

31b) Notation von UML mit PlantUML

Prof. Dr. Uwe Aßmann
Institut für Software- und
Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 20-0.4, 05.06.20



DRESDEN
concept
Exzellenz aus
Wissenschaft
und Kultur

- ▶ PlantUML <https://plantuml.com/de/> ist ein tolles Werkzeug für das Zeichnen und automatisierte Layout von UML-Diagrammen
 - Klassendiagramme <https://plantuml.com/de/class-diagram>
 - Sequenzdiagramme <https://plantuml.com/de/sequence-diagram>
 - Komponentendiagramme (eingeschränkt)
<https://plantuml.com/de/component-diagram>
 - Nutzfalldiagramme <https://plantuml.com/de/use-case-diagram>
 - Zustandsdiagramme <https://plantuml.com/de/state-diagram>
 - Aktivitätsdiagramme <https://plantuml.com/de/activity-diagram-beta>
 - Objektdiagramme <https://plantuml.com/de/object-diagram>
- ▶ Man erstellt eine Datei `datei.plantuml` in textueller Syntax mit “Tripeln” (entspricht einfachen Sätzen wie “X ist-ein Y”)
- ▶ `plantuml datei.plantuml --> datei.png`

Nachinstallation mit einem Paketmanager

- ▶ Q: Wie bekommt man `plantuml` auf seinen Rechner?
- ▶ A: mit `brew` oder `apt-get`, den Paketmanagern von Linux:
- ▶ `brew install plantuml`
- ▶ `sudo apt-get install plantuml`

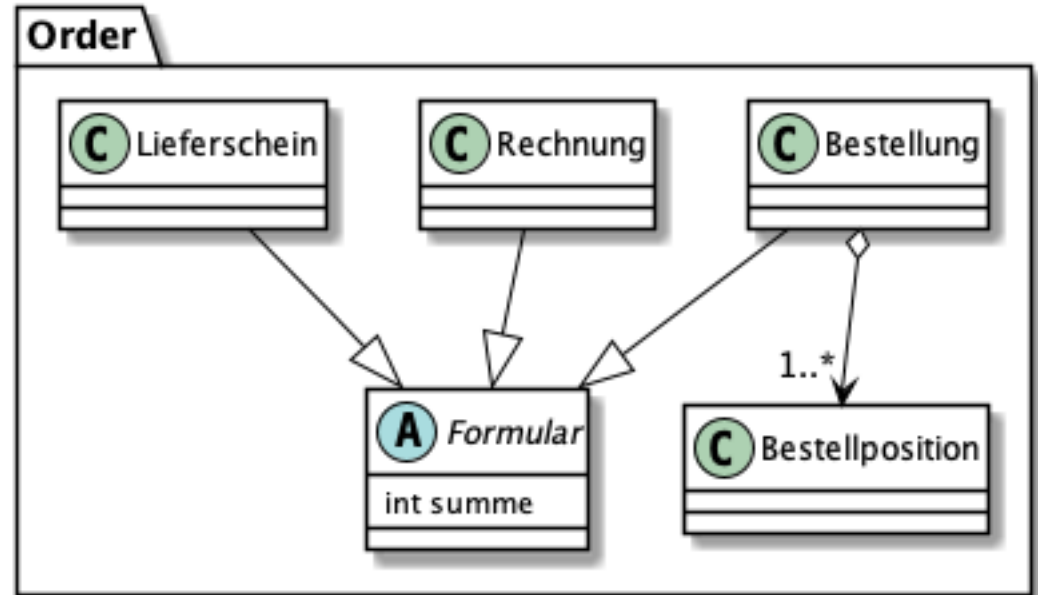
- ▶ Lustig:
- ▶ `plantuml -tutxt diagramm.plantuml`
- ▶ `--` erzeugt `utf8text`-Diagramm
- ▶ `plantuml -tpdf diagramm.plantuml`
- ▶ `--` erzeugt `pdf`-Diagramm

Unsere Formulare

plantuml/simple-form.plantuml

4 Softwaretechnologie (ST)

```
@startuml
' @author Uwe Assmann
' @version 0.1
' @date 2020-05-30
skinparam class {
  BackgroundColor White
  BorderColor Black
  ArrowColor Black
}
namespace Order {
  abstract class Formular
  Formular : int summe
  Bestellung o--> "1..*" Bestellposition
  Bestellung --|> Formular
  Rechnung --|> Formular
  Lieferschein --|> Formular
}
@enduml
```

