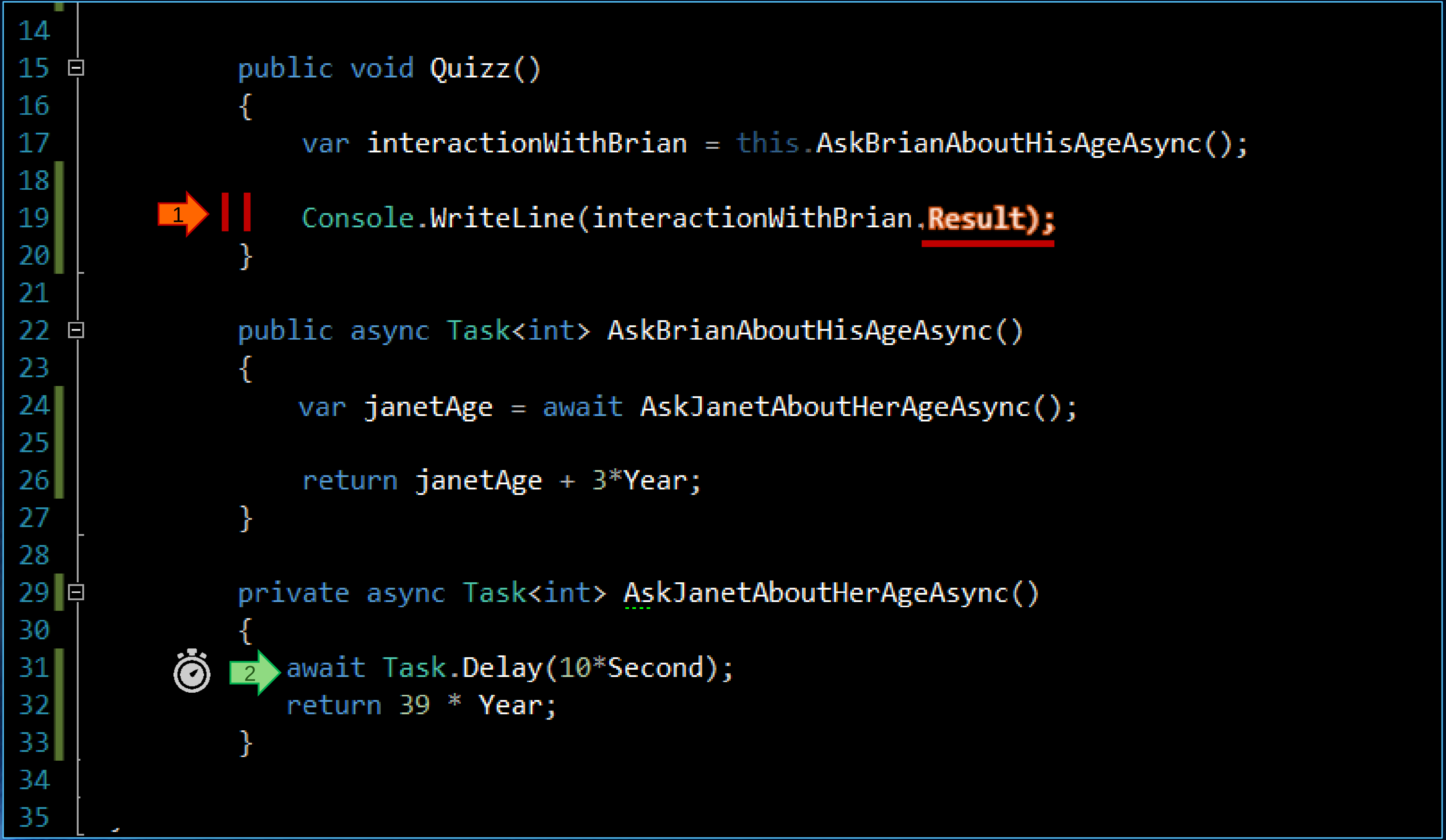
```
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



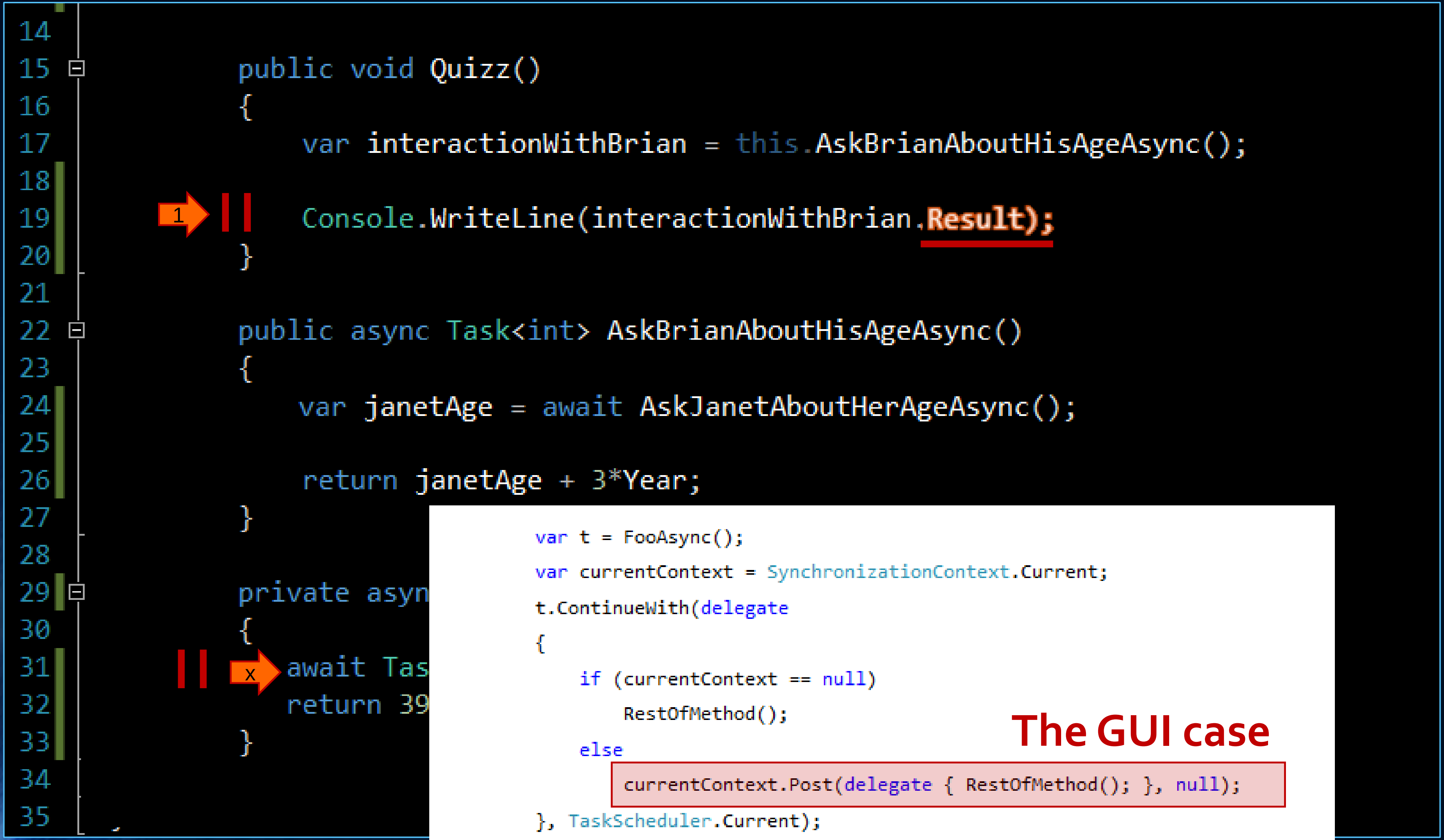
```
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

