

**Softwareproject**

**The Matrix**

Raman Talwar

Othello Clemens

Thomas Pellegrims

Gaetan Van Acker

Begeleiders:

Leen Brouns

Veerle Ongenae

*Universiteit Gent*

*Faculteit Ingenieurswetenschappen en Architectuur*

*2020-2021*

Inhoud

[Inleiding 3](#_Toc66982304)

[Use Case Diagrams 3](#_Toc66982305)

[Uitleg over usecase diagram 3](#_Toc66982306)

[Use case 1: Matrixvermenigvuldiging aanleren 3](#_Toc66982307)

[Use case 2: Interactieve matrixvermenigvuldigingen 4](#_Toc66982308)

[Use case 3: Determinanten 4](#_Toc66982309)

[Use case 4: Transponeren 5](#_Toc66982310)

[Use case 5: Inverse 5](#_Toc66982311)

[Use case 6: Toepassing 1: ‘Decrypt the wallet’ (Hill cipher encryptie) 6](#_Toc66982312)

[Use case 7: Toepassing 2: Markov ketens 7](#_Toc66982313)

[Sequentiediagram 8](#_Toc66982314)

[Uitleg over sequentiediagram: 9](#_Toc66982315)

[Klassendiagram 10](#_Toc66982316)

[Uitleg over klassendiagram 10](#_Toc66982317)

# Inleiding

In deze analyse zal het project worden beschreven aan de hand van verschillende diagrammen. Deze tool kan gebruikt worden door leerlingen om te leren werken met matrices en het nut ervan te ontdekken.

Het idee van dit project is om leerlingen kennis te maken met matrices en wat de verschillende bewerkingen ervan zijn. Dit wordt aangetoond met een stap-voor-stap uitleg die de gebruiker de bewerking aanleert. Daarna zijn er enkele oefeningen waarbij de leerling op een interactieve manier de matrixbewerkingen inoefent, en krijgt dan ook feedback. Er worden ook manieren voorzien om de gebruiker een hint te geven bij moeilijkheden.

Ten slotte zijn er enkele toepassingen waarmee gepoogd wordt de leerlingen de schoonheid van de aangeleerde theorie over matrices te laten inzien. Markovketens worden gebruikt om de evolutie van een bepaald systeem te modelleren. Daarna wordt er nog naar de moderne wereld gegrepen van de bitcoins om een wallet te decrypteren aan de hand van een matrix encryptie.

# Use Case Diagrams

## Uitleg over usecase diagram

Hieronder staan de tekstuele beschrijvingen van de usecases. In het algemeen moet het project theorie over verschillende matrix bewerkingen aanleren aan de gebruiker en moet de gebruiker verschillende oefeningen kunnen oplossen. Ook bevat het project 2 toepassingen op matrices om de gebruiker het nut van matrices te laten inzien. Het aantal toepassingen kan afhankelijk van de vorderingen van het project nog uitgebreid worden.

## Use case 1: Matrixvermenigvuldiging aanleren

* Naam:
  + Matrix aanleren
* Doelstelling:
  + De gebruiker kan via een aantal stappen een matrix oplossen
* Actoren:
  + Gebruiker
* Precondities
  + Gebruiker moet begrip matrices verstaan
  + Gebruiker moet kunnen vermenigvuldigen en optellen
* Postcondities
  + Begrip matrices bestaan
  + Kennis matrixvermenigvuldiging en voorwaarden
* Successcenario
  + Gebruiker krijgt de algemene uitleg
  + Stap voor stap uitleggen
  + Gebruiker kan naar volgende stap door op knop te drukken
  + Na laatste stap verschijnt knop naar volgende oefeningen
* Alternatief scenario
  + Gebruiker klikt op vorige stap knop om vorige stap te herbekijken

## Use case 2: Interactieve matrixvermenigvuldigingen

* Naam:
  + Interactieve matrixvermenigvuldigingen
* Doelstelling:
  + Gebruiker kan een aantal matrixvermenigvuldigingen uitvoeren
* Actoren:
  + Gebruiker
* Precondities
  + Tutorial matrix aanleren uitgevoerd hebben
* Postcondities
  + Gebruiker moet oefeningen juist oplossen
* Successcenario
  + Gebruiker krijgt 2 matrices
  + Gebruiker vult element per element
  + Per ingevuld element krijg feedback
  + Na correct oplossen verschijnt knop naar volgende oefening
  + Felicitaties aan gebruiker na oplossen alle oefeningen
* Alternatief scenario
  + Bij foute invoer hintknop verschijnt
  + Bij aanklikken hintknop verschijnt de hint

