# Group 36 Assignment 2

Clara Lyngeraa - clly
Asger Ramlau Jørgensen - asjo
Nicklas Kristoffer Brichs Jensen - nkrj

September 2022

**Link to the Github repository**
`https://github.com/Clara-Lyngeraa/assignment-02`

## 1 Foreword

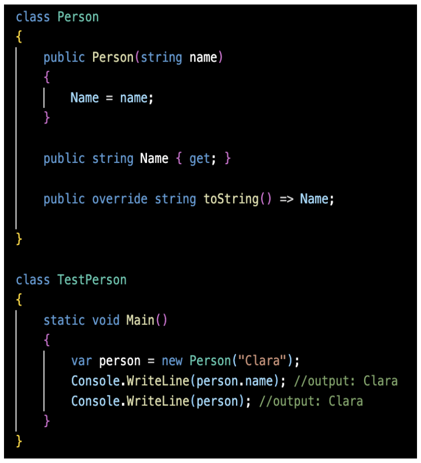
Due to illness in the group and a group members unforeseen personal matters we have not been able to finish all exercises.

## Contents

## 2 Types

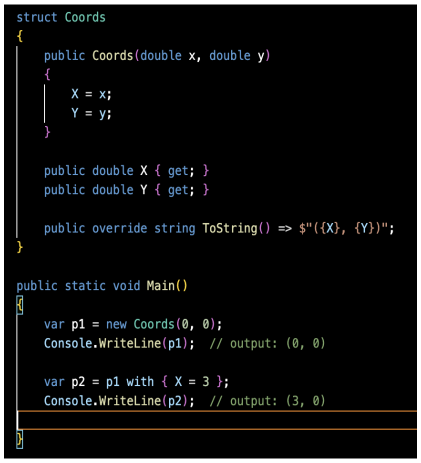
A class is a *reference type* this means that when you create a variable of a class, that variable stores references to the class' objects. A struct is a *value type* this means that a variable of a struct contains an instance of the stuct.

With a class two variables can reference the same object, so operations on one variable can affect the other variable's object. With variables of a struct, the variables contains their own copy of the data, so the operations of one variable cannot affect the other.



```
class Person
{
    public Person(string name)
    {
        Name = name;
    }

    public string Name { get; }

    public override string toString() => Name;

}

class TestPerson
{
    static void Main()
    {
        var person = new Person("Clara");
        Console.WriteLine(person.name); //output: Clara
        Console.WriteLine(person); //output: Clara
    }
}
```

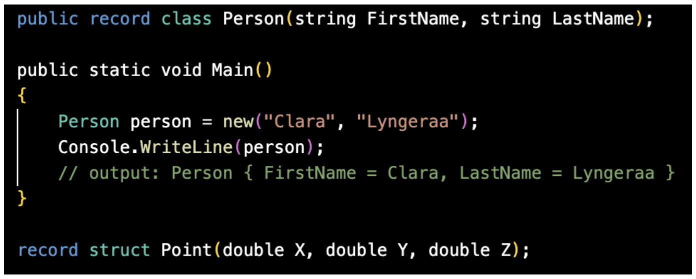
(a) Example of a class



```
struct Coords
{
    public Coords(double x, double y)
    {
        X = x;
        Y = y;
    }

    public double X { get; }
    public double Y { get; }

    public override string ToString() => $"({X}, {Y})";
}

public static void Main()
{
    var p1 = new Coords(0, 0);
    Console.WriteLine(p1);  // output: (0, 0)

    var p2 = p1 with { X = 3 };
    Console.WriteLine(p2);  // output: (3, 0)

}
```

(b) Example of a struct

A record class and record struct are record types. Record types use value-based equality, which means that two variables are equal if the record type definitions are identical and if for every field, the values in both records are equal. A record class is a reference type and a record struct is a value type.



```
public record class Person(string FirstName, string LastName);

public static void Main()
{
    Person person = new("Clara", "Lyngeraa");
    Console.WriteLine(person);
    // output: Person { FirstName = Clara, LastName = Lyngeraa }
}

record struct Point(double X, double Y, double Z);
```

Figure 2: Example of record class and record struct

# 3 Exercise 1

## 3.1 Describe the difference between a scenario and a use case. Describe for each of the two concepts when and for what they are used.

A *scenario* is a description of how a system can be used for a particular task. It is a textual description with a clear outline. A *scenario* is very structured, not very descriptive. They have a specific input and output.