1   Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second);
   return 39 * Year;
}
```

INITIAL QUESTION, BUT IN A GUI CONTEXT

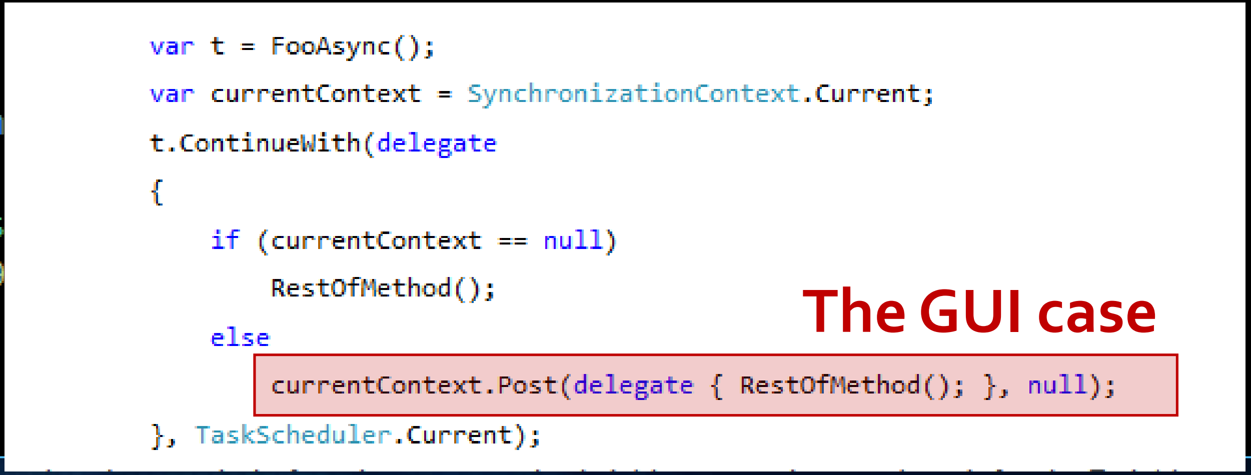


```
    public void Quizz()
    {
        var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

        Console.WriteLine(interactionWithBrian.Result);
    }

    public async Task<int> AskBrianAboutHisAgeAsync()
    {
        var janetAge = await AskJanetAboutHerAgeAsync();

        return janetAge + 3*Year;
    }

    private async Task<int> AskJanetAboutHerAgeAsync()
    {
      await Task.Delay(10*Second);
      return 39 * Year;
    }
```

INITIAL QUESTION, BUT IN A GUI CONTEXT



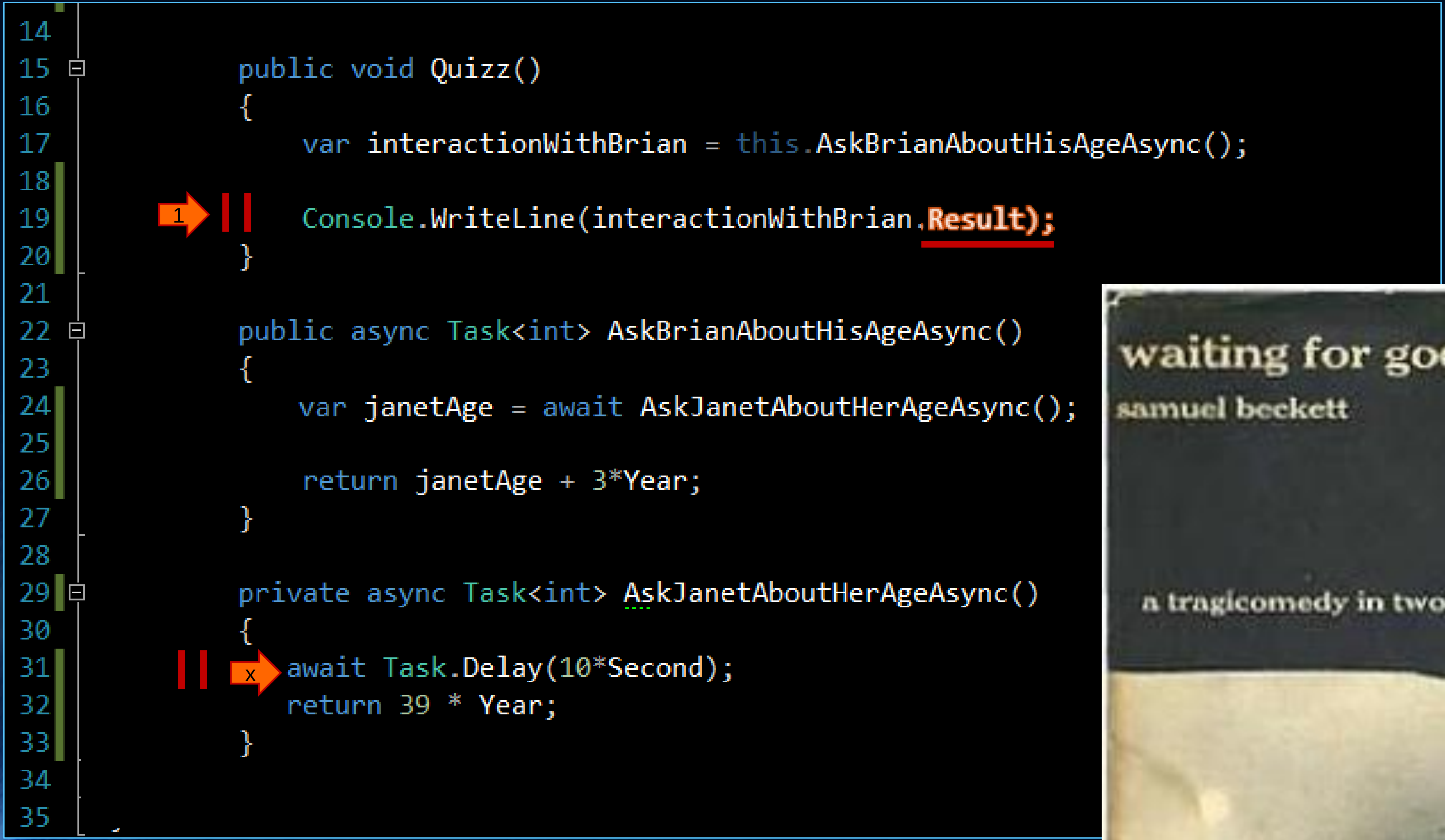
```
        public void Quizz()
        {
            var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

            Console.WriteLine(interactionWithBrian.Result);
        }

        public async Task<int> AskBrianAboutHisAgeAsync()
        {
            var janetAge = await AskJanetAboutHerAgeAsync();

