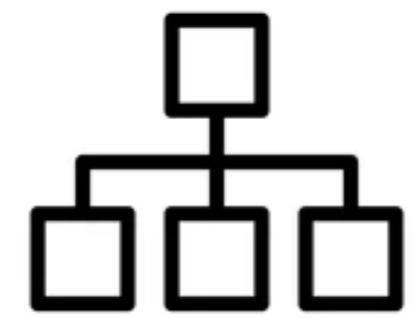


Modelling & Databases

Les 10:

UML: class diagram & use case diagram



Onderwerpen les:

- Wat is UML?
- UML software
- Class diagram
- Use case diagram

Leerdoelen

- ✓ Aan het einde van de dag weet je wat UML is en ken je de twee type diagrammen
- ✓ Aan het einde van de dag kan diagrammen tekenen in Lucidchart
- ✓ Aan het einde van de dag weet je wat Use case en class diagrammen zijn
- ✓ Aan het einde van de dag kan je een class diagram voor een simpele app maken

UML

Wat is UML?

Unified Modeling Language

UML staat voor Unified Modeling Language en is een **industriestandaard grafische taal** dat gebruikt wordt voor **specificeren, visualiseren, construeren en documenteren** van de artefacten van softwaresystemen, evenals voor bedrijfsmodellering en andere niet-softwaresystemen.

UML wordt gebruikt om het complexe proces van softwareontwerp te **vereenvoudigen**. Er zijn veel soorten UML-diagrammen om de vereisten en specificaties van software duidelijker te communiceren.

UML gebruikt **diagrammen om softwareoplossingen te visualiseren** en wordt vaak gebruikt bij softwareontwikkeling.

UML

Voordelen UML

Een foto zegt meer dan duizend woorden. Daarom zijn UML-diagrammen gemaakt. Bij ontwikkeling is het belangrijk om de software-architectuur te communiceren op een manier die gemakkelijk te begrijpen is voor iedereen die bij het project betrokken zijn. Omdat software steeds belangrijker werd voor bedrijven, was er behoefte aan een standaard manier om **softwarevereisten te communiceren**.

Meest gebruikt en flexibel

UML is nu een industiestandaard en wordt door veel ontwikkelaars goed begrepen. UML is niet specifiek voor een type technologie of codeertaal.

Effectieve en duidelijke manier van communiceren en breed inzetbaar

UML is niet alleen bedoeld om OOP-software-engineering te communiceren, maar ook om gedrag, bedrijfsprocessen en meer te structureren.

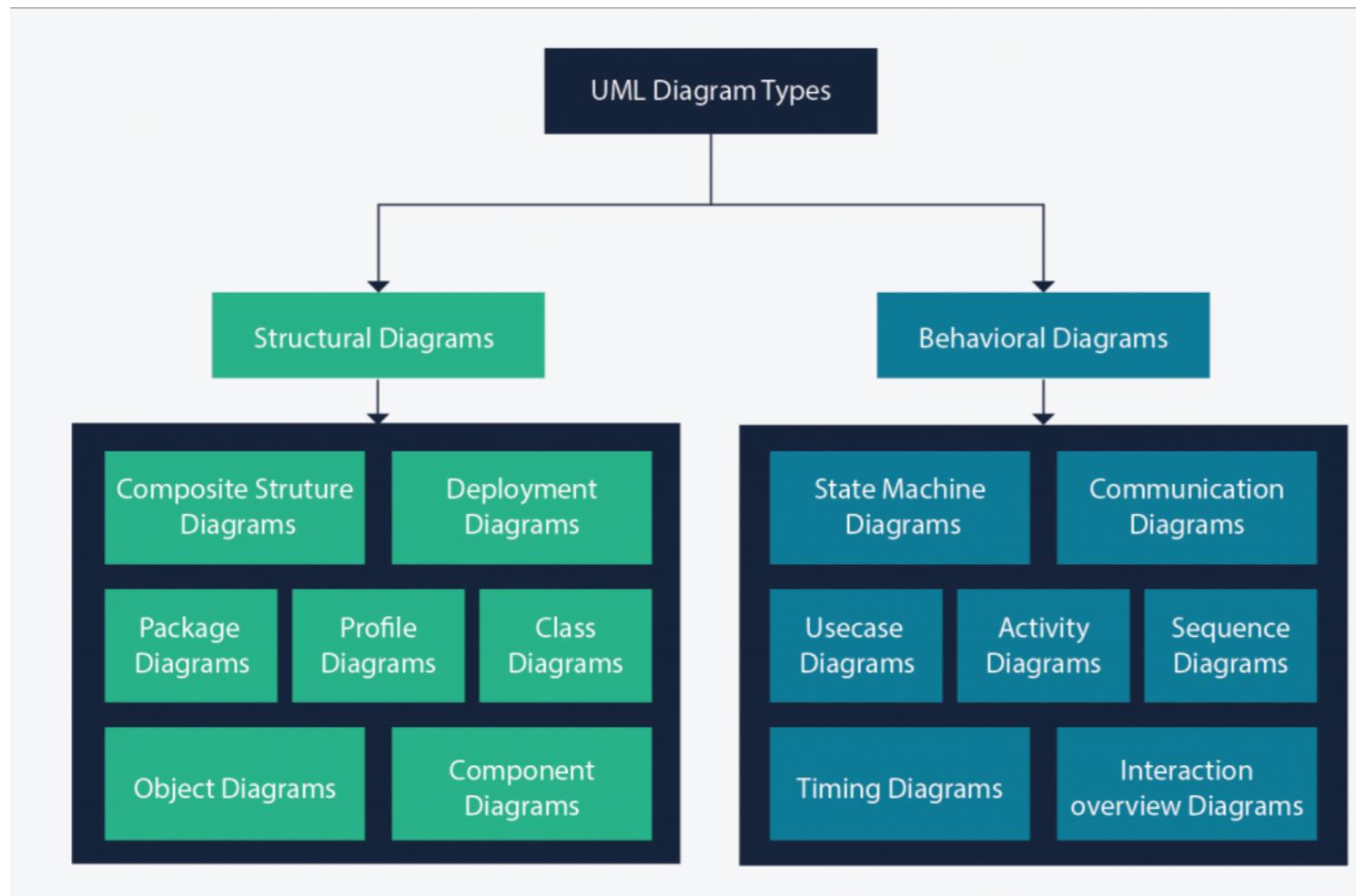
Je hoeft niet alles over UML te weten om het te gebruiken

UML heeft veel verschillende soorten diagrammen voor verschillend gebruik. Je hoeft ze niet allemaal te kennen om bijvoorbeeld een klassendiagram te gebruiken

UML

Type diagrammen

UML heeft in totaal **14 verschillende** soorten diagrammen. Deze hebben 2 hoofdcategorieën; **structuurdiagrammen en gedragsdiagrammen**.



Structuurdiagrammen tonen de dingen in het gemodelleerde systeem. In een meer technische term tonen ze verschillende **objecten in een systeem**.

Gedragsdiagrammen laten zien wat er in een systeem zou moeten gebeuren. Ze beschrijven **hoe de objecten met elkaar omgaan** om een functionerend systeem te creëren.