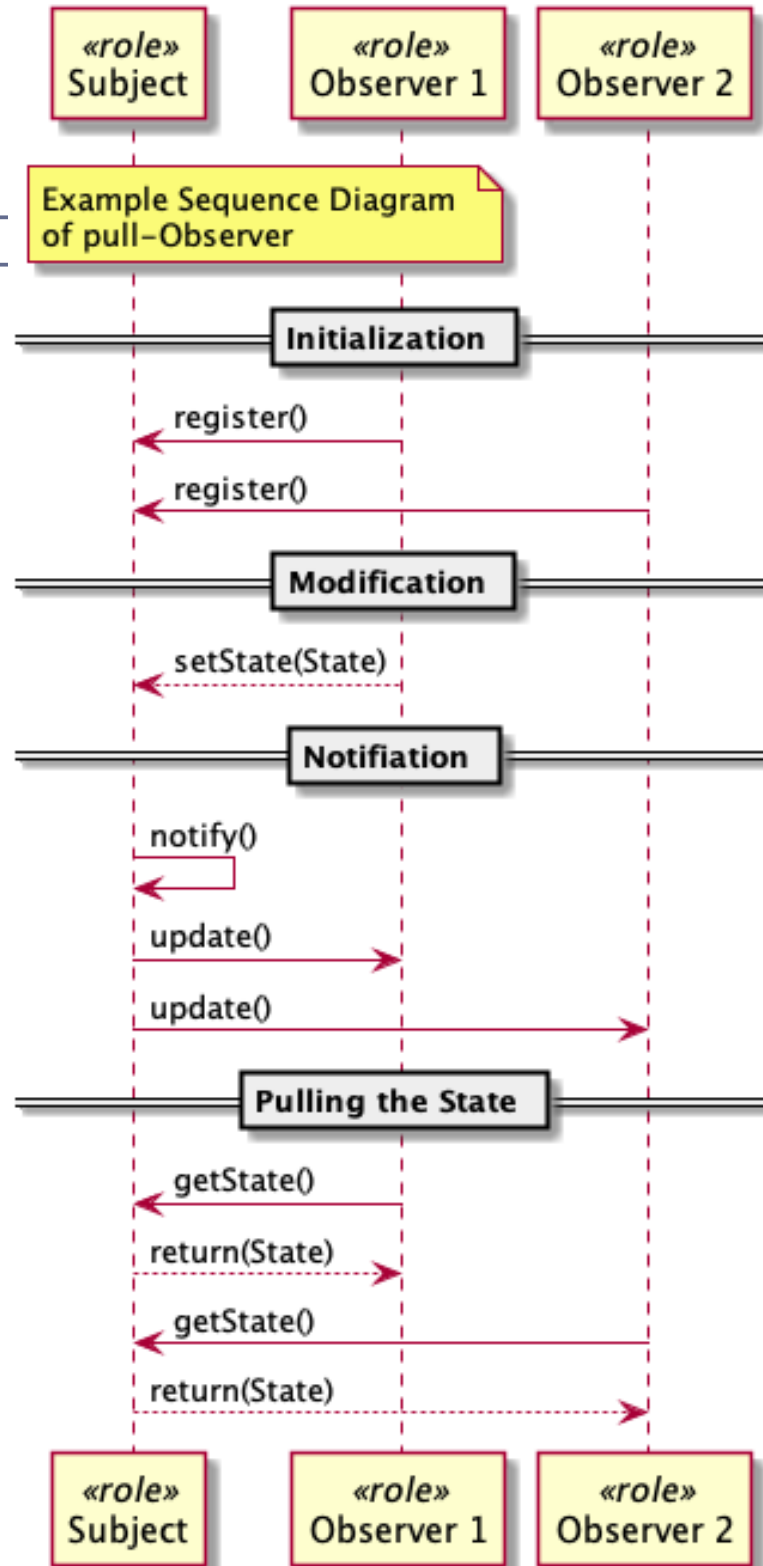


Sequenzdiagramm des pull-Observer (mit Phasen)