            return janetAge + 3*Year;
        }

        private asyn
        {
            await Tas
            return 39
        }
```



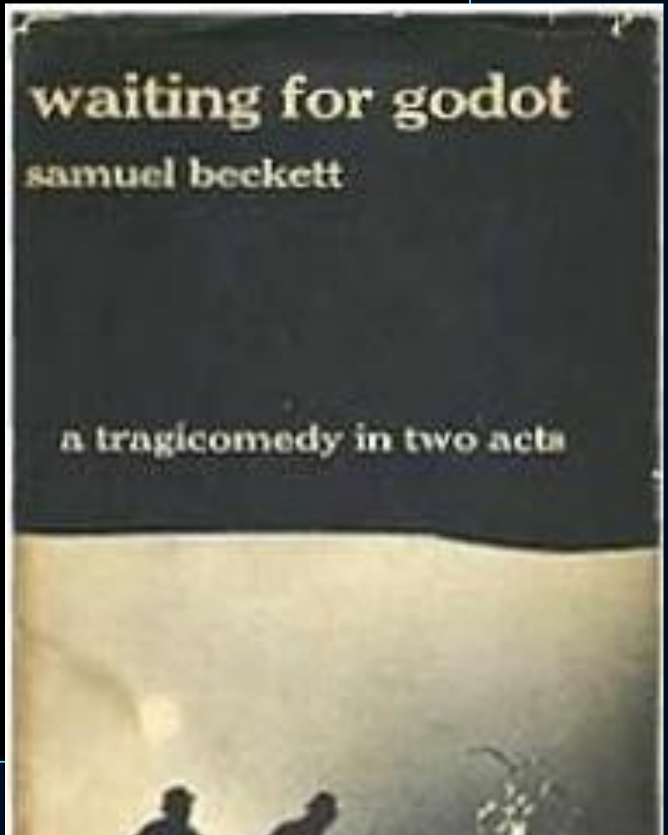
```
var t = FooAsync();
var currentContext = SynchronizationContext.Current;
t.ContinueWith(delegate
{
    if (currentContext == null)
        RestOfMethod();
    else
        currentContext.Post(delegate { RestOfMethod(); }, null);
}, TaskScheduler.Current);
```

The GUI case

INITIAL QUESTION, BUT IN A GUI CONTEXT

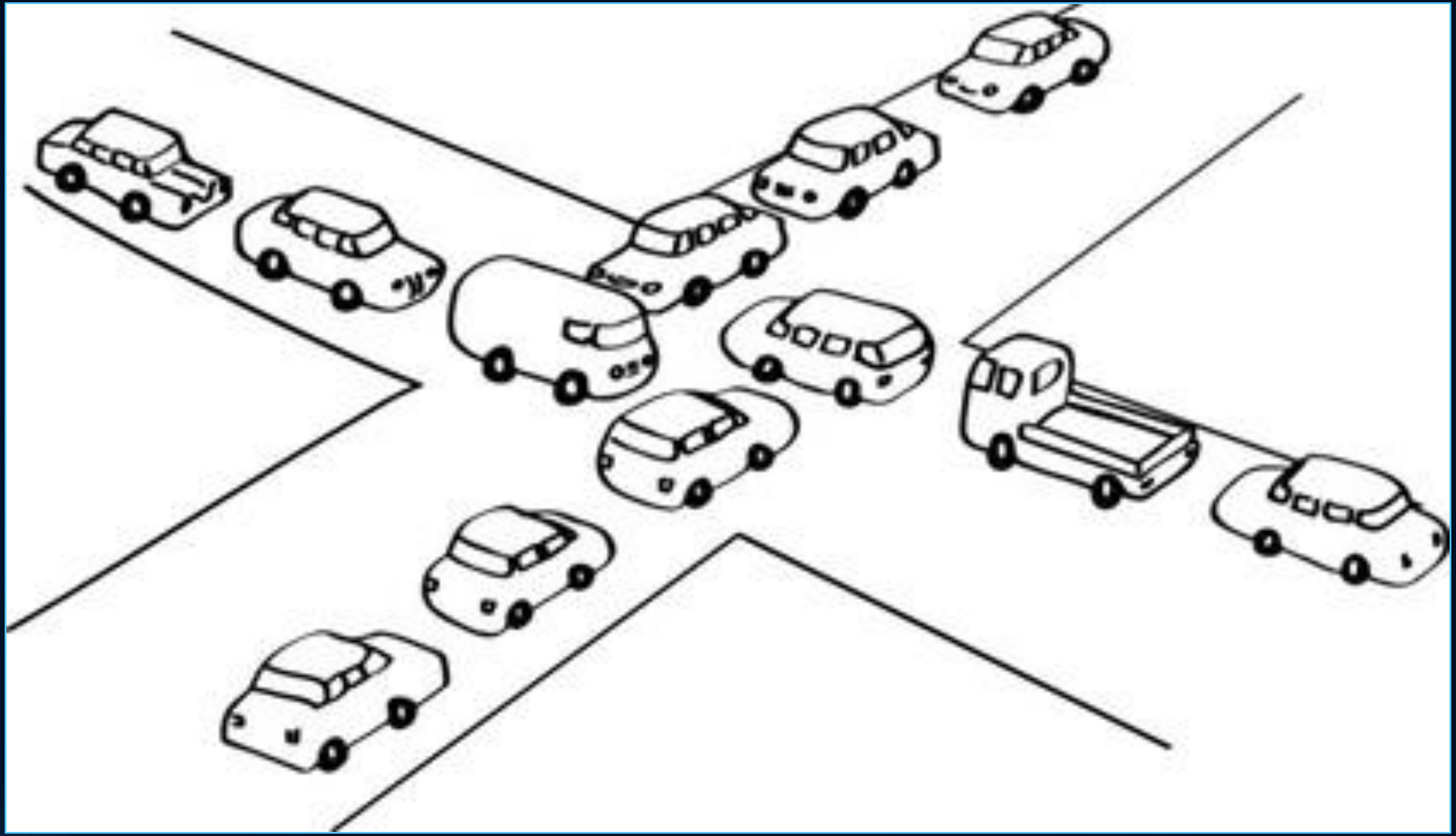


```
public void Quizz()
{
    var interactionWithBrian = this.AskBrianAboutHisAgeAsync();

    Console.WriteLine(interactionWithBrian.Result);
}

public async Task<int> AskBrianAboutHisAgeAsync()
{
    var janetAge = await AskJanetAboutHerAgeAsync();