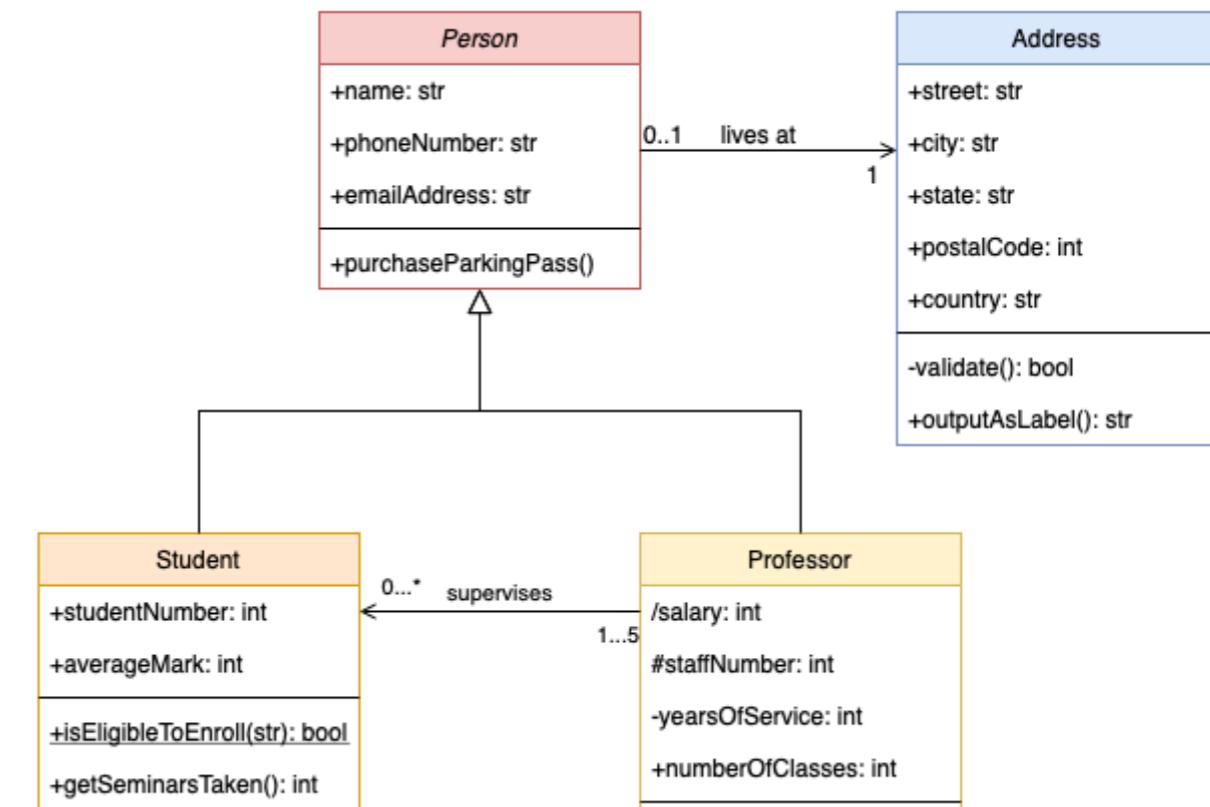
UML

Type diagrammen

De twee diagammen die we vandaag gaan bespreken: Class diagram, Use case diagram. In het kort:

Class diagram

Dit is het meest populaire structuurdiagram. Een klassediagram is een geweldig hulpmiddel om te gebruiken voor het communiceren van objectgeoriënteerde programmeeroplossingen. Het toont de klassen in een systeem, kenmerken en bewerkingen van elke klasse en de relatie tussen elke klasse.

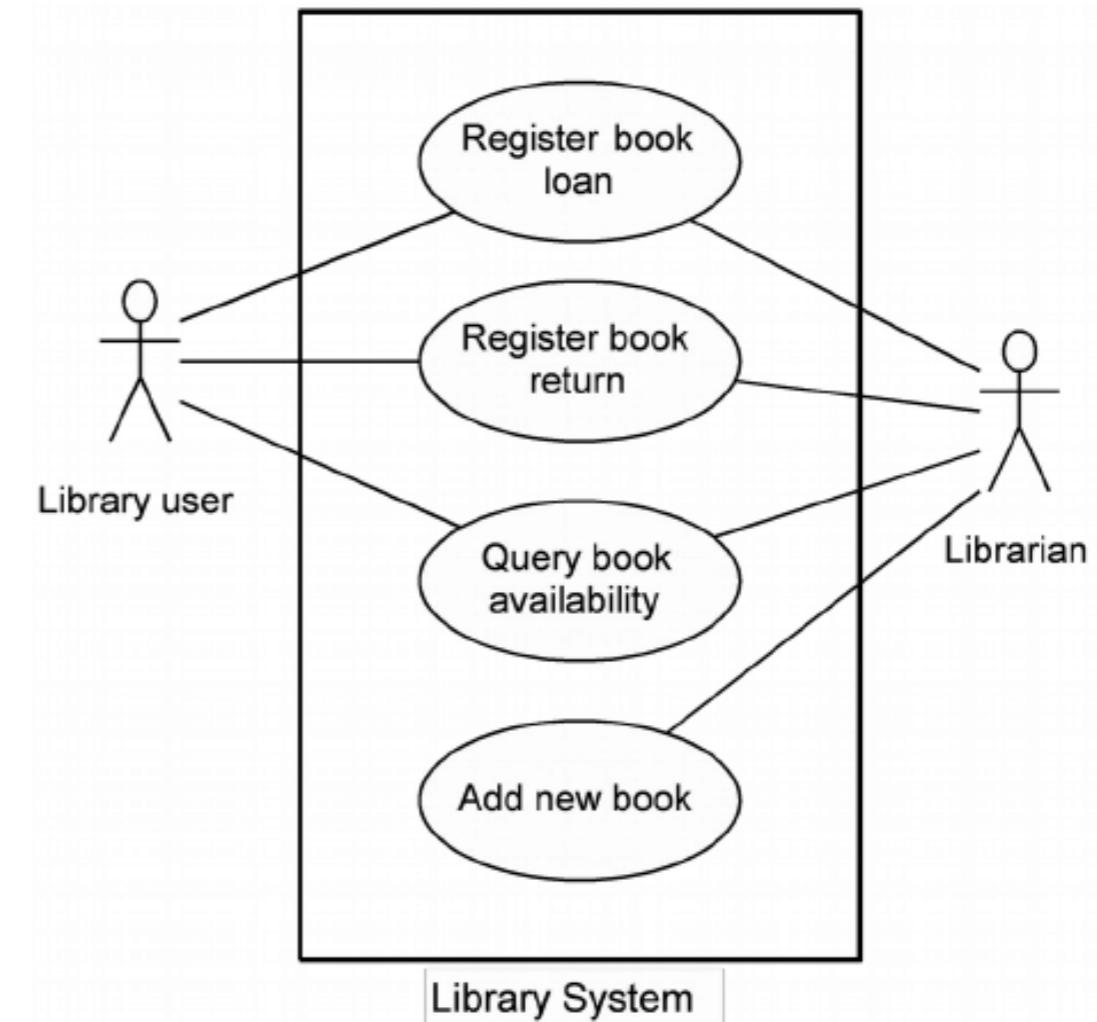


UML

Type diagrammen

Use case diagram

Dit is het meest bekende gedrags UML-diagram. Het use case-diagram geeft een grafisch overzicht van de actoren die betrokken zijn bij een systeem, de verschillende functies die deze actoren nodig hebben en hoe deze verschillende functies op elkaar inwerken.



UML

Bekijk ook andere UML diagrammen

Tijdens je reis als ontwikkelaar heb je misschien verschillende soorten UML-diagrammen nodig, of je werkt aan een project waarbij ze verschillende UML-diagrammen gebruiken om softwareoplossingen te communiceren. Het is dus goed om te weten welke soorten er zijn en waarvoor ze worden gebruikt. Bezoek de onderstaande link en lees kort over alle soorten UML-diagrammen.

| <https://creately.com/blog/diagrams/uml-diagram-types-examples/>

extra bronnen

| <https://www.lucidchart.com/pages/what-is-UML-unified-modeling-language>

| <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/>

| <https://www.youtube.com/watch?v=7BSBWhGFmJ0>

| <https://www.youtube.com/watch?v=WnMQ8HlmeXc>

UML

Software

Er zijn veel goede opties als het gaat om software voor het maken van UML-diagrammen. De meest populaire zijn:

- **Lucidchart**
- **Gleek**
- **Diagrams**
- **Cacoo**
- **Gliffy**
- **EndrawMax**
- **Microsoft Visio**

UML

Software

Een van de meest populaire keuzes is Lucidchart, en ook de keuze voor deze opleiding.

