# group-4-adjr-aguh-fefa-assignment-02

adjr, aguh, fefa

September 2022

## 1 C#

GitHub repository

### 1.1 Classes

**Class:**
The traditional way of defining a type of data. A reference type. Useful for defining very complex data, data where inheritance occurs. Also for structuring the application (e,g, classes representing Model, View and Controller), Factory-classes that don't contain data themselves.

**Struct:**
Unlike record and class, struct is not a reference type (the variable contains an instance, not just a reference). This is useful for defining own data-types that e.g. are quite simple, don't use inheritance.

**Class record:**
Records use value-based equality, meaning that comparing 2 identical records returns true, even though they might be 2 different instances. If the data one wants to store is more complex, immutable, then it is useful to use records. Could represent a tuple in a database.

**Struct record:**
Struct records are like class records but with write-privileges, meaning that you can both write to and read from struct records.

## 2 Software Engineering

### 2.1 Exercise 1

Use cases look at individual interactions that can happen between the user and the system. A scenario, also called a story, consists of one or many interactions and describes a journey that the user might go through. Stories are easier to understand for those not playing a role in the development of the system.

## 2.2 Exercise 2

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

**System requirements:**

Document with detailed description of functions, operational constraints and services. Should define exactly what there is going to be implemented, usually part of a deal between developers and buyers.

**User requirements:**

Broad statements or detailed descriptions about the services that are expected to being provided to the system users. These requirements are described in natural language or by diagrams.

**Non functional requirements:**

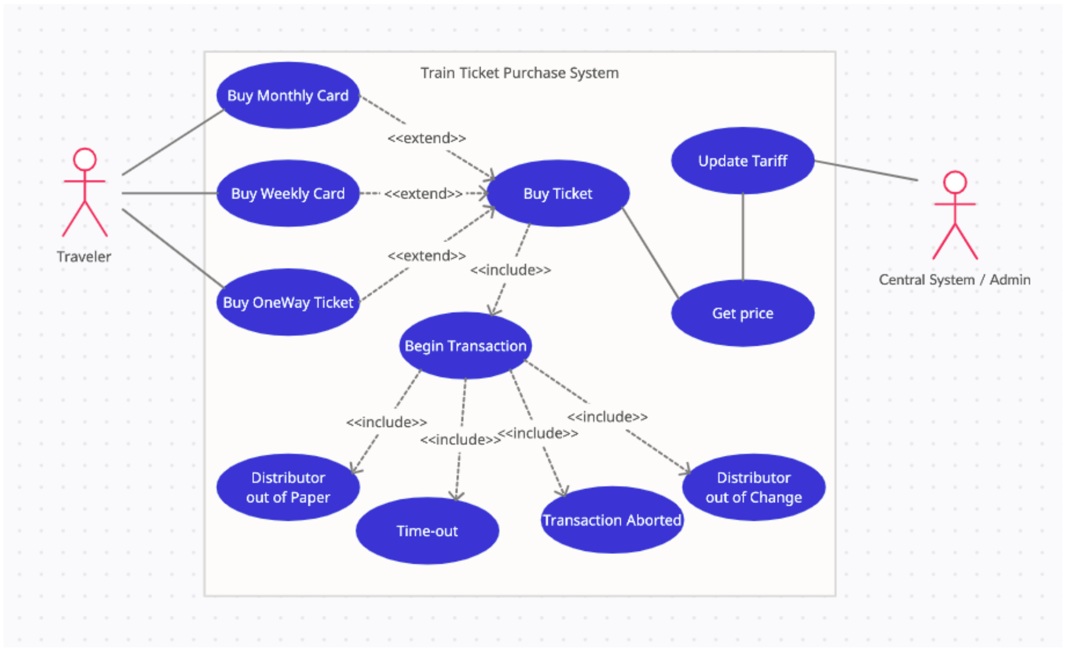
Constraints in regards to the services of the system. Could be timing constraints, development process constraints etc. These requirements often apply to the whole system and not only specific features or services of the system

**Functional requirements:**

Statements stating what services are to be provided, how the system will react to different input and the behaviour of the system in different situations. Can also state what a system should not do.