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
    await Task.Delay(10*Second);
    return 39 * Year;
}
```



„HOUSTON, WE HAVE A
PROBLEM.“

DEADLOCKS



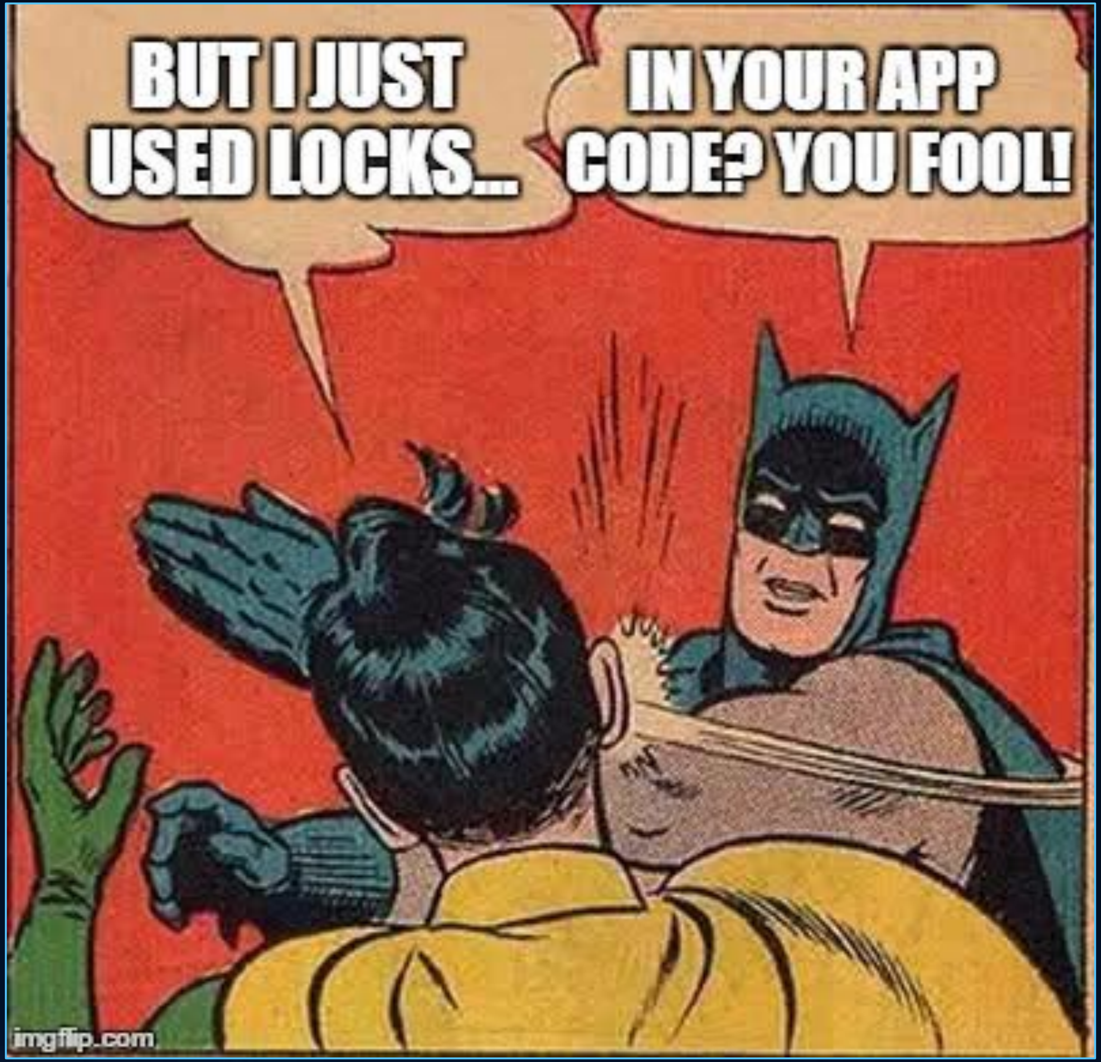
THE DUPDOB PRINCIPLE

@cyrdup

*"QUI DIT LOCKS...
DIT DEADLOCKS"*

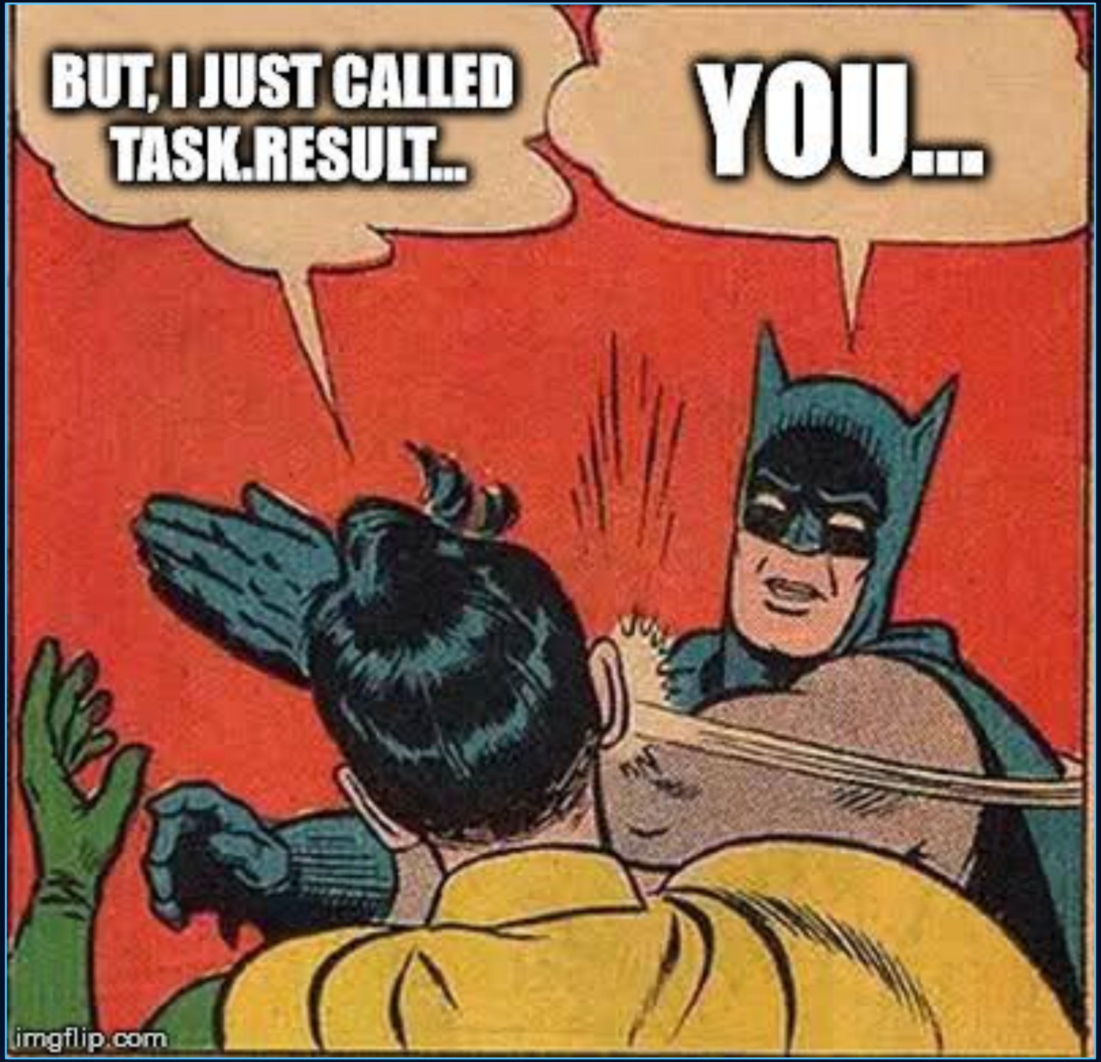
*"WHOEVER LOCKS...
EVENTUALLY DEADLOCKS"*

DEADLOCK MEME



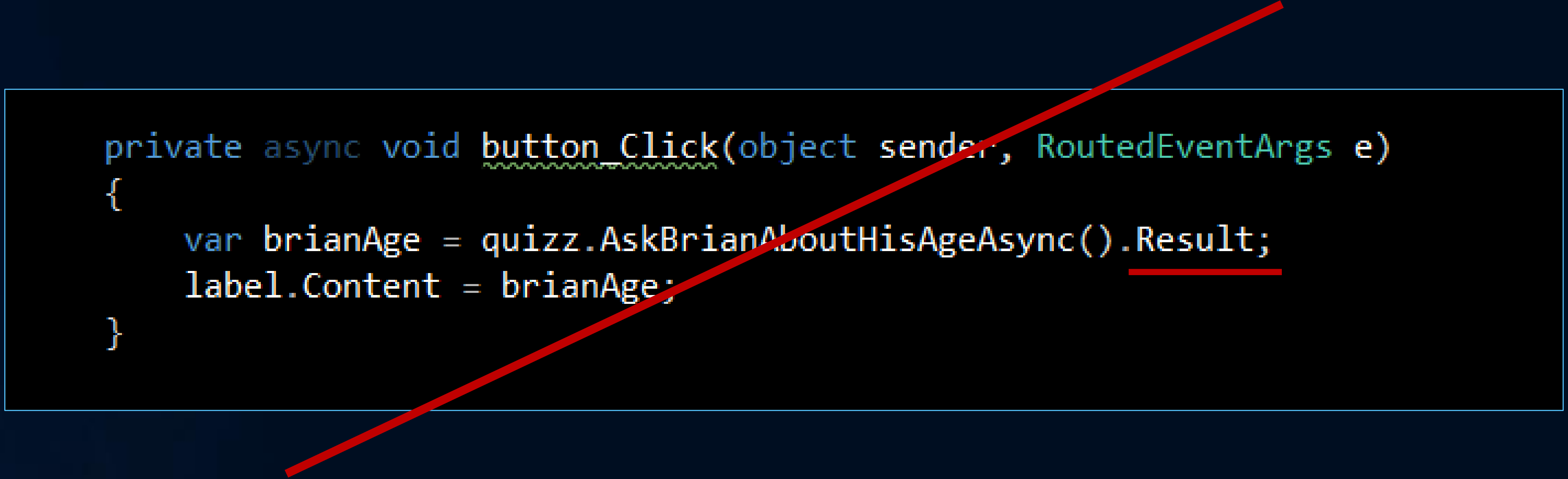
DEADLOCK MEME

DEADLOCK MEME

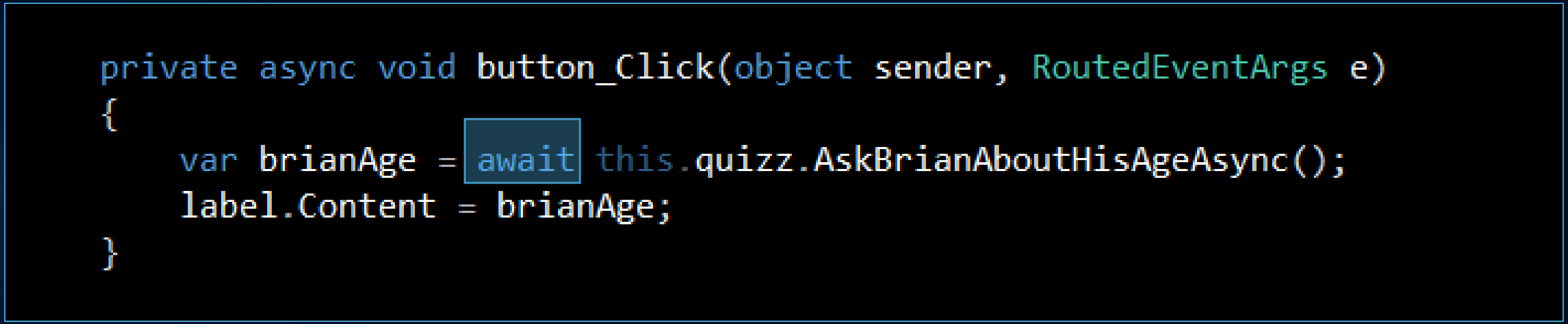


DO NOT BLOCK TO AVOID DEADLOCK

DO NOT BLOCK TO AVOID DEADLOCK