*Use-cases* are a way of describing interactions between users and a system using a graphical model and structured text.

The difference between a scenario and a use-case is that use-cases are documented using a use case diagram.

# 4 Exercise 2

## 4.1 Identify and briefly describe four types of requirements that may be defined for a computer-based system.

### 4.1.1 User Requirements

These requirements are often in a natural language, with diagrams, showing the services the system provides.

### 4.1.2 System Requirements

this is often a structured document that sets out a description of the system's functions and operational constraints. Often described in detail.

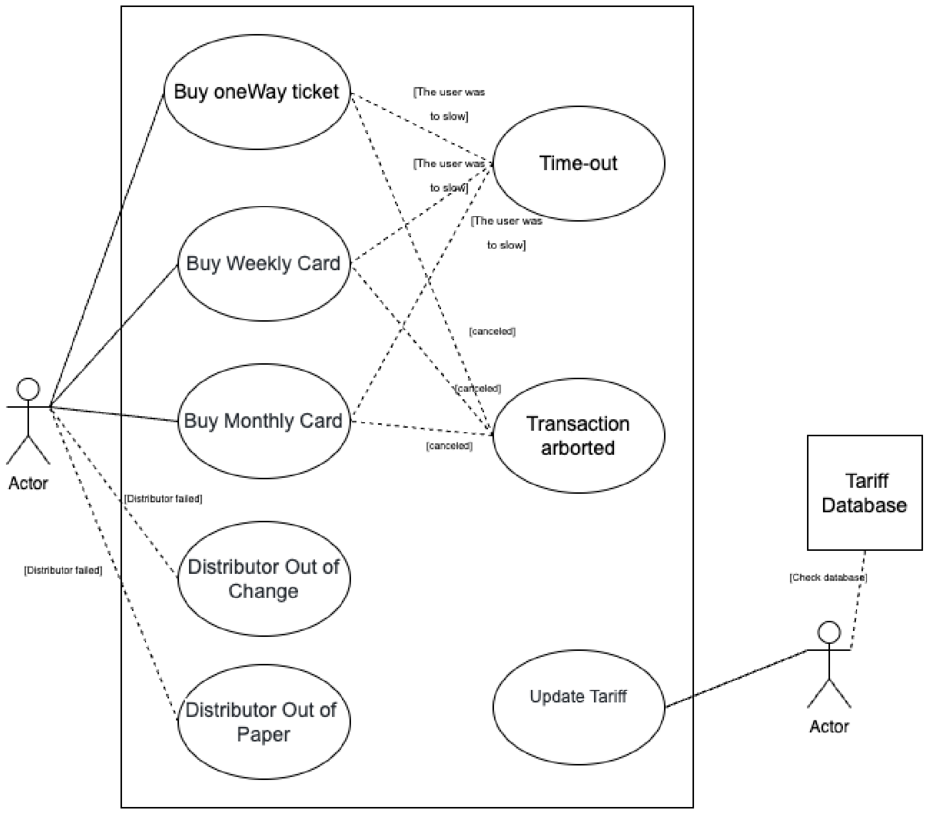
### 4.1.3 Functional Requirements

The statements of services the system should provide. This is how the system should react and behave in particular situations.

### 4.1.4 Non-functional Requirements

Constraints on the services and functions the system offers, such as timing constraints or financial constraints. Often applied to the system as a whole, and not just individual features.

## 5 Exercise 3


Buy oneWay ticket
Buy Weekly Card
Buy Monthly Card
Distributor Out of Change
Distributor Out of Paper
Time-out
Transaction arborted
Update Tariff
Tariff Database
Actor
Actor
[The user was to slow]
[The user was to slow]
[The user was to slow]
[canceled]
[canceled]
[canceled]
[Distributor failed]
[Distributor failed]
[Check database]

# 6 Exercise 4

## 6.1 In the following, you find a requirement that is specified for the "samlet socialfagligt it-system" (also called VUM 2.0):

### 6.1.1 Discover formulations in the requirement that are ambiguous.

1. **Medarbejdere skal digitalt understøttes i udførelsen af deres kerneydelser, og derved skal den digitale understøttelse være medvirkende til, at kommunen er et attraktivt sted at arbejde.**
   (a) This section describes how the systems in place should contribute to a more atractive workplace, but it's not specified how this would be the case. Good systems doesn't necessarily contribute to a good work environment.
2. **Det er altafgørende, at medarbejderne føler sig godt understøttet af den nye digitale løsning.**
   (a) This statement is slightly problematic because it's not possible for them to make a system that is going to satisfy every individual employee.