5 Softwaretechnol plantuml/sequence-diagram-observer.plantuml

```

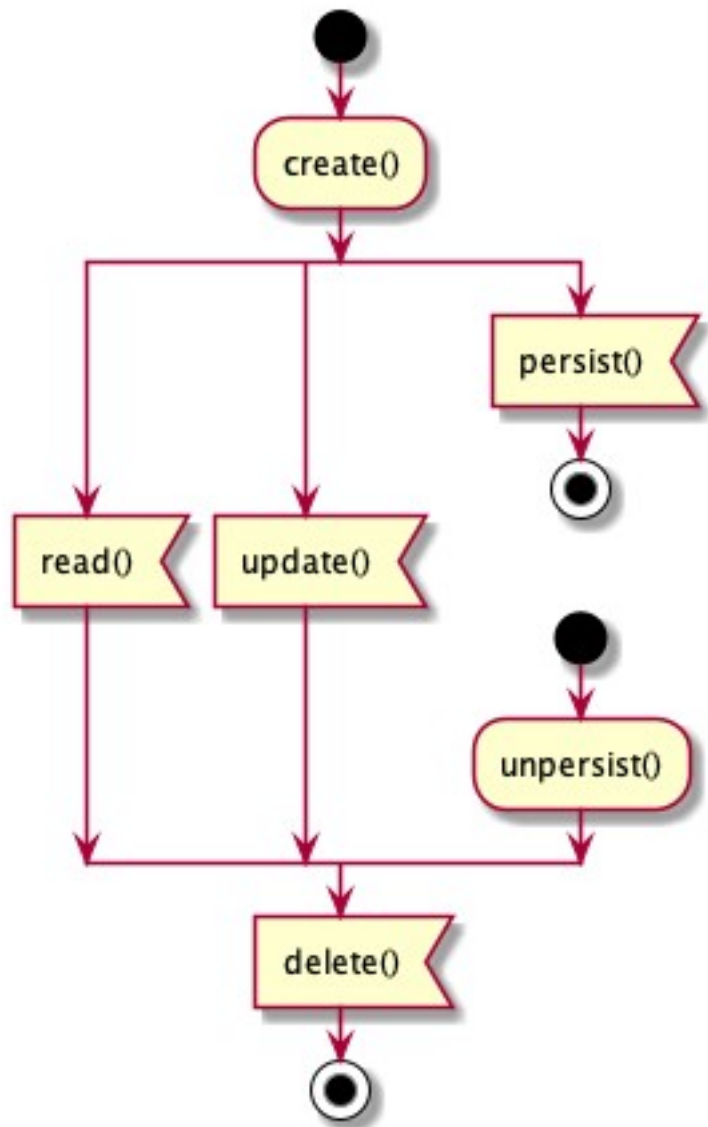
@startuml
' @author Uwe Assmann
' @version 0.1
' @date 2020-05-30
participant Subject <<role>>
participant "Observer 1" <<role>>
participant "Observer 2" <<role>>
note over Subject, "Observer 1": Example Sequence Diagram \nof pull-Observer
== Initialization ==
"Observer 1" -> Subject: register()
"Observer 2" -> Subject: register()
== Modification ==
"Observer 1" --> Subject: setState(State)
== Notifiatation ==
Subject -> Subject: notify()
Subject -> "Observer 1": update()
Subject -> "Observer 2": update()
== Pulling the State ==
"Observer 1" -> Subject: getState()
Subject--> "Observer 1" : return(State)
"Observer 2"-> Subject: getState()
Subject--> "Observer 2": return(State)
@enduml
    
```



Aktivitätendiagramm des CRUD Protokolls für Material-Objekte

```
@startuml
' @author Uwe Assmann
' @version 0.1
' @date 2020-05-30
start
:create();
split
:read()<
split again
:update()<
split again
:persist()<
stop
start
:unpersist();
end split

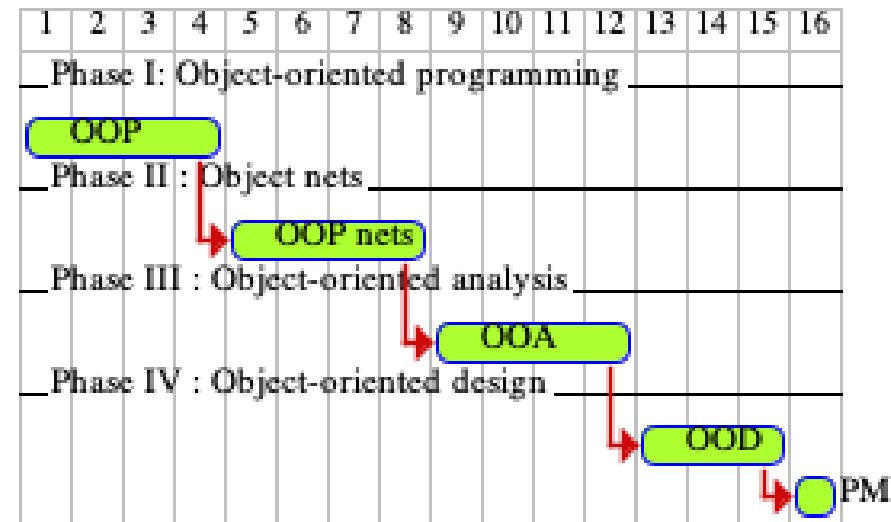
:delete()<
stop
@enduml
```