```
private async void button_Click(object sender, RoutedEventArgs e)
{
    var brianAge = quizz.AskBrianAboutHisAgeAsync().Result;
    label.Content = brianAge;
}
```



```
private async void button_Click(object sender, RoutedEventArgs e)
{
    var brianAge = await this.quizz.AskBrianAboutHisAgeAsync();
    label.Content = brianAge;
}
```

« AWAIT » DON'T BLOCK THE UI THREAD

YOU CODE A LIBRARY?

SYNCHRONIZATION CONTEXT IS NOT YOUR DECISION!

↓

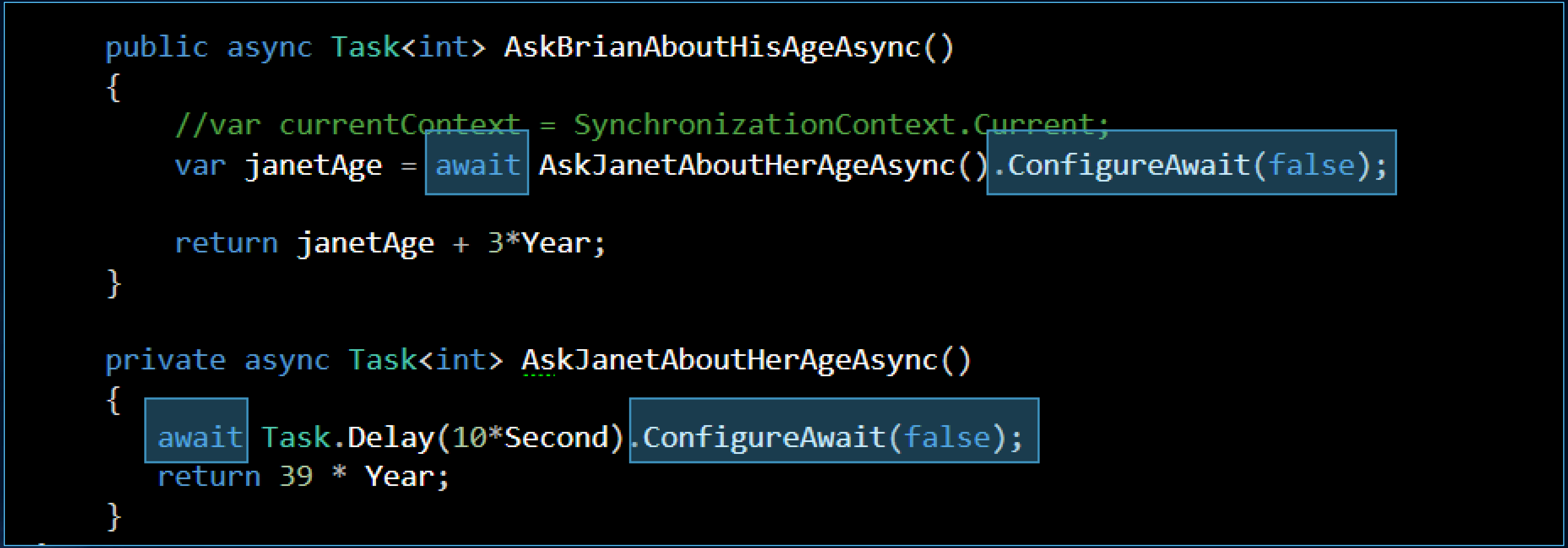
USE CONFIGUREAWAIT(FALSE) EVERYWHERE!

AVOID DEADLOCKS

IMPROVE PERFORMANCE

YOU CODE A LIBRARY?

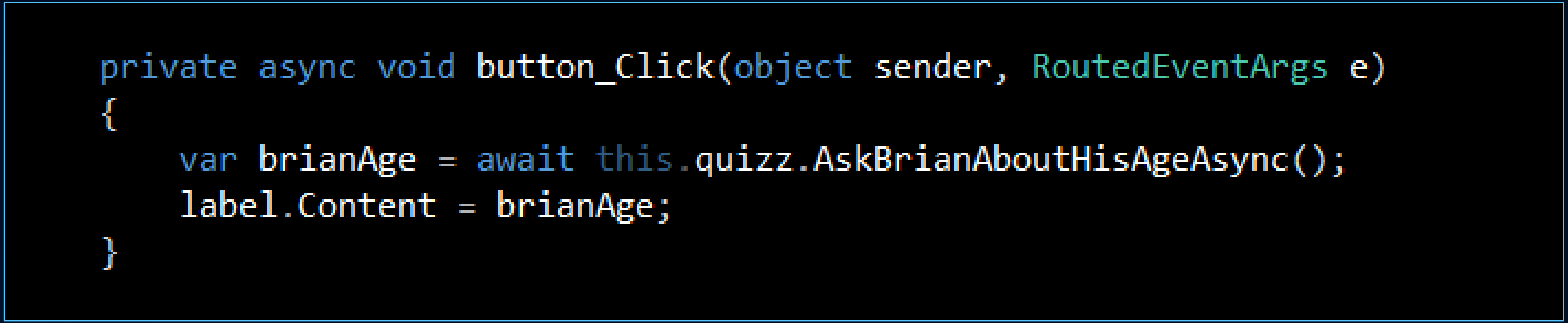
USE CONFIGUREAWAIT(FALSE) EVERYWHERE!



```
public async Task<int> AskBrianAboutHisAgeAsync()
{
    //var currentContext = SynchronizationContext.Current;
    var janetAge = await AskJanetAboutHerAgeAsync().ConfigureAwait(false);

