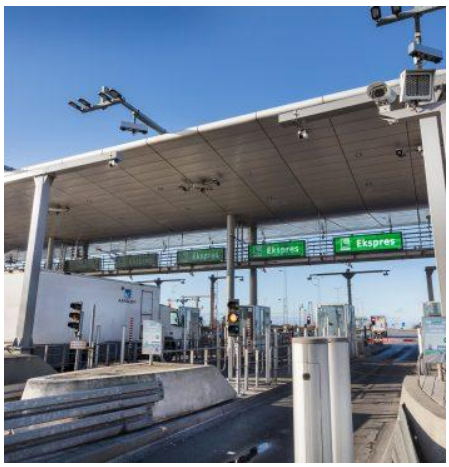
Denne obligatoriske opgave består af en programmerings del og en teknik del
Alt koden for alle opgaverne afleveres i wiseflow.

Opgaverne i programmering:

The Bridge Ticket System



Intro: We are implementing a ticketing system that will help various bridge companies in Denmark to charge/pay for crossing the bridge in cars and Motorcycles. The idea is to make a standard solution (therefore using a library) that will be used for two customers, Storebælt broen and Oresundsbron.
The standard solution must be changed to fit the individual's customers' needs (using overrides etc.). The default solution must be provided as a DLL that the individual customers can continue to work on. Each customer will also have an DLL (so 3 DLL's all in all at the end)
The solution must be **tested** and **well documented** through summaries of the individual classes and methods implemented.
All the code should be pushed to **GitHub**!

You are welcome to use more methods and properties than stated if that can help you.

What should be at GitHub?
The code for the libraries (DLL), the Repository and the test project

3. semester 2025F, obligatorisk opgave

What should you hand in at Wiseflow?
All the code zipped and the DLL's + XML

Task 1

Create a ClassLibrary project "Bridge" in Visual Studio.
Add it to git and Push to GitHub.

Task 2

Implement a class "Car", which has the following properties

- public string Licenseplate
- public DateTime Date

and the following methods

- public double Price() The price is fixed at 230 which must be returned
- public string VehicleType() which returns the string "Car"

Add a test project and test the two methods.

Note: Remember to give the test method a descriptive name

Note2: Did you remember to document your class and methods

Remember to commit and push

Task 3

Implement a Class "MC" that has the following properties

- public string License plate
- public DateTime date

and the following methods

- public double Price () .The price is fixed at 120 kr which must be returned
- public string Vehicle (). which returns "MC"

Add test in the test project that tests the two methods.

Remember to commit and push

Task 4

The chief It-Architect now sees that there is the possibility of using inheritance. Change the classes so they now inherit from a common base class. Consider which members to move to the baseclass and which members there should be in the specialized class.
Also consider if “abstract” should be used and also the “access modifiers” for the properties (public/protected/private)

After refactoring, run the test again to make sure the program works as before.
Remember to commit and push

Task 5

Now functionality must be implemented so that it is not possible to enter a license plate that are longer than 7 characters, if it is, an appropriate exception must be thrown.

Write a test that tests this.

Remember to commit and push

Task 6

A functionality must be implemented so that it is possible to specify whether a Brobizz (*) is used when buying a ticket, if a Brobizz is used, a standard 10% discount is given on the ticket price.

Write test that tests this, remember to use the overloaded static Assert AreEqual method where you can specify the delta. Check how much you've tested your code through Code Coverage.

Remember to commit and push

(*) A Brobizz is a way to get discount for returning customers

Task 7 – Generate the standard DLL

You are now finish with developing the standard functionality of “The Bridge Ticket” program. So now it’s time to generate the DLL and XML for the standard solution.
Generate:

- The DLL
- The XML file, who describe the methods and classes

Copy these files into a folder, for example at your desktop so you know where they are.

The DLL and XML file should be used in the upcoming task

Task 8 Use the standard DLL for the new customer: ”Storebæltsbroen”

The company “Storebæltsbroen” would like you to develop a little more functionality at the standard solution. Since it is a customer specific solution, this solution must be in its own library.

Add a new Library to your solution. The name of the Library should be ”StoreBaeltTicketLibrary”.

Create folder in your class library, name it ”extern dll” and copy your DLL + XML in here. Remember to create a reference to your DLL.