Projektpläne im GANTT Planungsformat (Balkendiagramm)

- ▶ Achtung: "days" sind hier einfach Zeiteinheiten
- ▶ (Probieren Sie das mal mit "weeks" aus)

```
@startgantt
-- Phase I: Object-oriented programming --
[OOP] lasts 4 days
-- Phase II : Object nets --
[OOP nets] lasts 4 days
[OOP nets] starts at [OOP]'s end
-- Phase III : Object-oriented analysis--
[OOA] lasts 4 days
[OOA] starts at [OOP nets]'s end
-- Phase IV : Object-oriented design--
[OOD] lasts 3 days
[OOD] starts at [OOA]'s end
[PM] lasts 1 day
[PM] starts at [OOD]'s end
@endgantt
```



Ende

- ▶ Erklären Sie den Begriff “Domänen-spezifische Sprache”. Denken Sie daran, was plantuml von Java unterscheidet.
- ▶ Erklären Sie, ob und warum Sie lieber mit textuellem plantuml oder mit diagrammatischem UML arbeiten.

31b) Notation von UML mit PlantUML

Prof. Dr. Uwe Aßmann
Institut für Software- und
Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 20-0.4, 05.06.20



DRESDEN
concept
Excellence aus
Wissenschaft
und Kultur

- ▶ PlantUML <https://plantuml.com/de/> ist ein tolles Werkzeug für das Zeichnen und automatisierte Layout von UML-Diagrammen
 - Klassendiagramme <https://plantuml.com/de/class-diagram>
 - Sequenzdiagramme <https://plantuml.com/de/sequence-diagram>
 - Komponentendiagramme (eingeschränkt) <https://plantuml.com/de/component-diagram>
 - Nutzfalldiagramme <https://plantuml.com/de/use-case-diagram>
 - Zustandsdiagramme <https://plantuml.com/de/state-diagram>
 - Aktivitätsdiagramme <https://plantuml.com/de/activity-diagram-beta>
 - Objektdiagramme <https://plantuml.com/de/object-diagram>
- ▶ Man erstellt eine Datei `datei.plantuml` in textueller Syntax mit "Tripeln" (entspricht einfachen Sätzen wie "X ist ein Y")
- ▶ `plantuml datei.plantuml --> datei.png`



Nachinstallation mit einem Paketmanager

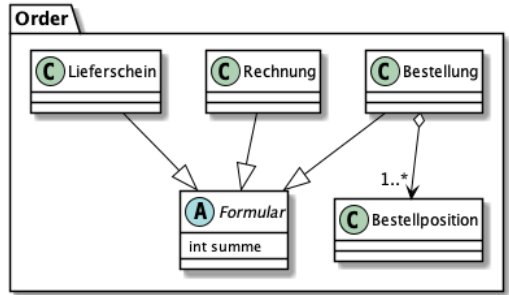
- ▶ Q: Wie bekommt man `plantuml` auf seinen Rechner?
- ▶ A: mit `brew` oder `apt-get`, den Paketmanagern von Linux:
- ▶ `brew install plantuml`
- ▶ `sudo apt-get install plantuml`

- ▶ Lustig:
- ▶ `plantuml -tutxt diagramm.plantuml`
- ▶ `--` erzeugt utf8text-Diagramm
- ▶ `plantuml -tpdf diagramm.plantuml`
- ▶ `--` erzeugt pdf-Diagramm

```

@startuml
' @author Uwe Assmann
' @version 0.1
' @date 2020-05-30
skinparam class {
  BackgroundColor White
  BorderColor Black
  ArrowColor Black
}
namespace Order {
  abstract class Formular
  Formular : int summe
  Bestellung o--> "1..*" Bestellposition
  Bestellung --|> Formular
  Rechnung --|> Formular
  Lieferschein --|> Formular
}
@enduml