    return janetAge + 3*Year;
}

private async Task<int> AskJanetAboutHerAgeAsync()
{
   await Task.Delay(10*Second).ConfigureAwait(false);
   return 39 * Year;
}
```

#2: ENTER THE ASYNC VOID

THE ASYNC VOID CASE



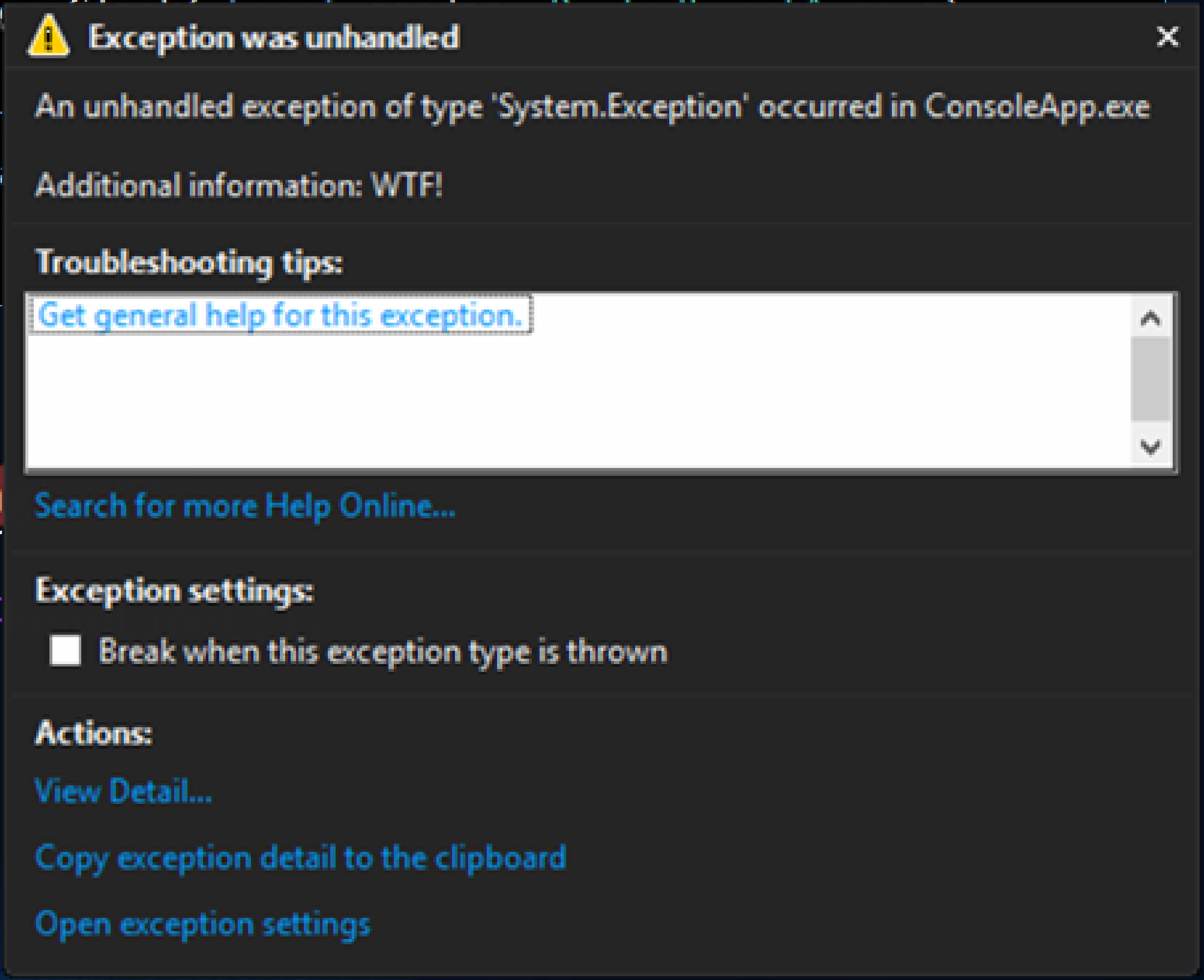
```
private async void button_Click(object sender, RoutedEventArgs e)
{
    var brianAge = await this.quizz.AskBrianAboutHisAgeAsync();
    label.Content = brianAge;
}
```

- Async void is a “fire-and-forget” mechanism…
- The caller is *unable* to know when an async void has finished
- The caller is *unable* to catch exceptions thrown from an async void
  - (instead they get posted to the UI message-loop)

THE ASYNC VOID CASE

```
private async void butt
{
    var brianAge = awai
    label.Content = bri
}
```

- Async void is a “fire-
- The caller is *unable* t
- The caller is *unable* t
  - (instead they get po



Exception was unhandled

An unhandled exception of type 'System.Exception' occurred in ConsoleApp.exe

Additional information: WTF!

Troubleshooting tips:

Get general help for this exception.

Search for more Help Online...

Exception settings:

☐ Break when this exception type is thrown

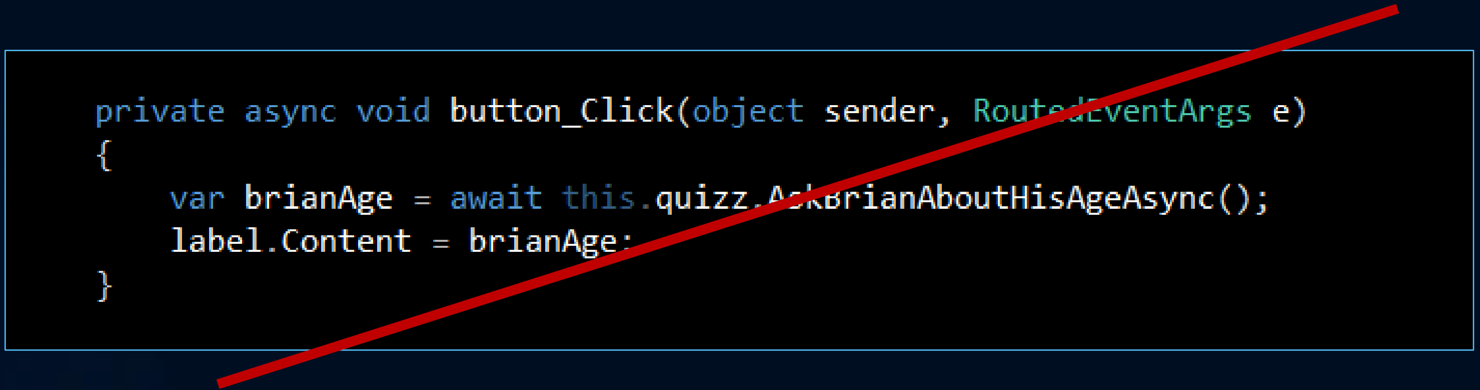
Actions:

View Detail...

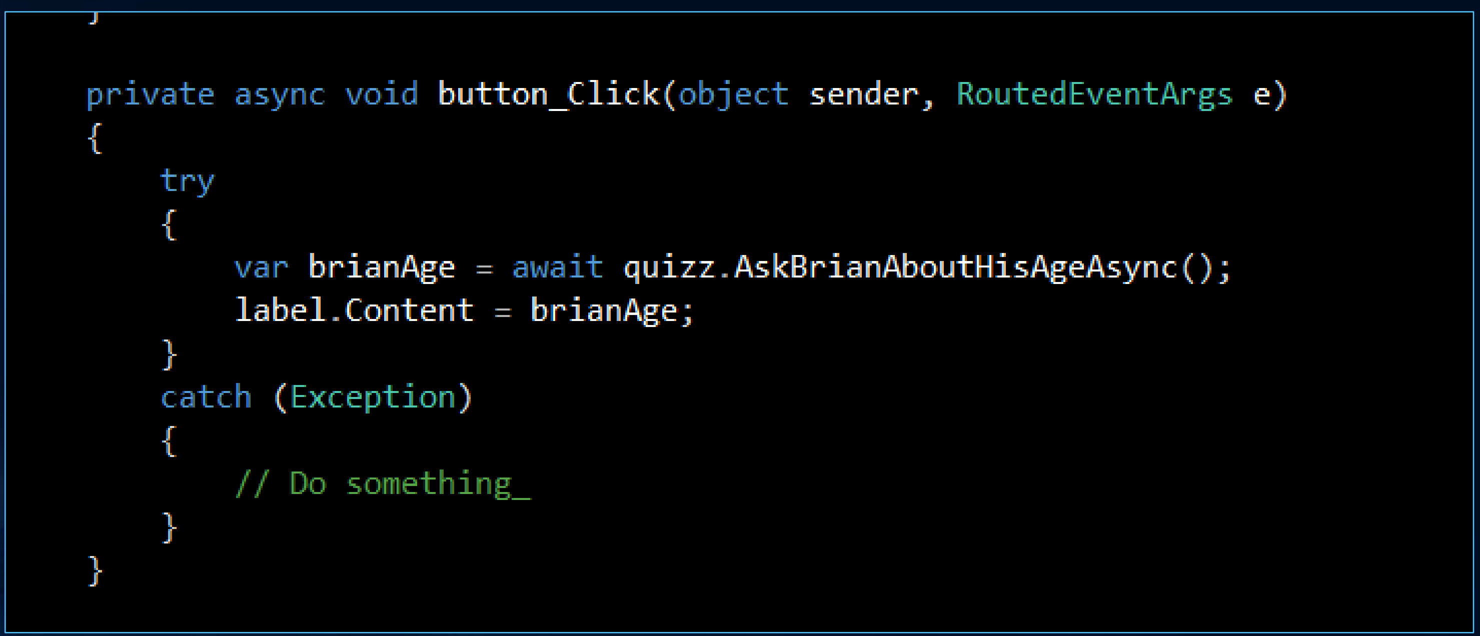
Copy exception detail to the clipboard

Open exception settings

TRY-CATCH YOUR EVENT HANDLERS!



