IHK Abschlussprüfung Sommer 2018

Entwicklung eines Softwaresystems

Thema: Netzplanerstellung

Martin Leonard Haufs

Fraunhofer IPT

WZL RWTH AACHEN

Inhalt

1. Vorstellung des Ausbildungsbetriebes
2. Eigene Projekte
3. Aufgabenstellung
4. Klassenstruktur
5. Algorithmen
6. Implementation

Fraunhofer IPT

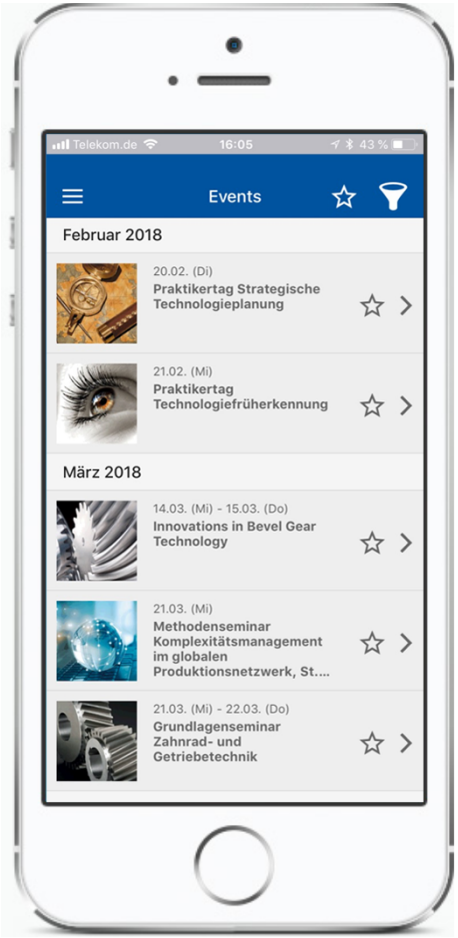
WZL RWTH AACHEN

1) Ausbildungsbetrieb

- Werkzeugmaschinenlabor (WZL)
  - Institut der RWTH Aachen
  - Abteilung für Werkzeugmaschinen
    - Prof. Dr. Brecher

    - Konstruktion und Berechnung von Produktionsanlagen
    - Maschinenelemente und –komponenten
    - Steuerungstechnik und Automatisierung
    - Informationstechnik und –management
    - Mensch-Maschine-Interaktion
    - Automatisierung