```



Sequenzdiagramm des pull-Observer (mit Phasen)

5

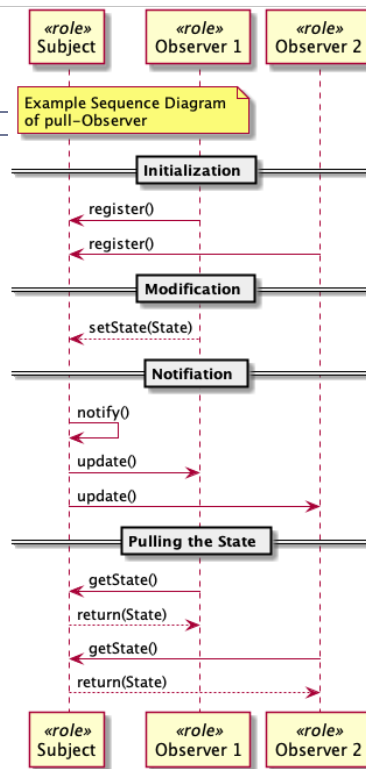
Softwaretechnol

plantuml/sequence-diagram-observer.plantuml

```

@startuml
' @author Uwe Assmann
' @version 0.1
' @date 2020-05-30
participant Subject <<role>>
participant "Observer 1" <<role>>
participant "Observer 2" <<role>>
note over Subject, "Observer 1": Example Sequence Diagram \nof pull-Observer
== Initialization ==
"Observer 1" -> Subject: register()
"Observer 2" -> Subject: register()
== Modification ==
"Observer 1" --> Subject: setState(State)
== Notifiatiion ==
Subject -> Subject: notify()
Subject -> "Observer 1": update()
Subject -> "Observer 2": update()
== Pulling the State ==
"Observer 1" -> Subject: getState()
Subject--> "Observer 1" : return(State)
"Observer 2"-> Subject: getState()
Subject--> "Observer 2" : return(State)
@enduml
    
```

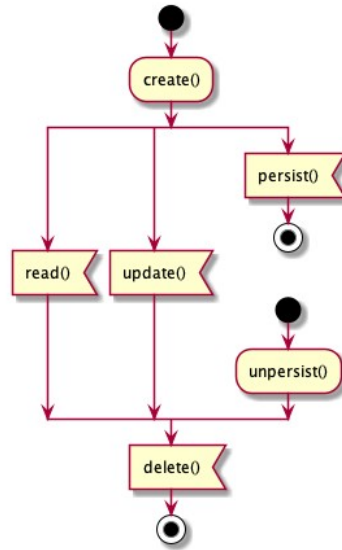
© Prof. U. Assmann



Aktivitätendiagramm des CRUD Protokolls für Material-Objekte

```
@startuml
' @author Uwe Assmann
' @version 0.1
' @date 2020-05-30
start
:create();
split
:read()<
split again
:update()<
split again
:persist()<
stop
start
:unpersist();
end split
:delete()<
stop
@enduml
```

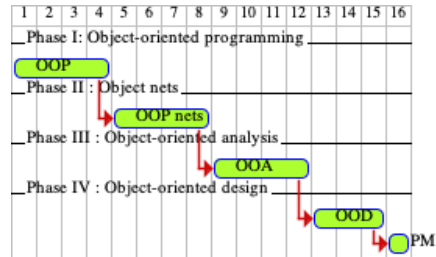
© Prof. U. Assmann



Projektpläne im GANTT Planungsformat (Balkendiagramm)

- ▶ Achtung: "days" sind hier einfach Zeiteinheiten
- ▶ (Probieren Sie das mal mit "weeks" aus)

```
@startgantt
-- Phase I: Object-oriented programming --
[OOP] lasts 4 days
-- Phase II : Object nets --
[OOP nets] lasts 4 days
[OOP nets] starts at [OOP]'s end
-- Phase III : Object-oriented analysis--
[OOA] lasts 4 days
[OOA] starts at [OOP nets]'s end
-- Phase IV : Object-oriented design--
[OOD] lasts 3 days
[OOD] starts at [OOA]'s end
[PM] lasts 1 day
[PM] starts at [OOD]'s end
@endgantt
```



Ende

- ▶ Erklären Sie den Begriff "Domänen-spezifische Sprache". Denken Sie daran, was plantuml von Java unterscheidet.
- ▶ Erklären Sie, ob und warum Sie lieber mit textuellem plantuml oder mit diagrammatischem UML arbeiten.