## 2.3 Exercise 3

Draw a use case diagram for a ticket distributor for a train system. The system includes two actors: a traveler, who purchases different types of tickets, and a central computer system, which maintains a reference database for the tariff. Use cases should include: Buy OneWay Ticket, Buy Weekly Card, Buy Monthly Card, Update Tariff. Also include the following exceptional cases: Time-Out (i.e., traveler took too long to insert the right amount), Transaction Aborted (i.e., traveler selected a cancel button without completing the transaction), Distributor Out of Change, and Distributor Out of Paper.



## 2.4 Exercise 4

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.

Kommunen har forskellige faglige målgrupper, som møder borgerne forskellige steder, og som arbejder, hvor borgerne er. Det kan være på gaden, i borgerens eget hjem, men også på dag- og døgninstitutioner uden for kontoret. Derfor er det vigtigt, at store dele af løsningen kan afvikles på mobile enheder.

Det er altafgørende, at medarbejderne føler sig godt understøttet af den nye digitale løsning. Det gælder både i forhold til at effektivisere tunge arbejdsgange via genbrug af data, deling af data og ikke mindst, at løsningen er medvirkende til god lovmedholdelighed i sagsbehandlingen, hvilket tilsammen kan understøtte en faglig stolthed.

Et andet og vigtigt parameter er, at kommunen gerne vil fastholde sine dygtige medarbejdere med en løsning, der understøtter og guider dem i deres daglige opgaveløsning. En stabil og driftssikker løsning er en medvirkende faktor til større tilfredshed med ansættelsen i kommunen.

Ambiguous passages are highlighted.

**Is there any information missing in the requirement?**
Apart from missing further detailing of the ambiguous parts of the requirement, some quantitative way of measuring the requirement is also missing. How are

peoples feelings evaluated?

**The requirement specifies a set of non-functional requirements. What is problematic about their formulation?**
It is hard to differentiate between the 'sub-requirements',

## 2.5 Exercise 5

**Identify actors that interact with a music tracker software system.**

- User
- Database of instruments
- Database of sounds/notes
- Keyboard / Input buttons
- Speaker

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

| **Music Tracker: Fill track with notes** | |
|---|---|
| **Actors** | User, sound database, keyboard, instrument database |
| **Description** | User can highlight a track and fill it with notes, defining the number of events, scale, fill type, and a range of the notes |
| **Data** | Instruments and sounds |
| **Stimulus** | User interaction with keyboard |
| **Response** | Display shows inserted notes |

| **Music Tracker: Play music** | |
|---|---|
| **Actors** | User, keyboard, speaker |
| **Description** | The user can hit a play button to play the music currently created on the music tracker through the speaker |
| **Data** | Instruments, notes and currently created music |
| **Stimulus** | Customer clicks play button on keyboard |
| **Response** | Music starts playing from the speaker actor |

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

- The tracker should pack soundtracks/songs in formats that take up as little space as possible(Normally this would also include a maximum size requirement but it is not available from the video).
- The user can with only one button click disable specific distractions.
- The music tracker must show the created music in a spreadsheet like manner.

| **Music Tracker: Preview fractions of music** | |
|---|---|
| **Actors** | User, keyboard, speaker |
| **Description** | The user can hit highlight a number of rows and tracks and only play the selected music through the speaker. |
| **Data** | Instruments, notes and currently created music |
| **Stimulus** | Customer clicks play buttons on keyboard |
| **Response** | Music starts playing from the speaker actor |

## 2.6 Exercise 6

Write down a use case in structured language that a canteen payment system has to support. For this use case write down three requirements and categorize them as functional or non-functional.

| **Canteen Payment System: Purchase warm/weighed lunch** | |
|---|---|
| **Actors** | Customer, price database, weighing system, credit card terminal |
| **Description** | A customer will put their food on a weight where the payment system will pull data from the weighing system and price database actors.<br>The system will calculate the price for the food and enable a purchase button.<br>The user will upon pressing the purchase button begin a transaction process in which the credit card terminal actor will receive the price to be paid.<br>Once the customer has inserted their credit card and possibly pin code, the credit card terminal will pull the money from the customer's account and deposit the money into the canteen's bank account. |
| **Data** | Food prices |
| **Stimulus** | Customer begins a purchase transaction for their food |
| **Response** | Approval or declination of credit card transaction from the credit card terminal |
| **Requirements** | - System must have a credit card terminal/integration (functional)<br>- System must know updated prices of food (functional)<br>- System must be deployable to a device that fits within a canteen line (non-functional) |