```
private async void button_Click(object sender, RoutedEventArgs e)
{
    var brianAge = await this.quizz.AskBrianAboutHisAgeAsync();
    label.Content = brianAge;
}
```

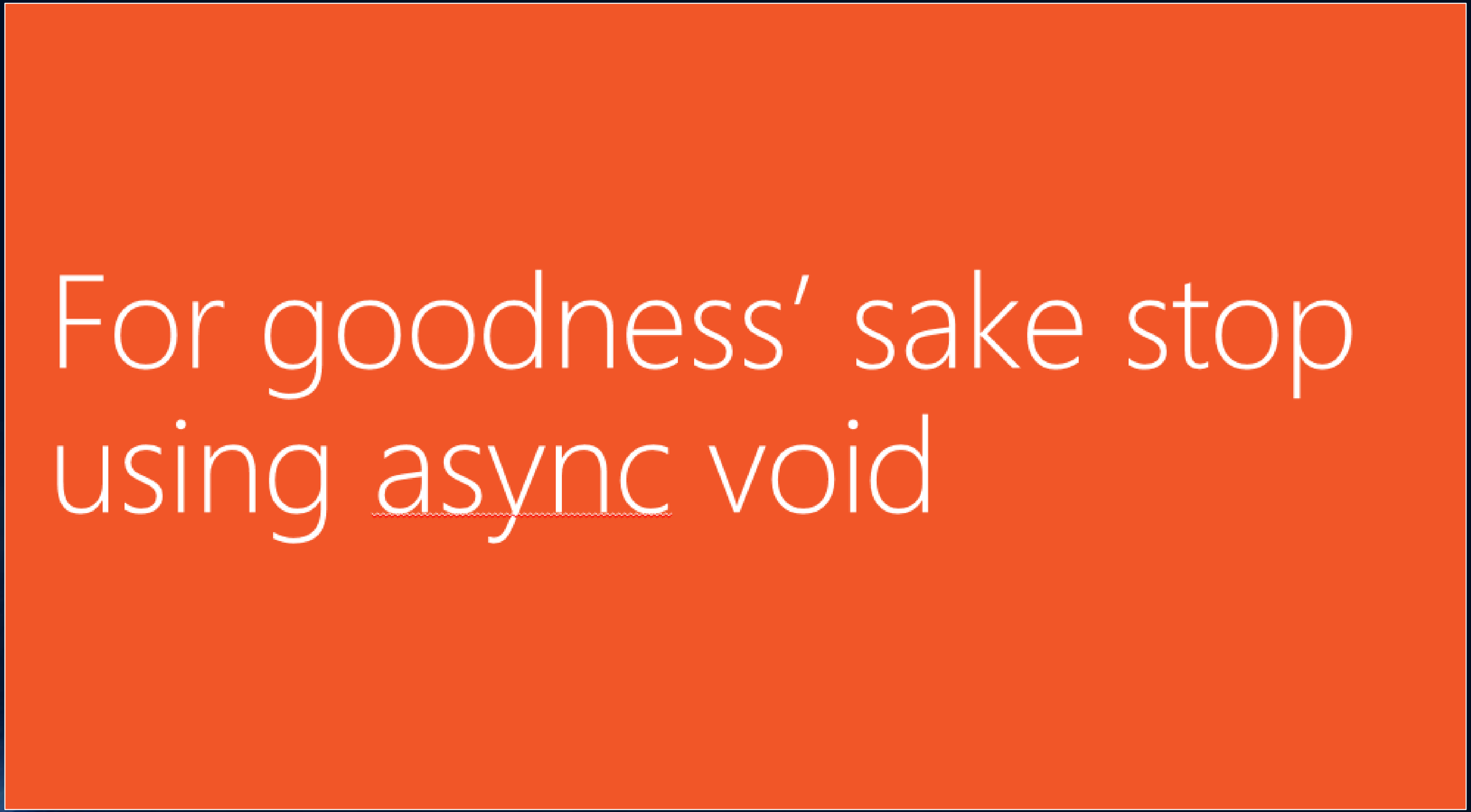


```
}

private async void button_Click(object sender, RoutedEventArgs e)
{
    try
    {
        var brianAge = await quizz.AskBrianAboutHisAgeAsync();
        label.Content = brianAge;
    }
    catch (Exception)
    {
        // Do something_
    }
}
```

MS GUIDELINES

- Use async void methods only for top-level event handlers (and their like)
- Use async Task-returning methods everywhere else
- Try-Catch your Async event handlers!

MS EVEN SAID:



For goodness' sake stop using async void

#3: USE IT WISELY

ONLY FOR I/O!

WRAP-UP

WRAP-UP

1. Never block! Unless you want to deadlock
   - ~~Locks, Wait without timeout, Task.Result…~~
   - Use top-level await when coding UI or Web
   - Use ConfigureAwait(false) everywhere within your libraries
2. Never create « async void » methods
   - And try catch all such existing event handlers
3. Only for I/Os

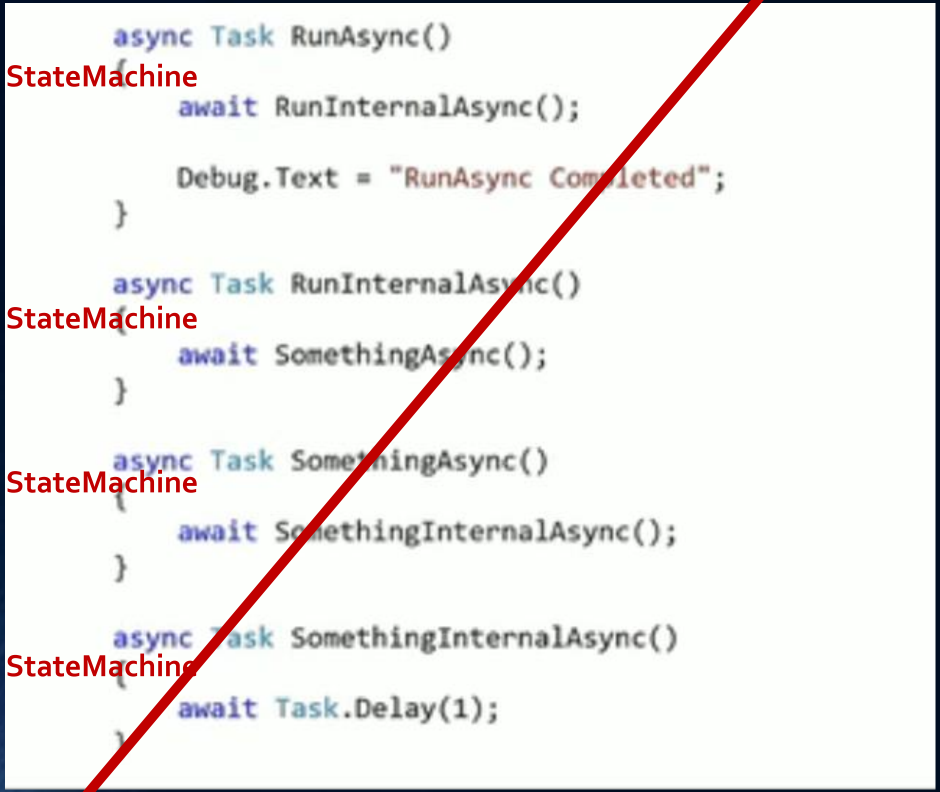
DON'T USE ASYNC-AWAIT

UNLESS YOU UNDERSTAND HOW IT WORKS

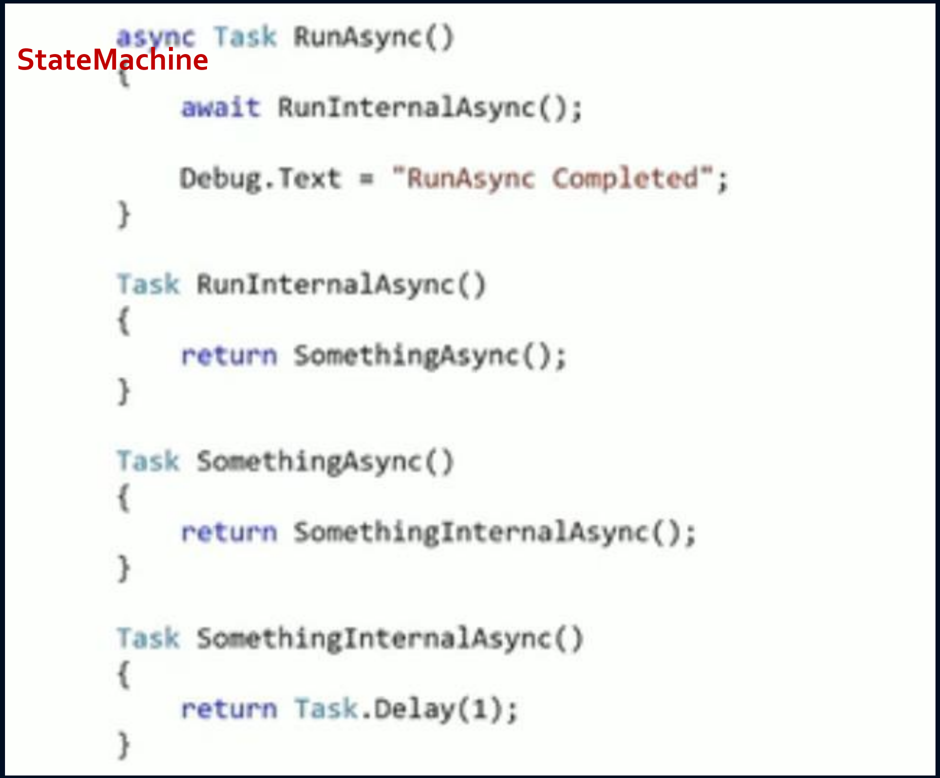
THANKS!

APPENDIX

DON'T SYSTEMATIZE ASYNC-AWAIT?