Maak een gratis account

| <https://lucid.app/pricing/lucidchart?referer=https%3A%2F%2Flucid.app%2Fusers%2Flogin#/pricing/chart>

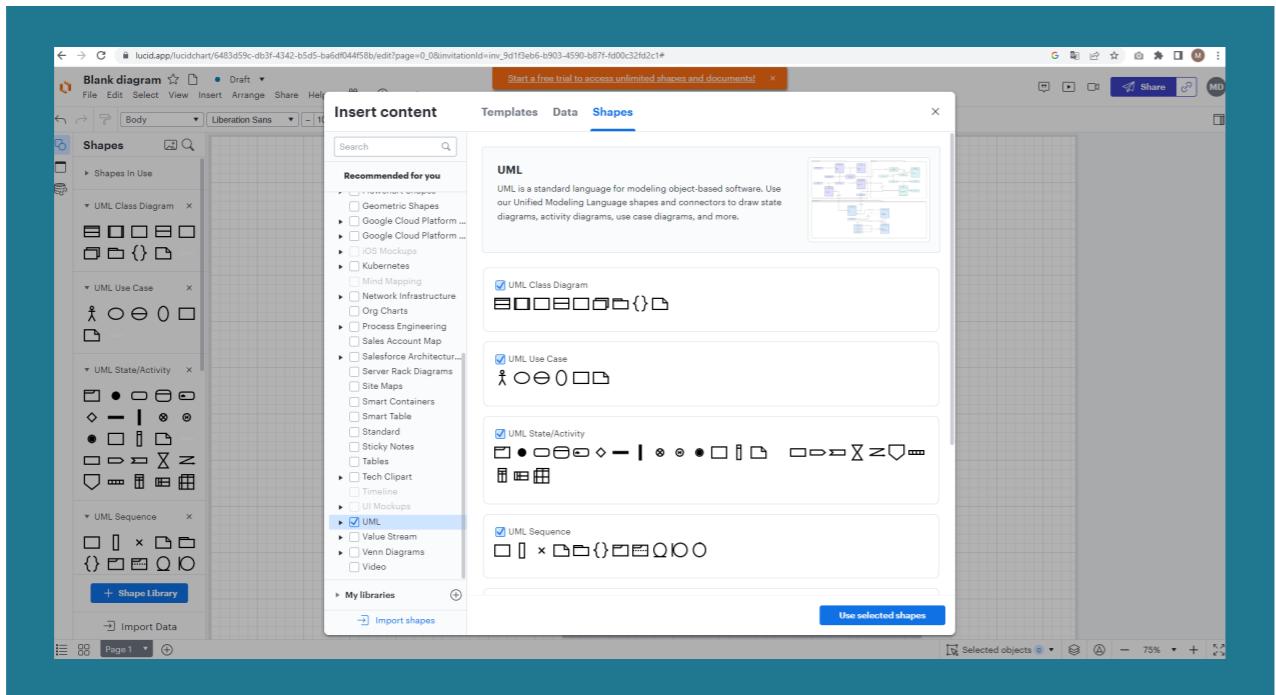


In dit gratis account kun je 3 bewerkbare documenten maken.

UML

Software

Lucidchart interface



- 1 Login
- 2 Open (new blank) document
- 3 Open shape library -> Select UML -> Use selected shapes
- 4 Drag and drop wat je nodig hebt
- 5 Dubbelklik tekst om te editen

Class diagram

Wat is een class diagram?

Een UML-klassendiagram is een **statisch model van een objectgeoriënteerd systeem** dat objecten definieert op basis van hun **klassen, attributen en functies**. Het is een geweldige manier om objectgeoriënteerd systeem visueel te maken voor andere ontwikkelaars en belanghebbenden zonder naar de code te hoeven kijken.

Je kunt een UML-klassendiagram zien als een "recept" voor een objectgeoriënteerd systeem. Net zoals een recept ingrediënten, hoeveelheden en richtingen bevat, legt een klassendiagram het systeem uit in termen van welke objectklassen erbij betrokken zijn, welke attributen ze bevatten en welke bewerkingen ze moeten uitvoeren.

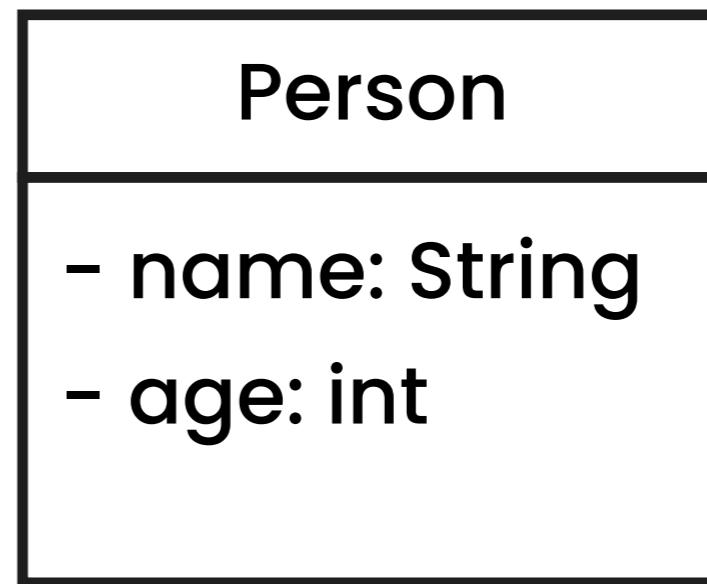
De klassen en hun **namen in dit diagram zullen hetzelfde zijn als in de broncode**, dit geldt ook voor hun attributen en hun methoden.

Class diagram

Classes in een class diagram

Laten we eens kijken naar een eenvoudige klasse Persoon, met 2 attributen.

```
Person.java ×  
1 ▼ public class Person {  
2     private String name;  
3     private int age;  
4 }
```



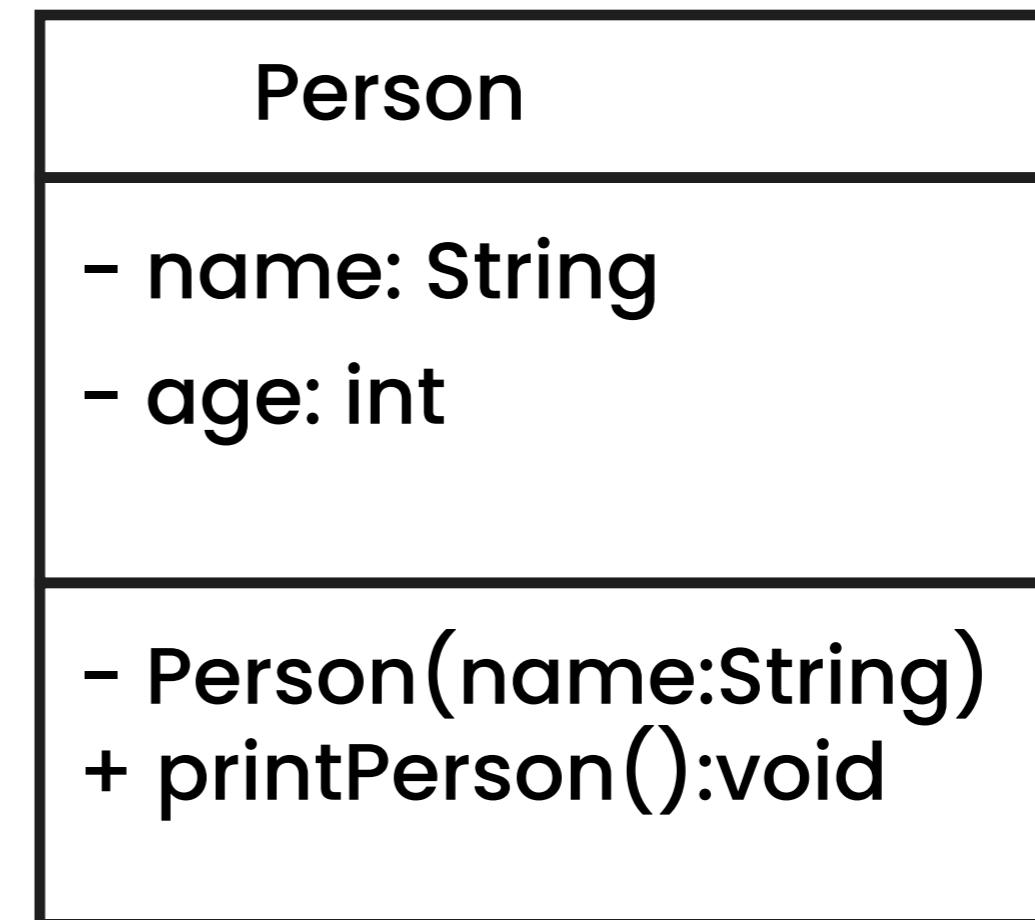
- In een klassendiagram wordt een klasse weergegeven als een rechthoek met de naam van de klasse in de bovenste balk.
- De balk eronder is gereserveerd voor de attributen. Elk attribuut krijgt zijn eigen rij.
- Onder deze balk staat een balk voor de methodes.