### 6.1.2 Is there any information missing in the requirement?

They describe what they need to do to streamline the process but not how they are going to implement it.

### 6.1.3 The requirement specifies a set of non-functional requirements. What is problematic about there formulation?

### 6.1.4 Rewrite the requirement according to what you identified as problematic in the three bullet points above.

# 7 Exercise 5

## 7.1 Identify actors that interact with a music tracker software system.

For a software music tracker system we imagine that there would be different actors:

A user who would use the system for making music.

An administrator who would maintain and update the software.

Some form of hardware which could actually run the software.

Other systems probably would help in regards to saving music files, or downloading different instruments or sounds for your music software.

## 7.2 Formulate three use cases in structured language that a software music tracker system has to support.

| Usecase nr | 01 |
|---|---|
| Name | find instrument and play it |
| purpose, description | a user wants to find a specific instrument in the software, and play it, to hear how it sounds. |
| initiating actor | user |
| initiating act | The user wants to test out how a particular instrument sounds in the music software. |
| Starting conditions | The user has the music software installed and knows how to navigate the machine he or she is using, to run the software. |
| Main course | The user would use the music softwares GUI in order to search in the collection of instruments which are available in the software. Ideally the user would be able to search in the collection of instruments, to find the one he/she is looking for.<br>The user finds and chooses the desired instrument.<br>The user must now pick which note he wants the instrument to play.<br>The user now hits a play button of some sort, and the desired note will play. |
| result | The user has found and listened to the desired note and instrument |
| Ending state | N/A nothing in the software has changed |

| Usecase nr | 02 |
|---|---|
| Name | creating and listening to a song |
| purpose, description | a user wants to try to make a song, by combining the different instruments and sounds . |
| initiating actor | user |
| initiating act | The user wants to create a song using the softwares features |
| Starting conditions | The user has the music software installed and knows how to navigate the machine he or she is using, to run the software. |
| Main course | The user starts by opening the program and via the GUi he/she chooses to make a new ‘pattern’. The pattern is the template for creating a song.<br>The user would use the music softwares GUI in order to search in the collection of instruments which are available in the software. Ideally the user would be able to search in the collection of instruments, to find the one he/she is looking for.<br>The user finds and chooses the desired instrument.<br>The user must now pick which note he wants the instrument to play.<br>The user now has to add the instrument/note into the ‘Pattern’ template..<br>The user can add, move and remove multiple notes and sounds into this pattern and choose where in the ‘Pattern’ he/she wants to put the selected notes.Depending on the hardware this can be done by using the keys on the display or via mouse interaction with the GUI.<br>The user can play and pause the ‘Pattern’ in order to listen to particular parts of the ‘pattern’ or just the whole thing. |
| result | The user has now tried to create a song and has listened to the outcome |
| Ending state | An unfinished pattern template which can be edited, deleted or saved. |

| Usecase nr | 03 |
|---|---|
| Name | Saving a created song |
| purpose, description | a user wants to save a song they created |
| initiating actor | user |
| initiating act | The user wants to save a song which they have created using the software. |
| Starting conditions | The user has the music software installed and knows how to navigate the machine he or she is using, to run the software. The user has also created a song using the software and is now ready to save it. . |
| Main course | Via keys on the display the user could interact with the softwares gui to save the file.<br>the user would be able to select a field 'file' or key that could save the file.<br>By selecting this would choose where to store the desired music project, and what to call it. This could be obtained by interacting with the keys on the display<br>When the user has chosen where they wish to store their project they can click on save, and the desired file would now be saved. |
| result | The user has now successfully saved a music project |
| Ending state | The software would still display whatever interface the user was on, when they saved their project. |

## 7.3 Express three non-functional requirements for a music tracker software system.

1. Copyright issues. The sounds and instruments you can pick in the software have to be able to be used without it being copyrighted.

2. Sound. You have to be able to have some functional speakers either embedded in the tracker itself, or having the option of hooking connecting speakers or a headset to the hardware, in order for you to hear the music and the system should be fast enough to play the desired sound without buffering or any delay

3. Memory. The system should be able to save and store music projects, which can be accessed at a later time.