2) Meine Projekte

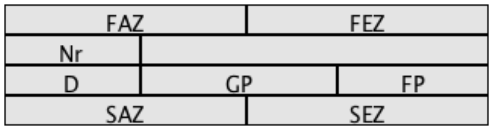


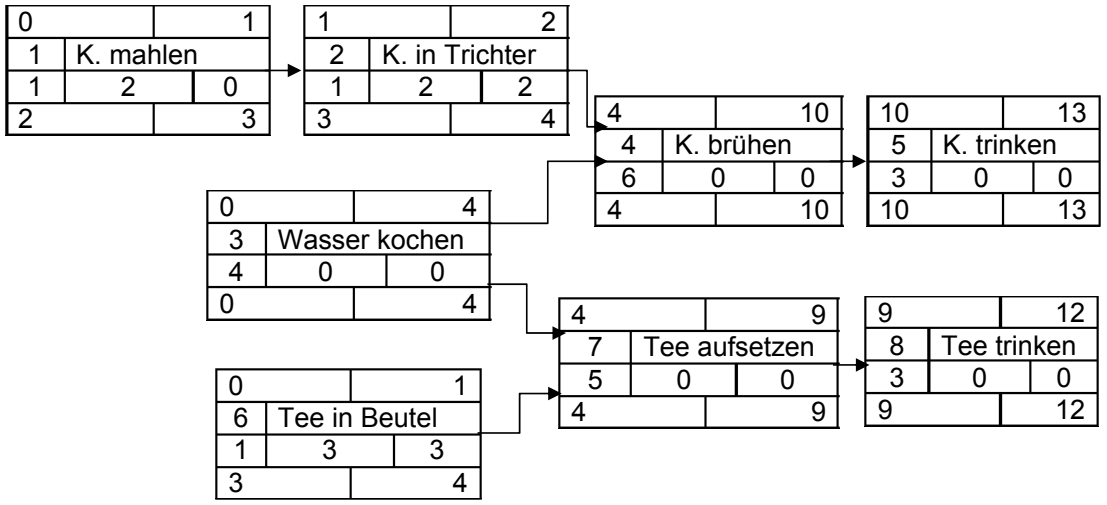
Hybride mobile Anwendungsentwicklung

- Tagungsapp für das Aachener Werkzeugmaschinen-Kolloquium (AWK)
  - Appcelerator Titanium SDK
  - Entwicklung nativer iOS-Module für Titanium in Objective C
- Messeapp für den DGSV Kongress
  - Weltkongress der Sterilgutversorgung
  - Technologie: Ionic 1 mit AngularJS
- Eventapp für das WZL-Forum
  - Ionic 1 mit Typescript
- Neukonzeption als Multieventapp
  - Umstellung von Ionic1 auf Ionic 2