```
async Task RunAsync()
{
    await RunInternalAsync();

    Debug.Text = "RunAsync Completed";
}

async Task RunInternalAsync()
{
    await SomethingAsync();
}

async Task SomethingAsync()
{
    await SomethingInternalAsync();
}

async Task SomethingInternalAsync()
{
    await Task.Delay(1);
}
```



```
async Task RunAsync()
{
    await RunInternalAsync();

    Debug.Text = "RunAsync Completed";
}

Task RunInternalAsync()
{
    return SomethingAsync();
}

Task SomethingAsync()
{
    return SomethingInternalAsync();
}

Task SomethingInternalAsync()
{
    return Task.Delay(1);
}
```

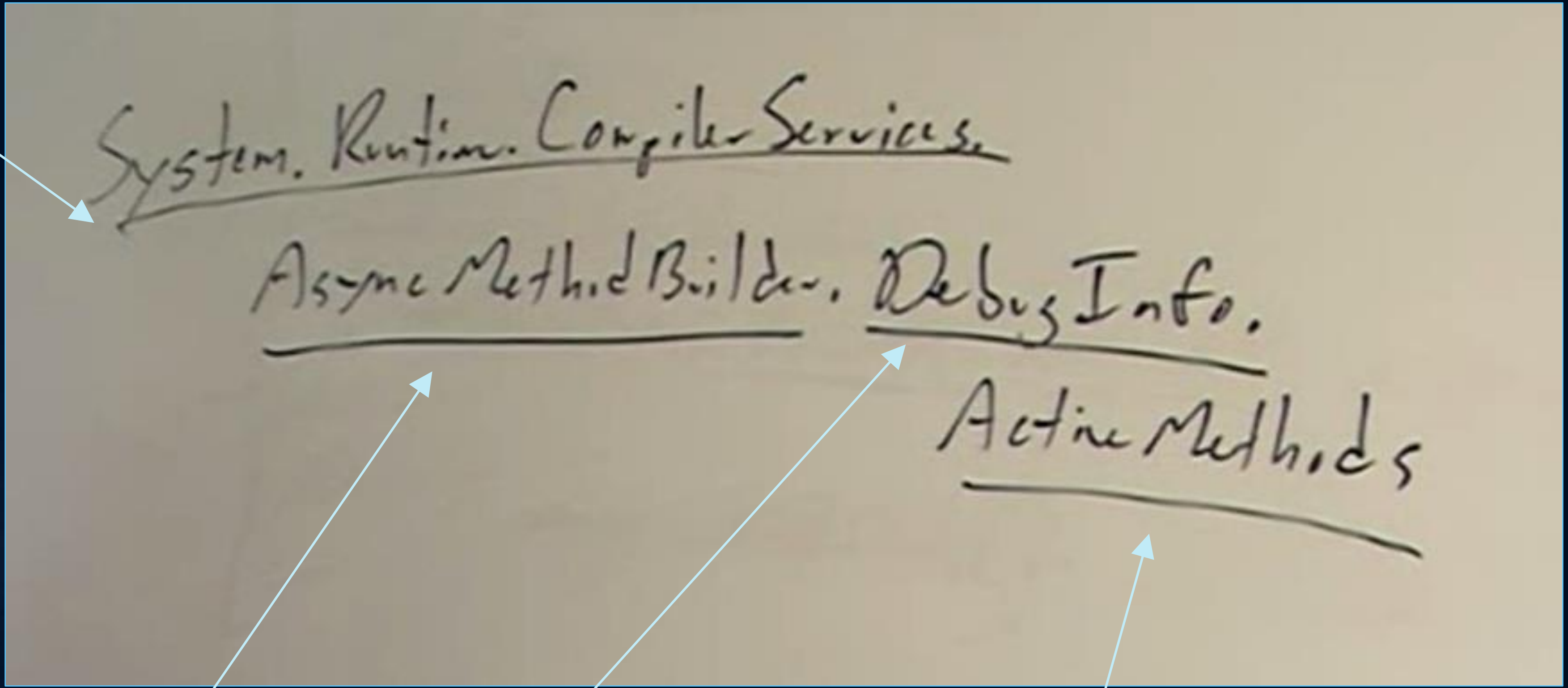
NOTHING IN THE CONTINUATION?

NO NEED FOR AWAIT! (UNLESS FOR 'USING')

ASYNC METHOD GC IMPACT

- 3 allocations for every Async method
  - Heap-allocated state machine
    - With a field for every local variable in your method
  - Completion delegate
  - Task

TROUBLESHOOTING?



FEW REFS

- Bart De Smet deep dive: https://channel9.msdn.com/Events/TechDays/Techdays-2014-the-Netherlands/Async-programming-deep-dive
- Filip Ekberg at Oredev 2016: https://vimeo.com/191077931
- Async-Await Best practices: https://msdn.microsoft.com/en-us/magazine/jj991977.aspx
- Compiler error: http://stackoverflow.com/questions/12115168/why-does-this-async-await-code-generate-not-all-code-paths-return-a-value
- Task.Run etiquette: http://blog.stephencleary.com/2013/11/taskrun-etiquette-examples-dont-use.html
- There is no thread: http://blog.stephencleary.com/2013/11/there-is-no-thread.html
- Does using Tasks (TPL) library make an application multithreaded? : http://stackoverflow.com/questions/23833255/does-using-tasks-tpl-library-make-an-application-multithreaded
- Eliding Async-Await : http://blog.stephencleary.com/2016/12/eliding-async-await.html