## Use case 3: Determinanten

* Naam:
  + Determinant uitleg volgen en oefeningen oplossen
* Doelstelling:
  + Gebruiker heeft kennis over determinanten en kan ze uitrekenen
* Actoren:
  + Gebruiker
* Precondities
  + Basiskennis matrices
* Postcondities
  + Gebruiker heeft kennis over determinanten en kan ze uitrekenen
* Successcenario
  + Algemene definitie determinant
  + Stap per stap bewerkingen uitleggen
  + Gebruiker leest de extra uitleg stap per stap
  + Gebruiker klikt op next knop om naar volgende stap te gaan
  + Gebruiker klikt op voltooien
  + Gebruiker krijgt een matrix
  + Gebruiker kiest juiste oplossing uit de gegeven oplossingen
  + Gebruiker krijg feedback
  + Gebruiker krijgt 2 analoge oefeningen
  + Felicitaties aan gebruiker na oplossen alle oefeningen
* Alternatief scenario
  + Bij verkeerde keuze hintknop verschijnt
  + Bij aanklikken hintknop verschijnt de hint

## Use case 4: Transponeren

* Naam:
  + Uitleg matrix transponeren
* Doelstelling:
  + Gebruiker weet hoe hij matrices moet transponeren
* Actoren:
  + Gebruiker
* Precondities
  + Basiskennis matrices
* Postcondities
  + Gebruiker weet hoe hij matrices moet transponeren
* Successcenario
  + Algemene definitie over transponeren van matrices
  + Stap per stap animatie tonen
  + Gebruiker bekijkt de animatie en de extra uitleg stap per stap
  + Gebruiker klikt op next knop om naar volgende stap te gaan
  + Gebruiker krijgt matrix
  + Gebruiker stelt zelf de getransponeerde
  + gebruiker krijgt feedback
  + Gebruiker krijgt 2 analoge oefeningen
  + Felicitaties aan gebruiker na oplossen alle oefeningen
* Alternatief scenario
  + Bij verkeerde keuze hintknop verschijnt
  + Bij aanklikken hintknop verschijnt

## Use case 5: Inverse

* Naam:
  + Inverse uitleg
* Doelstelling:
  + Gebruiker weet hoe hij matrices moet inverteren
* Actoren:
  + Gebruiker
* Precondities
  + Determinanten kunnen uitrekenen
* Postcondities
  + Gebruiker weet hoe hij matrices moet inverteren
* Successcenario
  + Algemene definitie over inverteren van matrices
  + Stap per stap bewerkingen uitleggen (adjunct berekenen)
  + Gebruiker klikt op next knop om naar volgende stap te gaan
  + Gebruiker klikt op voltooien
  + Gebruiker krijgt matrix
  + Gebruiker stelt zelf de inverse op
  + Gebruiker krijgt feedback
  + Gebruiker krijgt 2 analoge oefeningen
  + Felicitaties aan gebruiker na oplossen alle oefeningen
* Alternatief scenario
  + Bij verkeerde keuze hintknop verschijnt
  + Bij aanklikken hintknop verschijnt