titanium JS Objective-C ionic ANGULARJS TypeScript

3) Aufgabenstellung



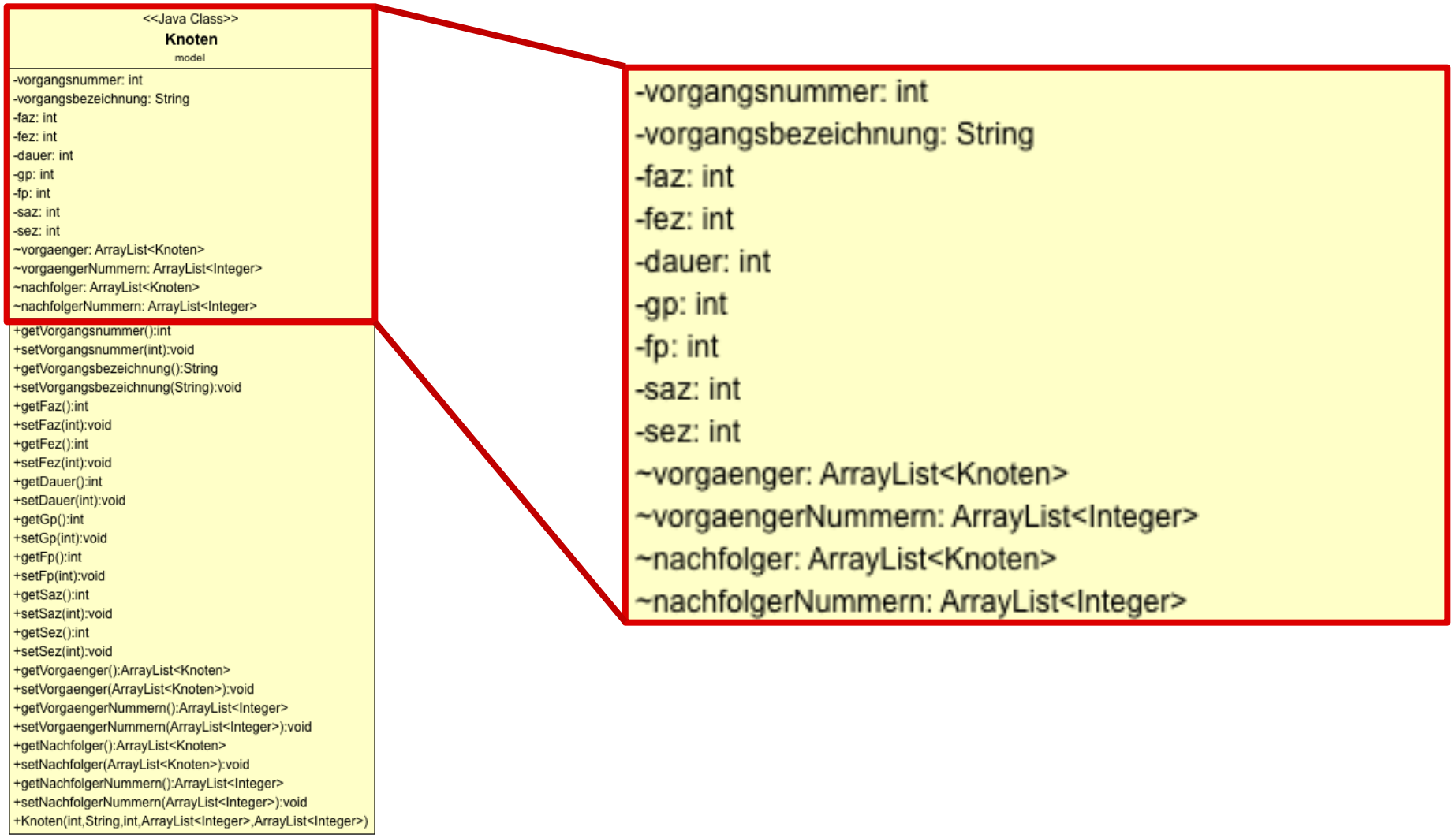


- Erstellung eines Netzplans
- Netzplan – Netz aus Knoten
  - Dauer (D)
  - Frühester Anfangszeitpunkt (FAZ)
  - Frühester Endzeitpunkt (FEZ)
  - Spätester Anfangszeitpunkt (SAZ)
  - Spätester Endzeitpunkt (SEZ)
  - Gesamtpuffer (GP)
  - Freier Puffer (FP)
- Ermittlung der Dauer des Gesamtprojektes
- Ermittlung der Kritischen Pfade