Class diagram

Constructors en andere methods

De constructor en alle andere methoden worden in de derde balk geplaatst.

```
Person.java ×  
1▼public class Person {  
2    private String name;  
3    private int age;  
4  
5▼    private Person(String name) {  
6        this.name = name;  
7        this.age = 42;  
8    }  
9  
10▼   public void printPerson() {  
11        System.out.println(this.name + " age: " + this.age);  
12    }  
13}
```

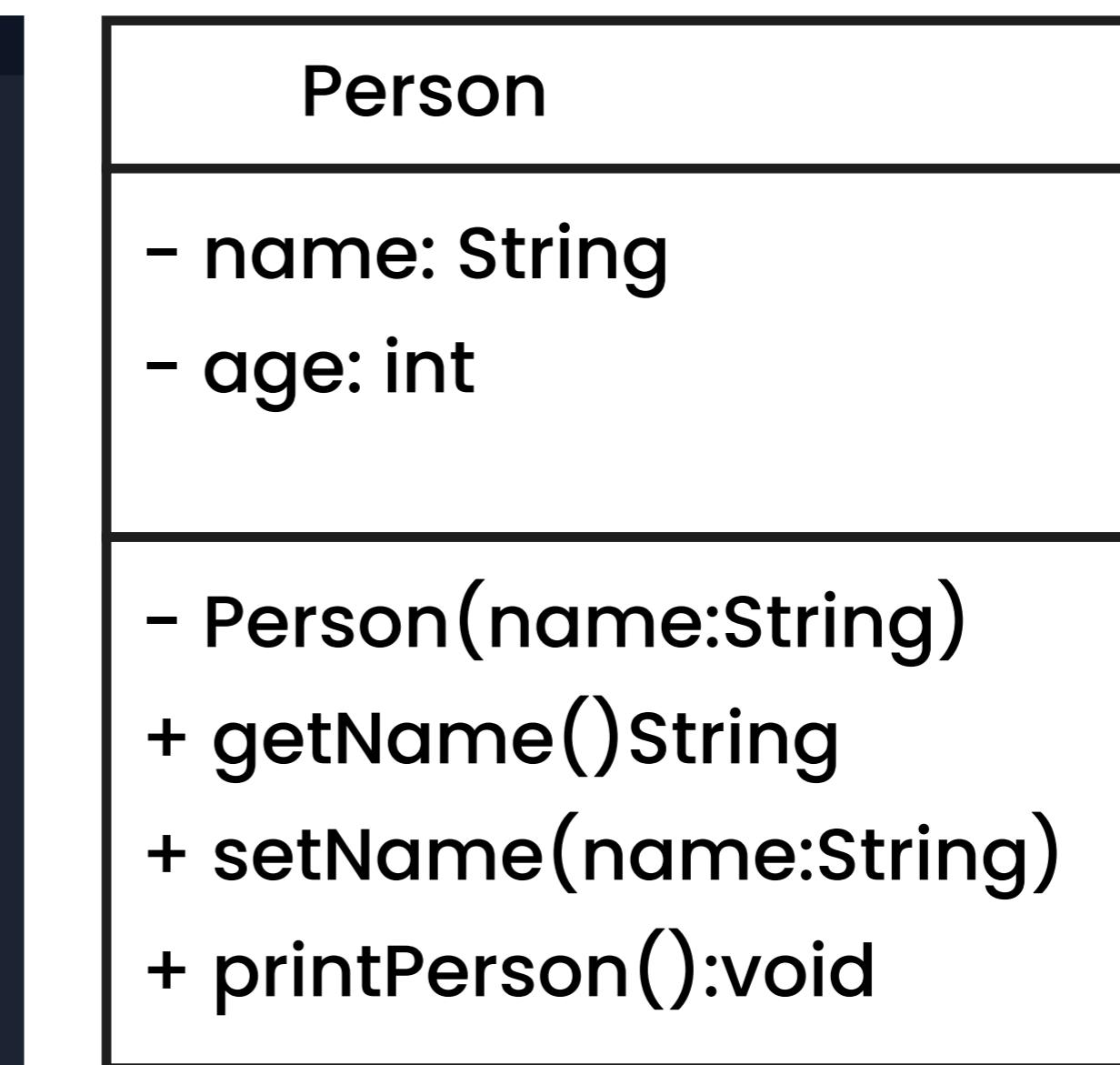


We laten zien wat de methode teruggeeft

Class diagram

Getters en setters

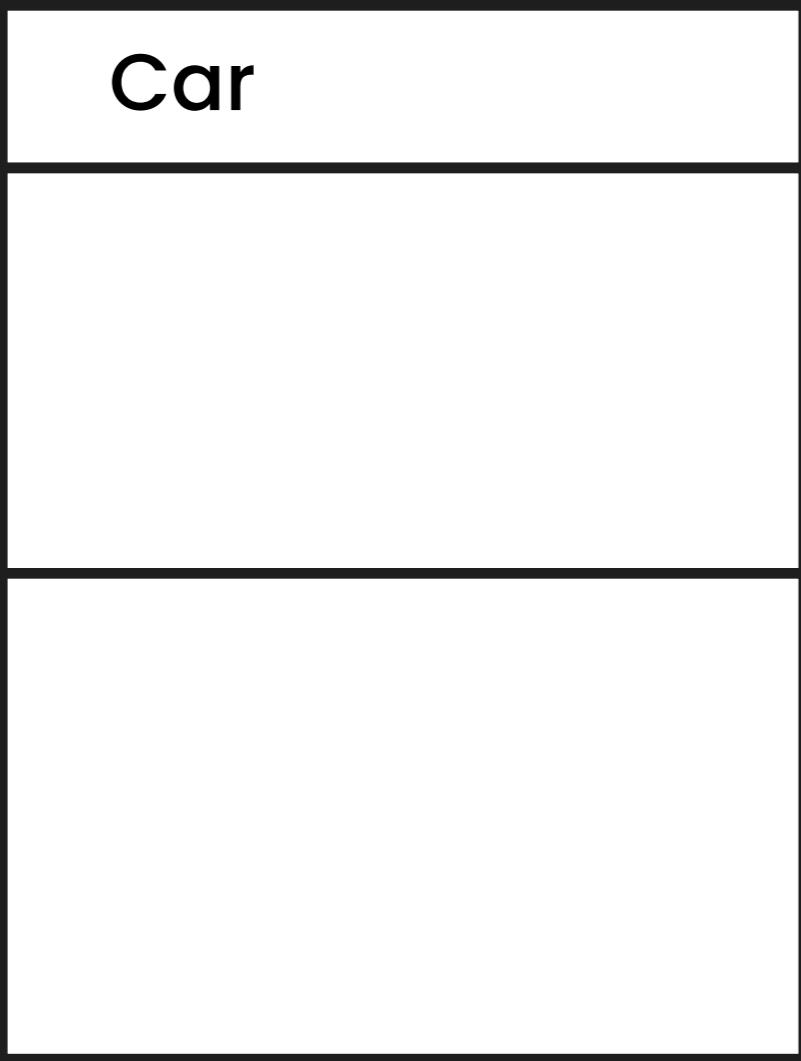
```
Person.java ×  
1▼ public class Person {  
2    private String name;  
3    private int age;  
4  
5▼  private Person(String name) {  
6      this.name = name;  
7      this.age = 42;  
8  }  
9  
10▼ public String getName() {  
11    return this.name;  
12  }  
13  
14▼ public void setName(String newName) {  
15    this.name = newName;  
16  }  
17  
18▼ public void printPerson() {  
19    System.out.println(this.name + " age: " + this.age);  
20  }  
21 }
```



Class diagram

Opdracht

```
Car.java ×
1▼public class Car {
2    private String brand;
3    private int mileage;
4
5▼ private Car(String newName, int newMileage) {
6    this.brand = newName;
7    this.mileage = newMileage;
8 }
9
10▼ public void driveCar(int miles) {
11    System.out.println(this.brand + " is driving: " + miles + " miles");
12    this.mileage+=miles;
13 }
14 }
```