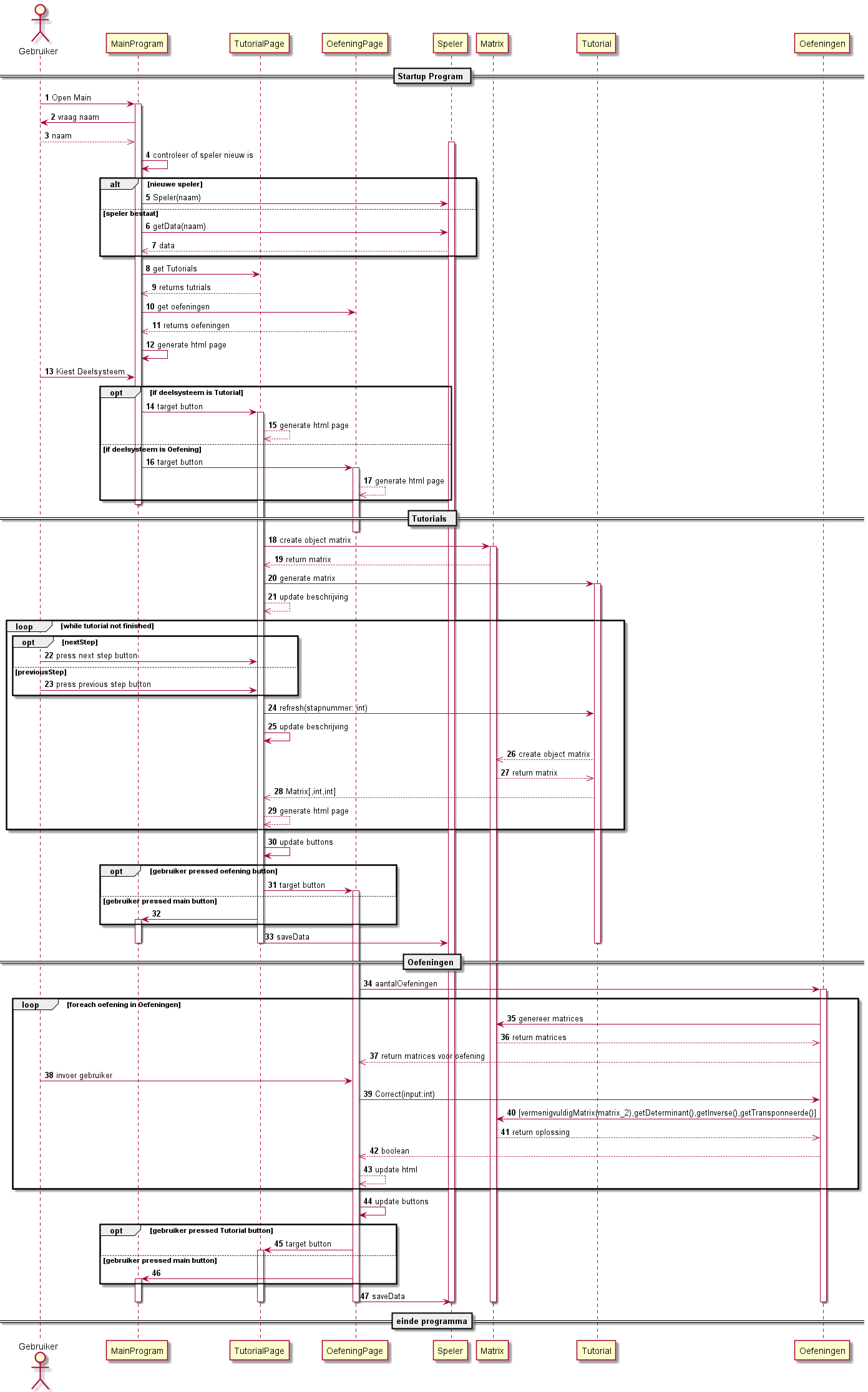
## Use case 6: Toepassing 1: ‘Decrypt the wallet’ (Hill cipher encryptie)

* Naam:
  + Toepassing: Decrypt the wallet
* Doelstelling:
  + Gebruiker kan opgestelde wachtwoord van de wallet ontcijferen
* Actoren:
  + Gebruiker
* Precondities
  + Gebruiker heeft alle tutorials uitgevoerd en begrepen
* Postcondities
  + Gebruiker kan letters aan de hand van matrix encrypteren en decrypteren (Hill cipher encryptie toepassen)
* Successcenario
  + Gebruiker krijgt verhaal over bitcoinwallet
  + Gebruiker krijgt uitleg over Hill cipher encryptie
  + Gebruiker krijgt versleutelingsmatrix
  + Gebruiker inverteert de versleutelingsmatrix
  + Gebruiker decrypteerd de key
  + Gebruiker geeft de juiste key in
  + Gebruiker krijgt bitcoins
* Alternatief scenario
  + Gebruiker geeft verkeerde key
  + Systeem wijst aan in welke matrix de fout zit