Fraunhofer IPT

WZL RWTH AACHEN

4) Klassenstruktur - Datenmodell



Fraunhofer IPT

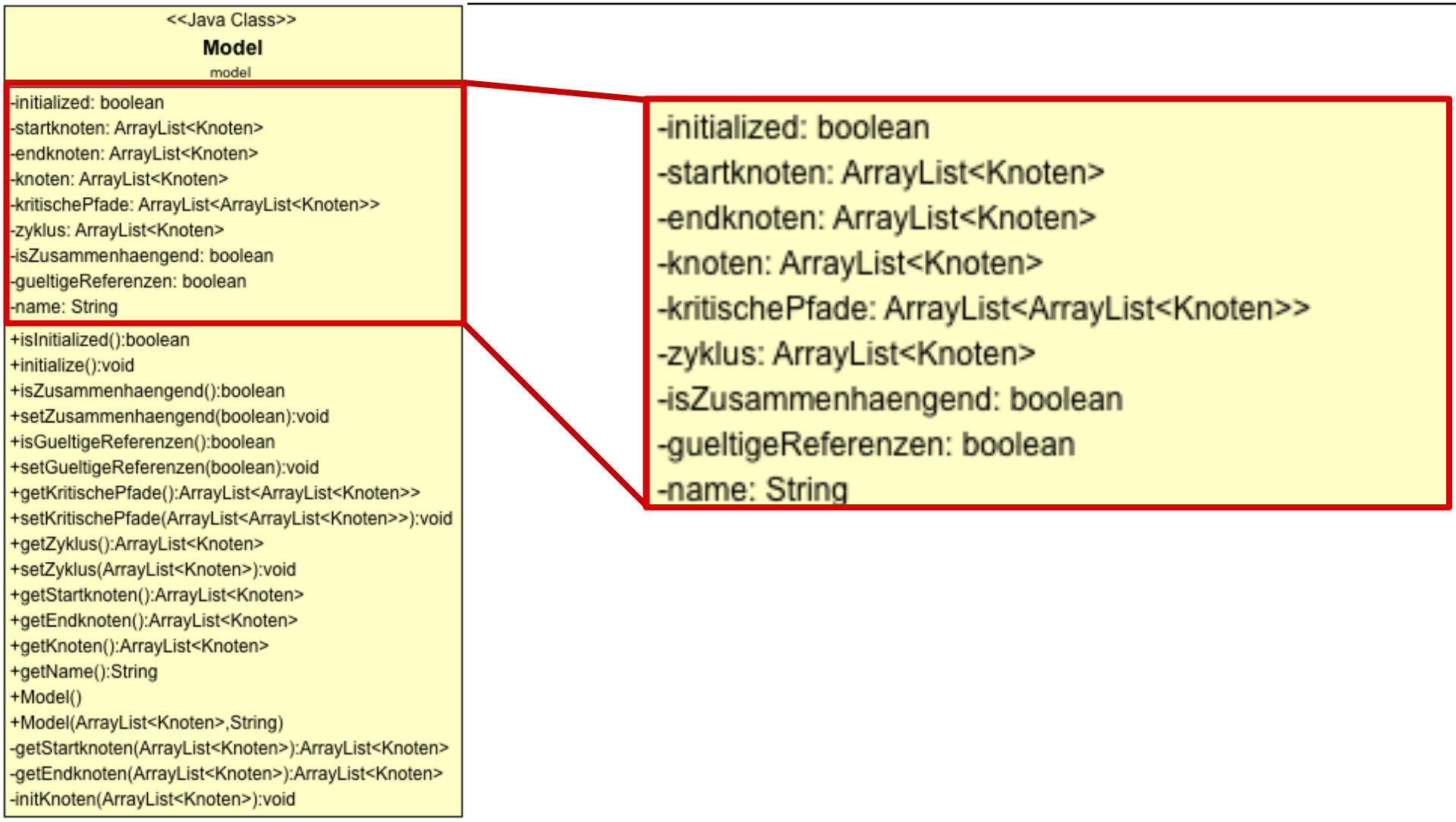
WZL RWTH AACHEN

Klassenstruktur - Datenmodell

Fraunhofer IPT

WZL RWTH AACHEN

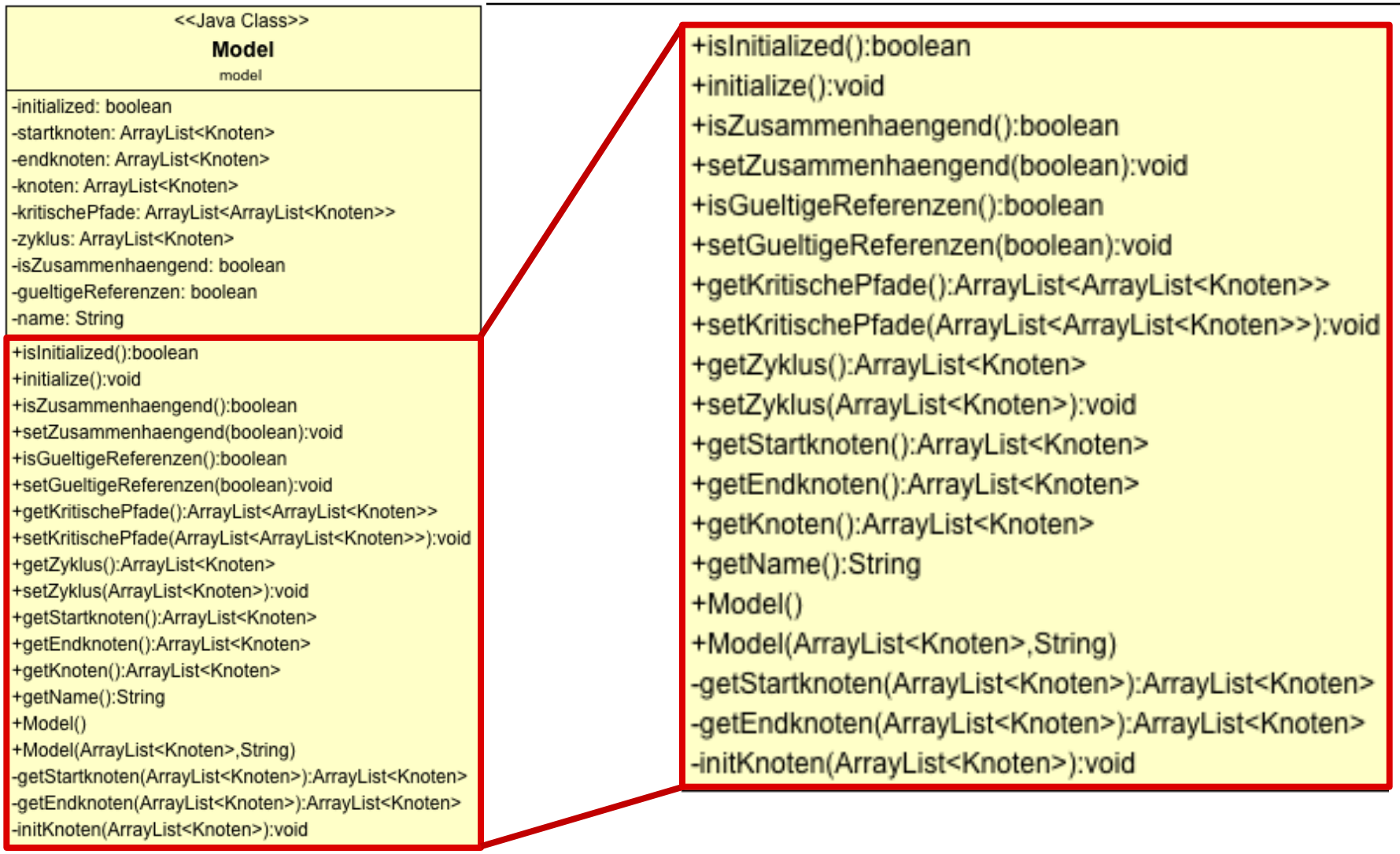
Klassenstruktur - Datenmodell



Fraunhofer IPT

WZL RWTH AACHEN

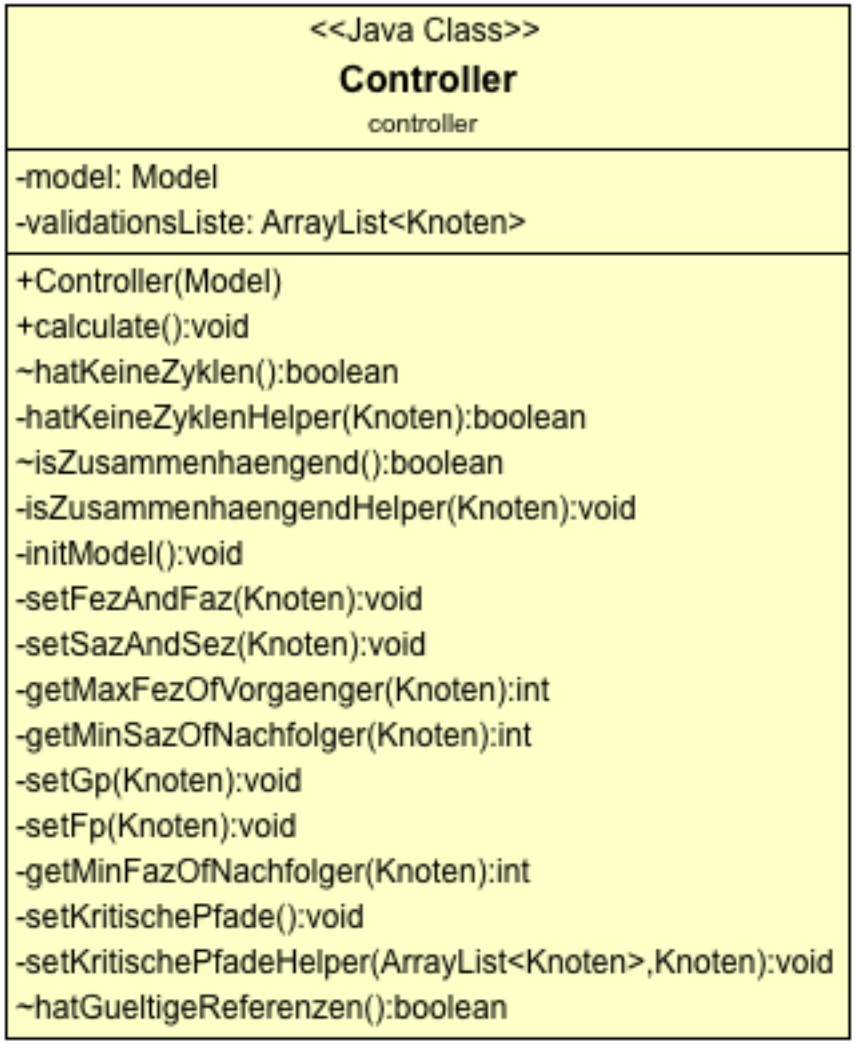
Klassenstruktur - Datenmodell



Fraunhofer IPT

WZL RWTH AACHEN

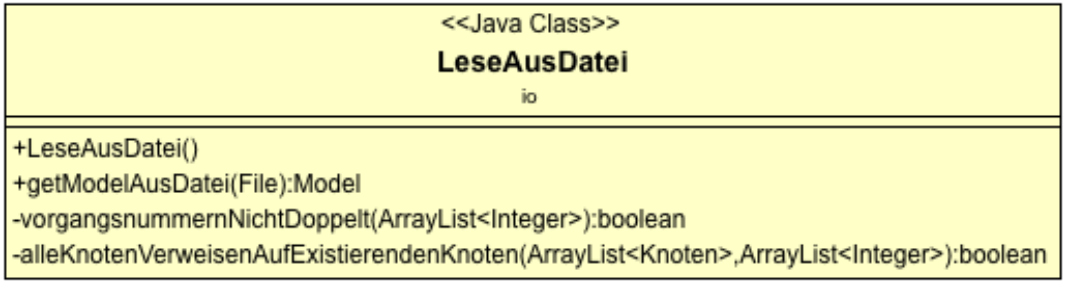