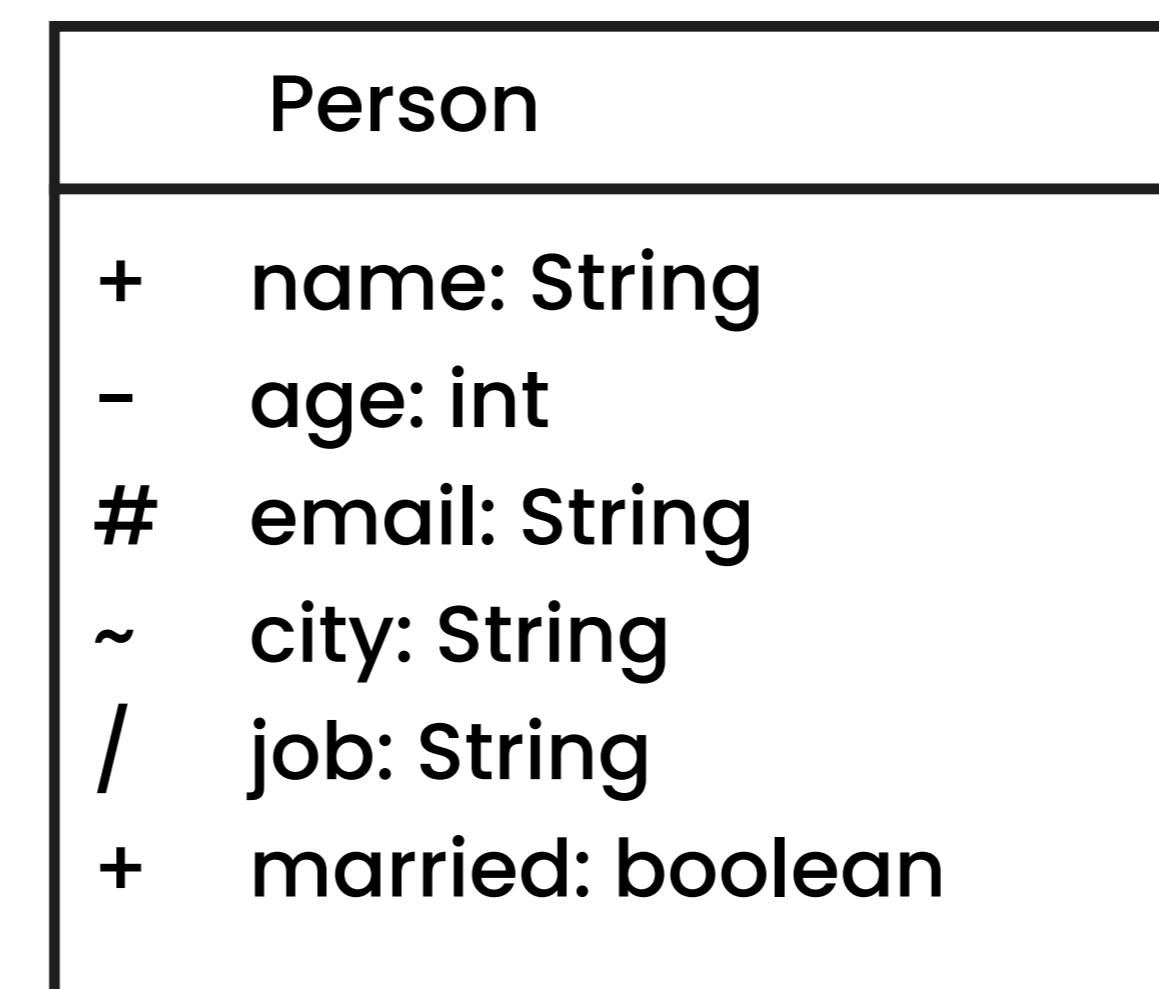


Class diagram

Acces modifiers

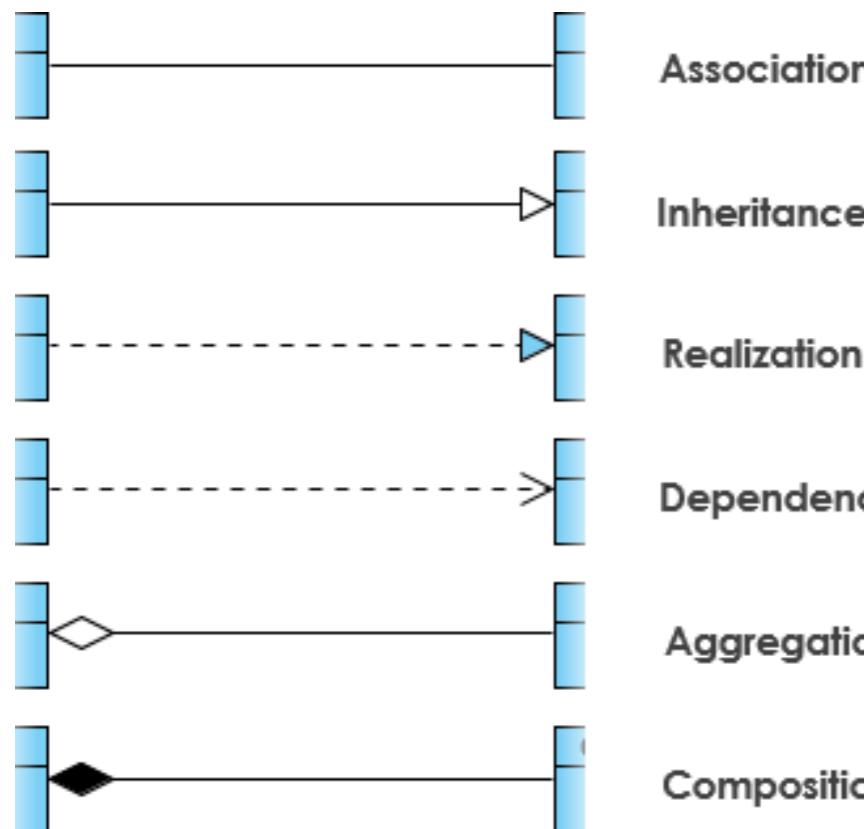
Alle klassen hebben verschillende **toegangsniveaus**, afhankelijk van de toegangsmodificator (zichtbaarheid). Dit zijn de toegangsniveaus met de bijbehorende **symbolen**:

- **Public (+)**
- **Private (-)**
- **Protected (#)**
- **Package (~)**
- **Derived (/)**
- **Static (underlined)**



Class diagram

Relaties tussen classes



Association

Is een verbinding. Er zijn twee vormen van associatie: compositie en aggregatie.

Inheritance

Relatie tussen een meer algemene classifier en een meer specifieke classifier. "is een" (een kat is een dier) (een programmeur is een werknemer).

Aggregation

Een bijzonder soort vereniging. "deel van". impliceert een relatie waarin het kind onafhankelijk van de ouder kan bestaan. Voorbeeld: klas (ouder) en leerling (kind). Verwijder de klas en de studenten bestaan nog steeds.

Composition

Een bijzonder soort vereniging. "heeft een". impliceert een relatie waarin het kind niet onafhankelijk van de ouder kan bestaan. Voorbeeld: Huis (ouder) en Kamer (kind). Kamers bestaan niet los van een huis.

Class diagram

Association multiplicity

Veelheid	Kardinaliteit
0..0	Collection must be empty
0..1	No instances or one instance
1..1	Exactly one instance
0..*	Zero or more instances
1..*	At least one instance
5..5	Exactly 5 instances
m..n	At least m but no more than n instances

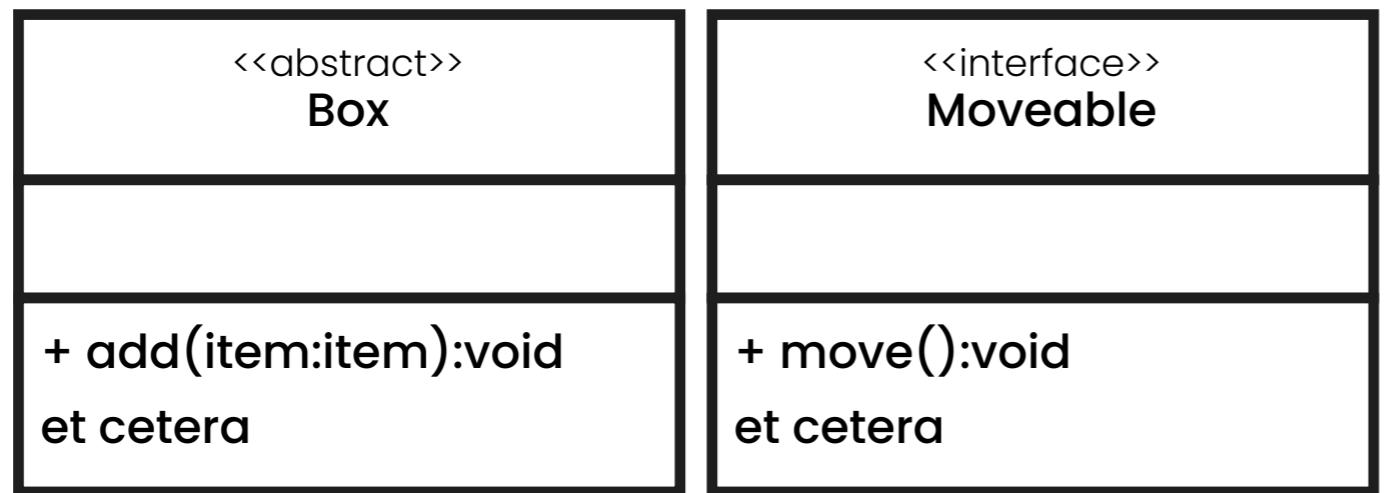
Class diagram

Association multiplicity



Class diagram

Abstract and interface



Class diagram

Extra bronnen

- | <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>
- | <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/>
- | <https://www.youtube.com/watch?v=UI6IqHOVHic>
- | <https://www.youtube.com/watch?v=ao1ESgly2Ws>

Use case diagram

Wat is een class diagram?

