Practice Async # 2

Async and exception handling

Practice 1

Create winform app with button with the above implementation
In LongRunningTask it can be usedThread.Sleep
At first the window will be blocked

- [ ] Solve Deadlock UI

```
private async Task<int> TestAsync()
{
    buttonRun.Enabled = false;
    await LongRunningTask();
    buttonRun.Enabled = true;
    return 42;
}



private void btn_Click(object sender, EventArgs e)
{
    label.Text = TestAsync().Result.ToString();
}
```

Solution
Needs to be added async and await in btn_Click
For running tasks in the background, it needs to be added backgroundWorker on the UI

Practice 2

Async lambda and exceptions

- [ ] Add async method that throws exception InvalidOperationException

```
private static async Task<int> TestAsync()
```

- [ ] Call directly TestAsync
- [ ] Call TestAsync from an async lambda

```
async() => await TestSync
```

- [ ] Add try catch
- [ ] Catch exception InvalidOperationException

The try catch block needs to be added to catch the exception