Klassenstruktur - Controller

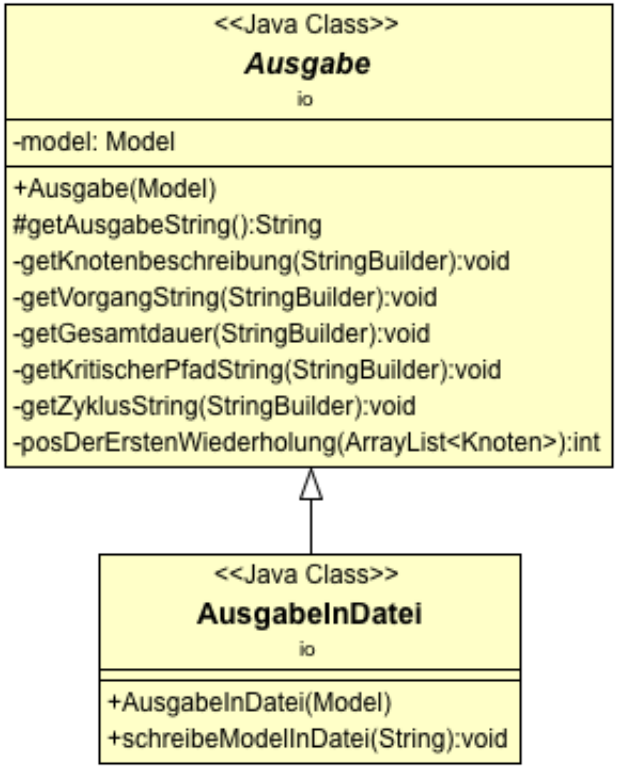


Fraunhofer IPT

WZL RWTH AACHEN

Klassenstruktur – Eingabe & Ausgabe





Fraunhofer IPT

WZL RWTH AACHEN

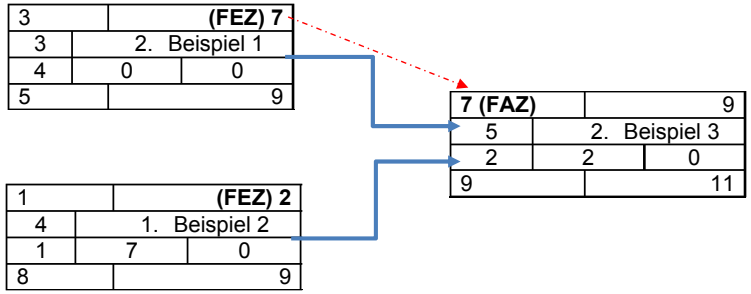
Algorithmus
a) Vorwärtsrechnung

- Startknoten hat FAZ = 0

Durchlaufen des Graphen von den Startpunkten → Endpunkten

- FEZ = FAZ + Dauer

- Größter FEZ eines Vorgängers ist FAZ des Nachfolgers



- Dauer (D)
- Frühester Anfangszeitpunkt (FAZ)