Een UML-use-case-diagram is een geweldige manier om het **verwachte gedrag/de verwachte interacties van een gebruiker** met het systeem weer te geven. Het is gebouwd met **symbolen en connectoren**.

In dit diagram zie je use cases, actors, relationships en boundaries van het systeem.

Use case

Dit kan een doel of actie zijn. Bijvoorbeeld in een online boekenwinkel: boeken lenen. of in een winkel: aankoop doen

Actors

De gebruikers die interactie hebben met een systeem. Een actor kan een persoon, een organisatie of een extern systeem zijn dat interactie heeft met uw applicatie of systeem. Het moeten externe objecten zijn die gegevens produceren of consumeren.

Use case diagram

Wat is een class diagram?

Relationships

Een lijn tussen actoren en use cases. Bij complexe diagrammen is het belangrijk om te weten welke actoren bij welke use cases horen. Ook kunnen er relaties zijn tussen verschillende use cases.

Boundaries of a system

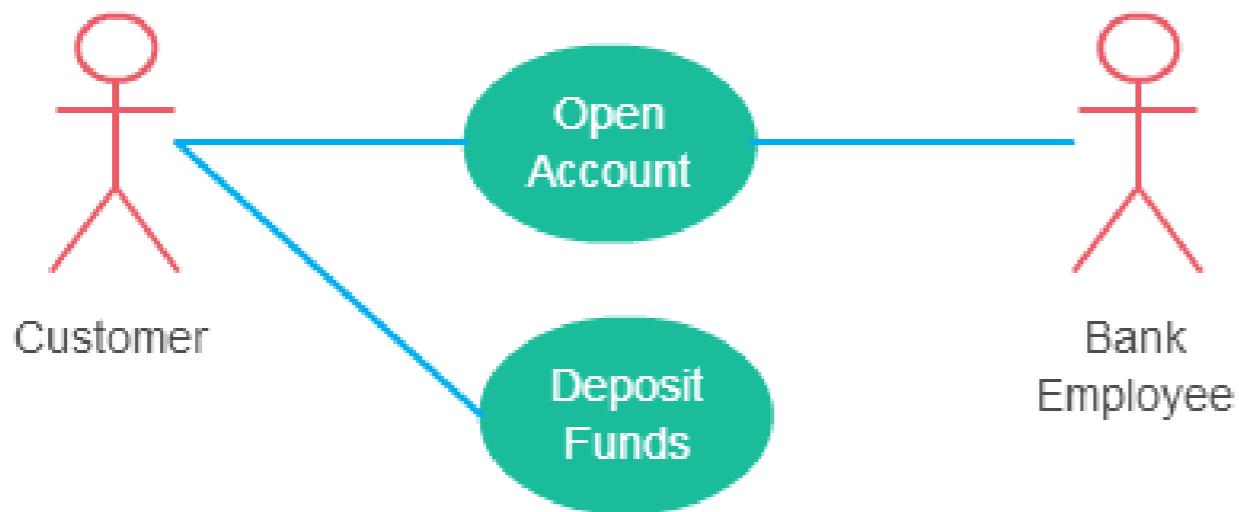
Een vak dat een systeembereik instelt om cases te gebruiken. Alle use-cases buiten de kaders vallen buiten de reikwijdte van dat systeem.

Use case diagram

Relations / communications

Association (Tussen Actor en Use Case)

Dit is in elke use-case. Een actor moet aan ten minste één use-case zijn gekoppeld, maar kan aan meerdere use-cases zijn gekoppeld. Het is ook mogelijk om meerdere actoren aan één use case te koppelen.

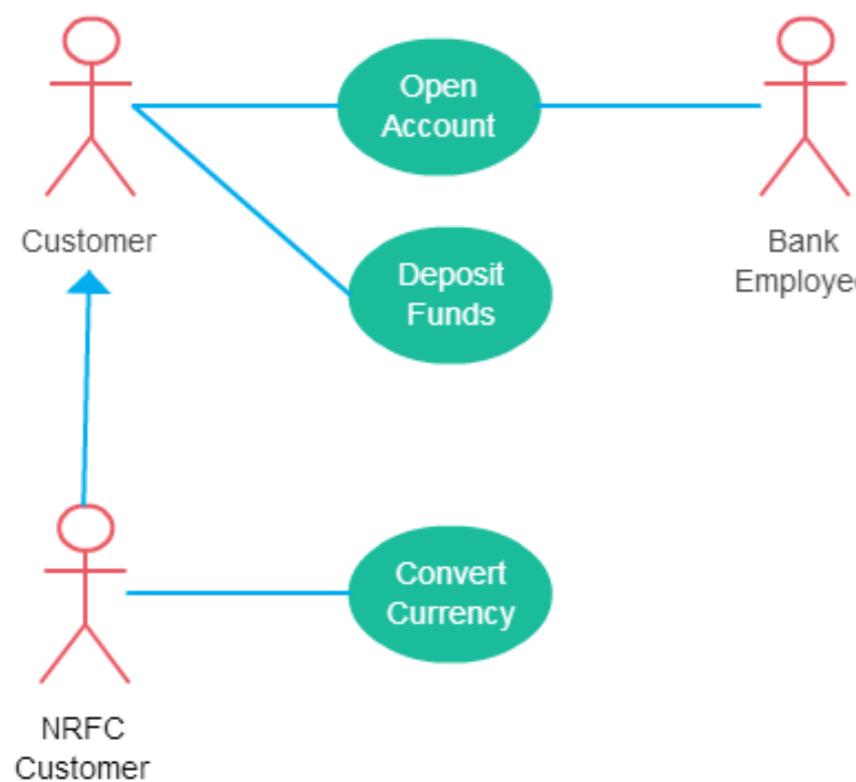


Use case diagram

Relations / communications

Generalisatie van een acteur

Generalisatie van een actor betekent dat de ene actor de rol van de andere actor kan erven. De afstammeling erft alle use cases van de voorouder. De afstammeling heeft een of meer use cases die specifiek zijn voor die rol. Laten we het vorige use case-diagram uitbreiden om de generalisatie van een actor te tonen.