Use case 7: Toepassing 2: Markov ketens

* Naam:
  + Toepassing in markov ketens
* Doelstelling:
  + Gebruiker begrijpt het belang van matrices in markov ketens
* Actoren:
  + Gebruiker
* Precondities
  + Gebruiker heeft alle tutorials uitgevoerd en begrepen
* Postcondities
  + Gebruiker begrijpt het belang van matrices in markov ketens en kan uit een tekst overgangsmatrix opstellen en gebruiken
* Successcenario
  + Uitleg over markov ketens
  + Voorbeelden van toepassingen (overgangsmatrix^50=>na 50 jaar)
  + Gebruiker krijgt een tekst
  + Gebruiker stelt overgangsmatrix op
  + Gebruiker krijgt feedback
  + Gebruiker kan interactief evolutie van systeem bekijken
  + Gebruiker kan invullen hoeveel evolutiestappen het systeem vordert
* Alternatief scenario
  + Bij verkeerde keuze hintknop verschijnt
  + Bij aanklikken hintknop verschijnt

# Sequentiediagram



Figuur 1: Sequentiediagram

## Uitleg over sequentiediagram:

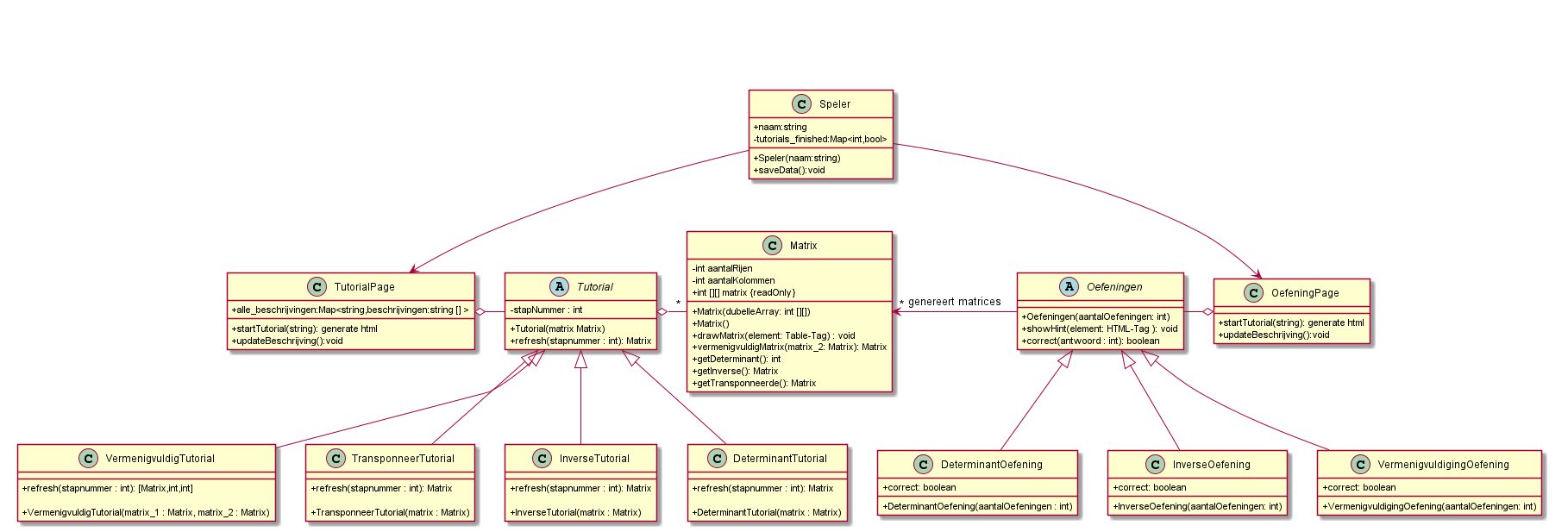
Het project begint met een aanvraag van de naam van de gebruiker. Hierbij gaan we na of de gebruiker eerder heeft deelgenomen aan het programma. Aan de hand van LocalStorage zullen we een map van gebruikers bijhouden welke gelinkt zijn aan het laatste deelsysteem dat ze hebben voltooid (dit aan de hand van een index). Wanneer we een nieuwe gebruiker tegenkomen zullen we de map aanpassen en de gebruiker verder sturen naar de eerste tutorial. Indien de gebruiker wel al heeft deelgenomen aan het programma, kan hij de keuze maken om een specifiek deelsysteem te herbekijken of verder te gaan naar de oefeningen.

In ons sequentiediagram hebben we een algemene sequentie aangemaakt die alle verschillende tutorials in een loop presenteert. De tutorials zullen allen een “next” en “previous” knop bevatten om systematisch door de stappen te verplaatsen. Nadat de gebruiker een gekozen tutorial heeft afgerond, krijgt hij de keuze om de oefeningen te maken omtrent de bekeken tutorial of kan hij terugkeren naar het hoofdmenu. Na elke afgeronde tutorial wordt de data van de gebruiker geüpdatet.