Task 9 – Storebæltsbroen weekend discount

A weekend discount is now to be introduced which means that if you drive across the bridge, Saturday or Sunday you get a 15% discount. This discount applies **only** to cars. The Brobizz discount must be deducted afterwards the weekend discount.

Check how much you've tested your code through Code Coverage. Write Test so you have covered your code reasonably. NOTE: Remember to give the test methods a descriptive name

Remember to commit and push

Task 10 – New customer: the company ”Oresundbron”

NOTE: In this task, a new Library must be created again, and you must then "build" on the standard DLL "TicketLibrary" you have already encoded. So be sure to add a new library project and reference to DLL + XML.

An agreement is now being reached with the “OresundBron” company that the ticketing system will also be able to handle cars and MC running across the bridge at Øresund. It is fortunate, of course, that we have coded it all as a library for then this library should just be expanded with some more functionality.

This functionality must be implemented:
The regular price for car is 460 kr. and for MC is 235kr.
If you have a Brobizz agreement the price for car is 178 kr. and for MC 92 kr.

The VehicleType method must return the following for cars: "Oresund car" and for motorcycles: "Oresund MC"

Check how much you've tested your code through Code Coverage. Write Test so you have covered your code reasonably.

Task 11 Repository class for the “Storebæltbroen”

You should now make it possible to use the Storebæltbroen DLL through a repository.

Add a Repository Class that contains a static list for all the tickets at “Storebæltbroen”

Add logic so it is possible to add(buy) a new ticket through the repository
Add logic so it is possible to see all the tickets through the repository
Add logic so it’s possible to see all the tickets for one specified licenseplate through the repository

Extract the interface from the repository class, which have all the three methods above.

Opgaverne i teknik:

Husk at gemme jeres løsning på GitHub

Opgave C#, TCP, simpel protokol

Du skal lave en **TCP server i C#** og benytte SocketTest som klient, der skal overholde den følgende protokol.

Serveren skal kunne modtage en string, som der kan indeholde en af følgende værdier (se skema):

Alt hvad der er skrevet i <> skal erstattes med værdier, inklusiv de 2 <> karakterer

Kommunikationen foregår i 2 steps. Først sender klienten en kommando (1), som serveren svarer tilbage på med en besked (2), og så sender klienten 2 tal til serveren adskilt af mellemrum (3), som server sender resultatet af (4).

Klient sender	Server svarer	Klient sender eksempel	Server svarer eksempel
1:Random 3:<tal1> <tal2>	2: Input numbers 4: <random tal mellem tal1 og tal2, begge inkluderet>	1: Random 3: 1 10	2: Input numbers 4: 3
1: Add 3: <tal1> <tal2>	2: Input numbers 4: <summen af de 2 tal>	1: Add 3: 3 8	2: Input numbers 4: 11
1: Subtract 3: <tal1> <tal2>	2: Input numbers 4: <tal2 trukket fra tal1>	1: Subtract 3: 19 4	2: Input numbers 4: 15

Bemærk, i både random og subtract er rækkefølgen vigtig.

Serveren skal være lavet så den kan blive ved med at håndtere flere klienter samtidigt (concurrent).

Klienten skal spørge brugeren om de 3 værdier, altså funktion Random eller Add eller Subtract, samt tal1 og tal2. Den skal altså spørge 3 gange.

Hvordan den spørger efter "metoden" er op til dig selv

Opgave JSON

Du skal lave en ny version af den tidligere TCP protokol, der kommunikerer med JSON objekter i stedet for simple strings.

Serveren skal kunne de samme metoder (dog på et nyt portnummer), men al kommunikation skal ske i JSON format i stedet og kun bruge ét kald fra klienten.

Det er op til dig at beslutte, hvordan JSON strengene skal se ud. Altså hvilke properties, der er i JSON kommunikationen.

Du skal igen benytte SocketTest som klient, der stadig spørger brugeren om de 3 input, men så pakker den dem ind i JSON format som din server skal kunne forstå.

JSON formatet kunne se ud som følger (kun et forslag):

{"method": "Random", "Tal1": 10, "Tal2": 20}

I denne version skal serveren sende noget tilbage (fejlbesked?), hvis klienten ikke overholder protokollen, du har defineret.

Aflevering i wiseflow