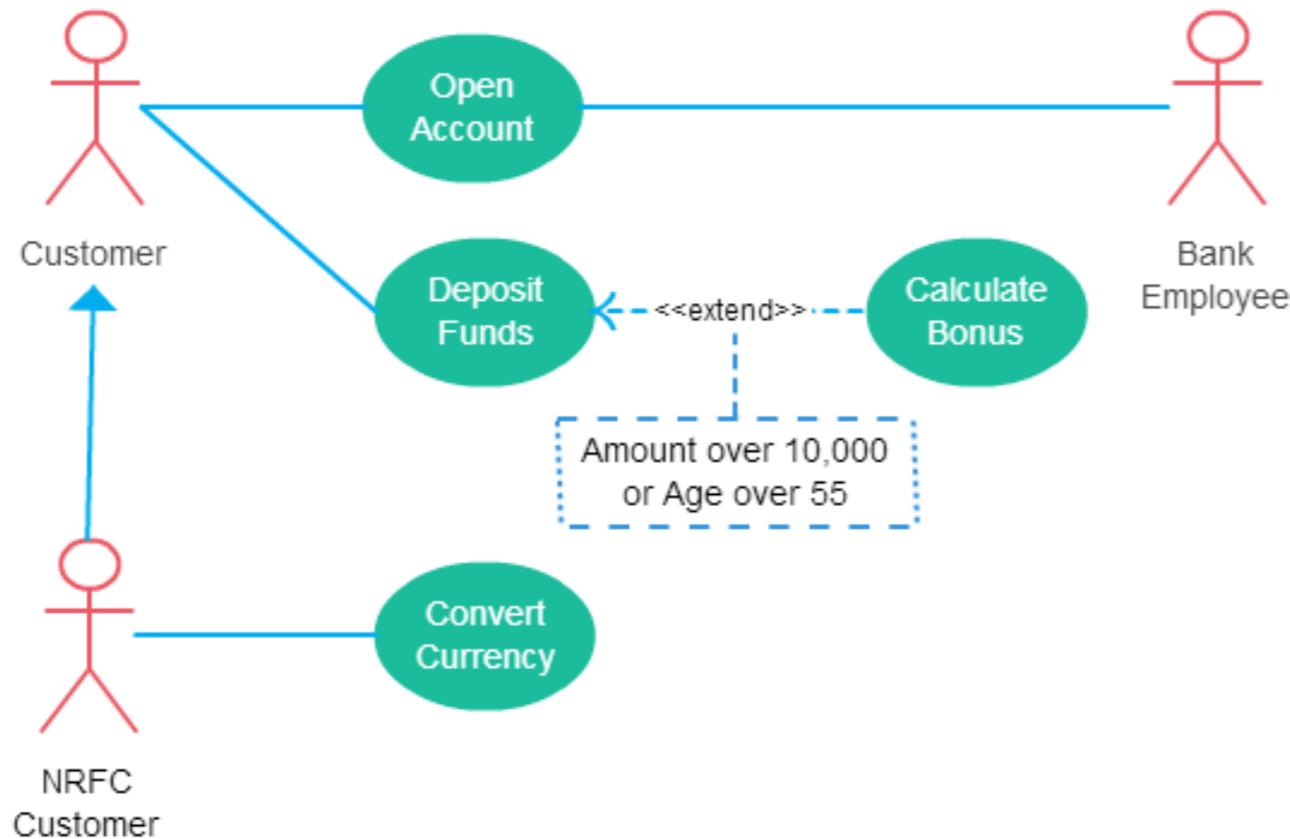


Use case diagram

Relations / communications

Extend (verleng) de relatie tussen twee use cases

Zoals de naam al aangeeft, breidt het de basis use case uit en voegt het meer functionaliteit toe aan het systeem. Hier volgen enkele zaken waarmee u rekening moet houden bij het gebruik van de <<extend>> relatie.



Use case diagram

Relations / communications

Extend (verleng) de relatie tussen twee use cases

De uitbreidende use case is afhankelijk van de uitgebreide (basis) use case. In het onderstaande diagram heeft de use case "Bonus berekenen" weinig zin zonder de use case "Geld storten".

De uitbreidende use-case is meestal optioneel en kan voorwaardelijk worden geactiveerd. In het diagram kunt u zien dat de use case voor uitbreiden alleen wordt geactiveerd voor stortingen van meer dan 10.000 of wanneer de leeftijd ouder is dan 55.

De uitgebreide (basis) use case moet op zichzelf zinvol zijn. Dit betekent dat het onafhankelijk moet zijn en niet afhankelijk mag zijn van het gedrag van de uitbreidende use-case.

Use case diagram

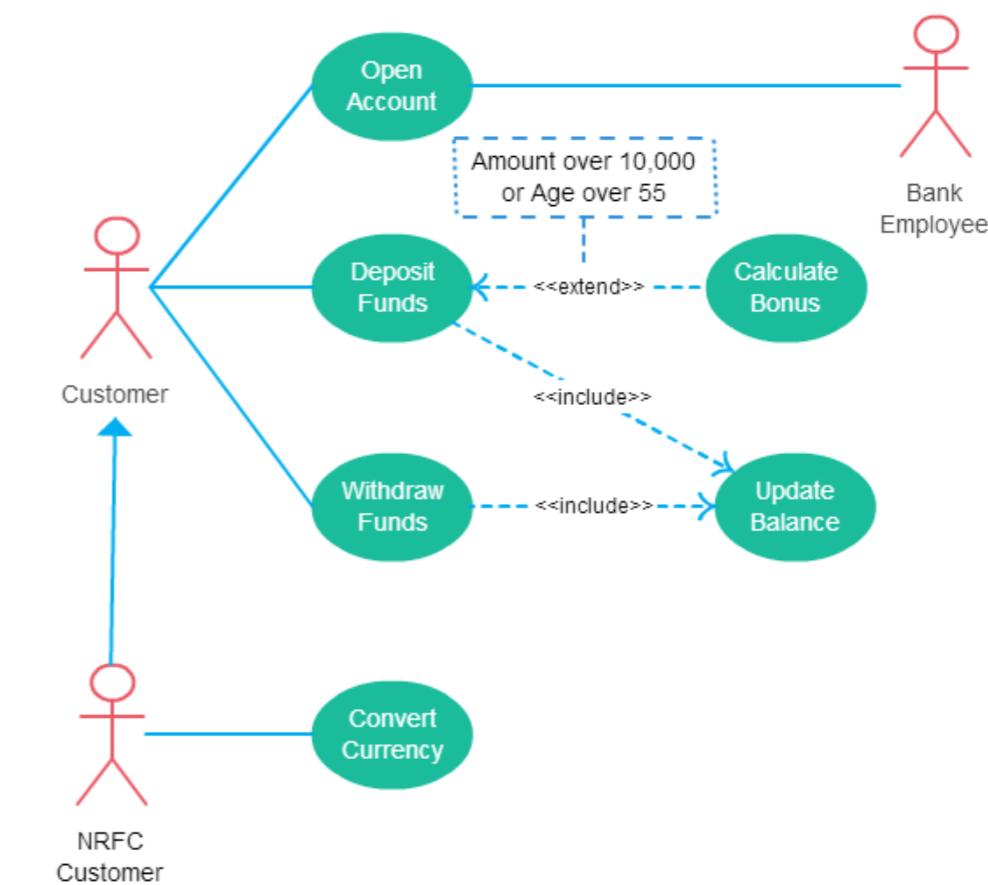
Relations / communications

Include Relationship tussen twee Use Cases

Include-relatie laat zien dat het gedrag van de meegeleverde use case deel uitmaakt van de include (basis) use case. De belangrijkste reden hiervoor is om gemeenschappelijke acties te hergebruiken voor meerdere gebruiksscenario's. In sommige situaties wordt dit gedaan om complex gedrag te vereenvoudigen. Weinig dingen om rekening mee te houden bij het gebruik van de <<include>> relatie

De basis use case is niet compleet zonder de meegeleverde use case.

De meegeleverde use case is verplicht en niet optioneel.



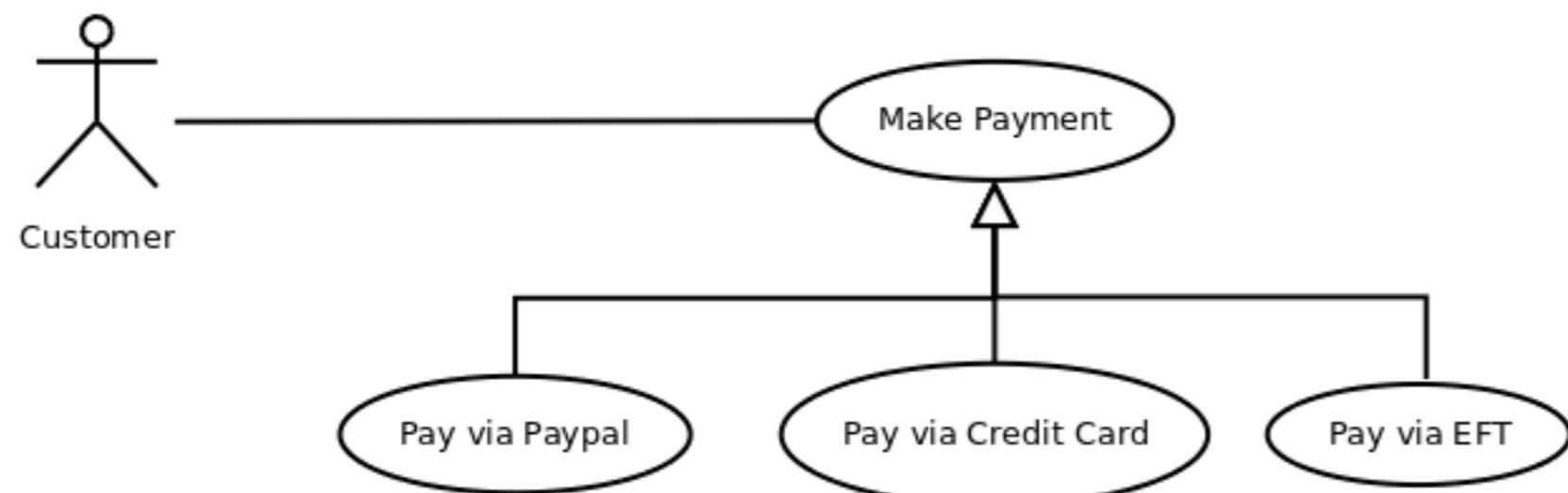
Use case diagram

Relations / communications

Generalization of a Use Case

Dit is vergelijkbaar met de generalisatie van een acteur. Het gedrag van de voorouder wordt geërfd door de afstammeling. Dit wordt gebruikt wanneer er gemeenschappelijk gedrag is tussen twee use-cases en ook gespecialiseerd gedrag dat specifiek is voor elke use-case.

