”SOLID Programming”

Lab Manual

Wincubate ApS

07-04-2025

V2.0

Table of Contents

Exercise types

The exercises in the present lab manual differs in type and difficulty. Some exercises can be solved by applying the techniques from the presentations in the slides in a more or less direct manner. Such exercises are not categorized further.

However, the remaining exercises differs slightly in the sense that they are not necessarily easily solvable. These are categorized as follows:

★	Labs marked with a single star denote that the corresponding exercises are a bit more loosely specified.
★★	Labs marked with two stars denote that the corresponding exercises contain only a few hints (or none at all!) or might be a bit more difficult or nonessential. They might even require additional searches for information elsewhere than in the slide presentations.
★★★	Labs marked with three stars denote that the corresponding exercises are not expected in any way to be solved. These are difficult, tricky, or mind-bending exercises for the interested participants – mostly for fun! ☺

Prerequisites

The present labs require the course files accompanying the course to be extracted in some directory path, e.g.

```
C:\Wincubate\SOLID
```

with Visual Studio 2022 and .NET 8 (or later) installed on the PC.

We will henceforth refer to the chosen installation path containing the lab files as *PathToCourseFiles* .

Module 1: “The SOLID Principles”

Lab 01.1: “Write to both Console and File” (★)

This step emulates **Sprint 1** after having done the initial application in the presentation.

- Open the starter project in

  PathToCourseFiles`\Labs\Lab 01.1\Begin` ,

  which contains an adapted version of the solution we produced in the presentation.

The story described on the Sprint 1 board reads:

"Allow writing results to both console storage and file storage"

- Implement the story in a SOLID manner.

Lab 01.2: “Send an SMS”

This step emulates Sprint 2 after having done the initial application in the presentation.

- Open the starter project in
  PathToCourseFiles`\Labs\Lab 01.2\Begin` ,
  which contains an adapted version of the solution we produced in the presentation.

The story described on the Sprint 2 board reads:

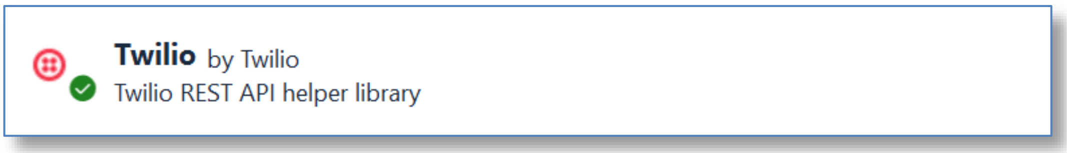
“Implement functionality for sending an SMS with the result”

- Implement the story in a SOLID manner
  - Create a SMS transmission strategy using the Twilio SMS API.
  - Test your implementation by sending yourself the results as SMS messages.
    - It may take up to 30 to 45 seconds for the SMS to arrive.
    - Note: Since Twilio is an international service, do remember to prefix your number with “+45”

Background

See the Twilio SMS API documentation at https://www.twilio.com/docs/sms/api/message-resource .

In order to use the Twilio API for sending SMS messages you must include their nuget package into your project:



Twilio by Twilio
Twilio REST API helper library

Once it is included, you can send an SMS message using the following code:

```
string _accountSid = "ACa5?64844f11c4152c5e4db4bc202c7??";
string _authToken = "4f14?6d?4826993?6c15a02a8605882b";

TwilioClient.Init(_accountSid, _authToken);
```

```
MessageResource mr = await MessageResource.CreateAsync(
    to: new PhoneNumber("<phone number>"),
    from: new PhoneNumber("+4676???9439"),
    body: "<contents of SMS>"
);
```

Your instructor will supply you with the remaining digits substituting the missing ?’s in the above codes.

Lab 01.3: “Retry the storage writes” (⭐⭐)

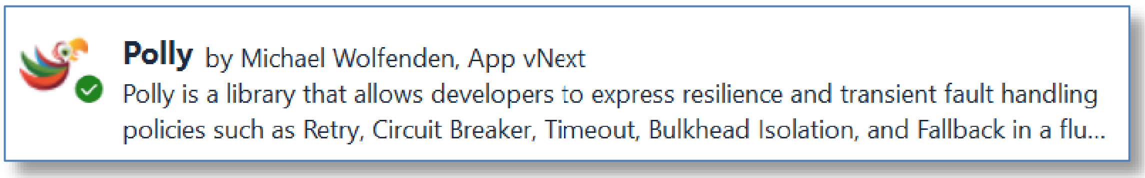
This step emulates Sprint 3 after having done the initial application in the presentation.

- Open the starter project in
  PathToCourseFiles`\Labs\Lab 01.3\Begin` ,
  which contains an adapted version of the solution we produced in the presentation.

The story described on the Sprint 3 board reads:

“Implement a retry strategy for writes”

Many such strategies can be implemented in a very simple fashion using the Polly nuget package:



Polly by Michael Wolfenden, App vNext
Polly is a library that allows developers to express resilience and transient fault handling policies such as Retry, Circuit Breaker, Timeout, Bulkhead Isolation, and Fallback in a flu...

You can read more about it here: https://www.nuget.org/packages/Polly/

Your task is now:

- Implement the story in a SOLID manner
  - Do whichever strategy you find correct, e.g. *“retry 3 times and then fail”* or similar.