Voor het oefeningen gedeelte geven we allereerst een aantal oefeningen mee aan de constructor van Oefeningen. Dit object zal de matrices kunnen aanspreken en bedienen. Nadat de Oefeningen zijn aangemaakt openen we een for-each lus waarin we beginnen met een matrix te genereren. Hierna kan de gebruiker de oefening beginnen oplossen en zijn gevonden uitkomst invoeren. Deze invoer wordt gecontroleerd. Zolang de invoer van de gebruiker niet correct is, moet hij opnieuw proberen (dit kan nog gecomplementeerd worden met een meegegeven tip). Wanneer de gebruiker gereed is met de reeks gekregen oefeningen, krijgt hij de keuze om terug te keren naar hoofdmenu of naar de volgende tutorial te verwezen worden. Hier zal ook zijn data bijgewerkt worden in de LocalStorage.

Dit is het tijdelijke verloop van het programma. In de loop van de komende weken willen we dit vervolledigen met de twee uitgewerkte toepassingen uit Use Case 6 en Use Case 7.

# Klassendiagram



Figuur 2: Klassendiagram

## Uitleg over klassendiagram

De klasse Speler houdt de naam van de huidige gebruiker en de oefeningen/tutorials die hij afgewerkt heeft. De klasse heeft een constructor waar de ingegeven naam van de huidige gebruiker meegegeven wordt, die gaat dan kijken of de speler al bestaat of niet en zo nodig een nieuwe aanmaken. De methode saveData() gaat de data van de huidige gebruiker doorgeven aan de LocalStorage. Die data wordt verkregen uit de klassen TutorialPage en OefeningPage.

De klasse TutorialPage houdt alle beschrijvingen van alle tutorials bij in de Map alle\_beschrijvingen. De klasse heeft een methode startTutorial die als parameter de naam van de specifieke tutorial meekrijgt en de basis html genereert. De klasse heeft ook een methode updateBeschrijving die de beschrijving zal updaten door in Tutorial naar het stapnummer te kijken en in de map alle\_beschrijvingen te kijken.

De abstracte klasse Tutorial heeft een attribuut stapNummer waarin hij het stapnummer bijhoudt. De klasse heeft een constructor waar een matrix aan gegeven wordt die als basis dient voor de tutorial. De klasse heeft ook een methode refresh(stapnummer:int) waar het stapnummer aan meegegeven wordt en die de matrix die bij die stap hoort teruggeeft.

De klassen VermenigvuldigTutorial, TransponeerTutorial, InverseTutorial en DeterminantTutorial zijn afgeleide klassen van Tutorial, deze klassen overschrijven de methode refresh omdat voor bv VermenigvuldigTutorial er meerdere matrices zijn. Ze hebben elk ook nog een constructor die met meegegeven matrices de tutorials aanmaakt.

De klasse OefeningPage is zoals TutorialPage maar houdt geen beschrijvingen bij.

De abstracte klasse Oefeningen heeft een constructor die het aantal oefeningen meekrijgt en die de matrices aanmaakt die nodig zijn. De klasse heeft ook een methode showHint die de hint zal tonen. En een methode correct die het antwoord van de gebruiker meekrijgt en zal controleren of het juist is en dat zal teruggeven met een boolean.

De klassen DeterminantOefening, InverseOefening en VermenigvuldigingOefening zullen elk de methode correct overschrijven doordat telkens andere methodes moeten opgeroepen worden van de klasse Matrix om de oplossing te vinden. Ze hebben ook nog elk hun constructor die ook telkens het aantal oefeningen meekrijgt.

De klasse Matrix houdt alles bij van een bepaalde matrix, dus het aantal rijen en kolommen en de matrix zelf in een dubbele array vorm. De klasse heeft dan ook een constructor om een object aan te maken die de dubbelle array meekrijgt of niks meekrijgt. Als de constructor niets meekrijgt zal een random matrix gemaakt worden. De klasse heeft ook een methode drawMatrix die de matrix in de meegegeven table-tag zal steken. Daarnaast heeft de klasse ook nog een paar andere methodes die verschillende bewerkingen gaan uitvoeren om gemakkelijk de antwoorden op de oefeningen te vinden.