In het vorige bankvoorbeeld kan er bijvoorbeeld een use-case zijn met de naam "Betaalrekeningen". Dit kan worden gegeneraliseerd naar "Betalen met creditcard", "Betalen met bankrekening" enz.



Use case diagram

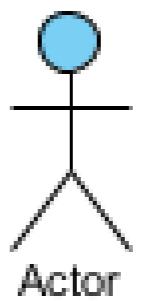
Tekenen van de components

Lucidchart

- 1 Open a new blank document
- 2 Open shape Library
- 3 Under UML check UML use cases
- 4 drag and drop the components
- 5 double click to type

Actor:

iemand die interactie heeft met het systeem (use cases). Je geeft het een zelfstandig naamwoord als naam. Primaire actoren (beïnvloeden) links, secundaire actoren (reageren) rechts.



Use case diagram

Tekenen van de components

Use case:

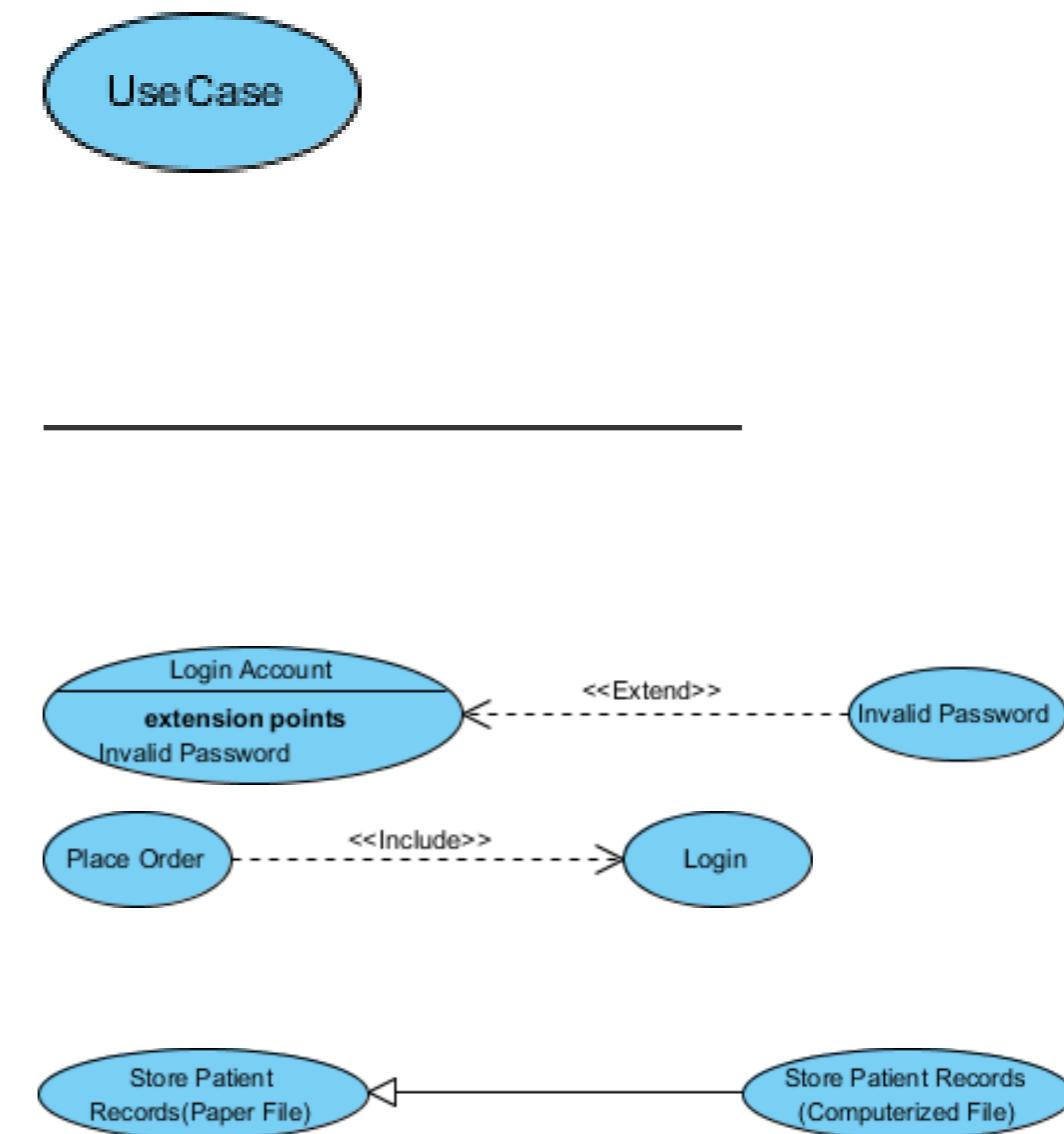
Systeemfunctie (proces - geautomatiseerd of handmatig). Wordt genoemd door werkwoord + zelfstandig naamwoord. Bijvoorbeeld een rekening openen of geld storten.

Relation / communication link:

De deelname van een actor aan een use case wordt weer-gegeven door een actor met een solide koppeling aan een use case te koppelen.

Relaties tussen use cases include en extend worden getekend met een stippe lijn en een pijl.

Generalisatie tussen actoren of use cases is een ononderbroken lijn met een open pijl.

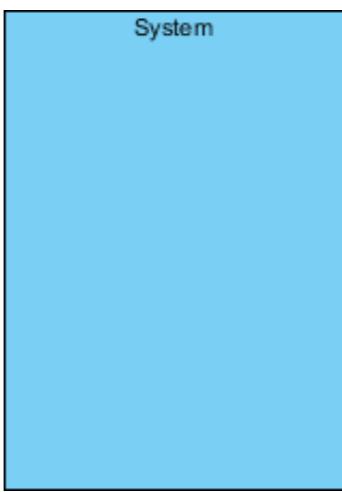


Use case diagram

Tekenen van de components

Boundary of system:

De systeemgrens is in potentie het gehele systeem zoals gedefinieerd in het eisendocument.



Alles in 1 video

| <https://www.youtube.com/watch?v=zid-MVo7M-E&t=294s>

Use case diagram

Extra bronnen

- | <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>
- | <https://www.lucidchart.com/pages/uml-use-case-diagram>
- | <https://creately.com/blog/diagrams/use-case-diagram-relationships/>
- | <https://www.youtube.com/watch?v=zid-MVo7M-E&t=294s>

Class diagram

Opdracht

In deze casus bouw je een **klassendiagram voor een coaching app.**

In deze app zijn er 4 type gebruikers:

- 1 stagiaire
- 2 coach
- 3 beheerder
- 4 hr medewerker

Alle gebruikers moeten kunnen inloggen (e-mail, wachtwoord). Daarnaast hebben we ook een naam en adres nodig. De gebruiker kan dit opgeven en later wijzigen. Ook moet de HR-medewerker de naam, het adres en het e-mailadres kunnen wijzigen.

Class diagram

Opdracht

Wat kan de app?

1:

De stagiair is de hoofdgebruiker van deze app. Het doel van deze stagiair is om zijn leerdoelen te formuleren en te rapporteren over zijn voortgang. Dit is voor harde en zachte vaardigheidsdoelen. Voor de harde vaardigheden zijn er cursussen die geen deel uitmaken van de app. Maar in de app kan een stagiair een lijst met cursussen vinden. De coach kan een lijst opstellen met programma's die de stagiair moet doen. De stagiair kan deze cursussen als voltooid markeren. Voor de soft skills krijgt de stagiair een uitnodiging van de coach voor een 1 op 1 of groepstraining. Van deze training moet de cursist een verslag schrijven en dit op zijn account zetten met de bijbehorende soft skill.

2:

De manager en de coach kunnen de voortgang van deze soft en hard skills volgen in hun dashboard. De leidinggevende en de coach kunnen stagiaires uitnodigen voor soft skill trainingen. De leidinggevende en de coach kunnen stagiairs uitnodigen voor evaluatiegesprekken.

Class diagram

Opdracht

Wat kan de app?

3:

De coach kan een lijst opstellen met must-do-cursussen voor de stagiairs.

4:

De HR-medewerker kan alleen de voortgang zien en naam, e-mail en adres van de medewerkers wijzigen.

De HR-medewerker kan elke gebruiker een reset-wachtwoord sturen.

5:

De beheerder kan alles wat de coach en hr medewerker kan. Daarnaast kan hij cursussen aanmaken / aanpassen / verwijderen. HR medewerkers en coaches aanmaken / aanpassen / verwijderen. De beheerder kan alles wat nodig is om de app functionerend te maken.