- Frühester Endzeitpunkt (FEZ)
- Spätester Anfangszeitpunkt (SAZ)
- Spätester Endzeitpunkt (SEZ)
- Gesamtpuffer (GP)
- Freier Puffer (FP)

Fraunhofer IPT

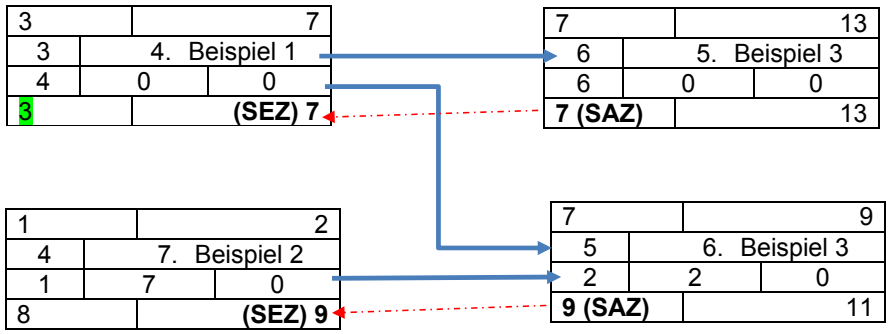
WZL RWTH AACHEN

Algorithmus
b) Rückwärtsrechnung

Durchlaufen des Graphen von den Endpunkten → Startpunkten

FAZ		FEZ
Nr	Name	
Dauer	GP	FP
SAZ		SEZ

- Endknoten haben SEZ = FEZ
- SAZ = SEZ – Dauer
- SEZ eines Knotens ist der minimale SAZ der Nachfolgeknoten

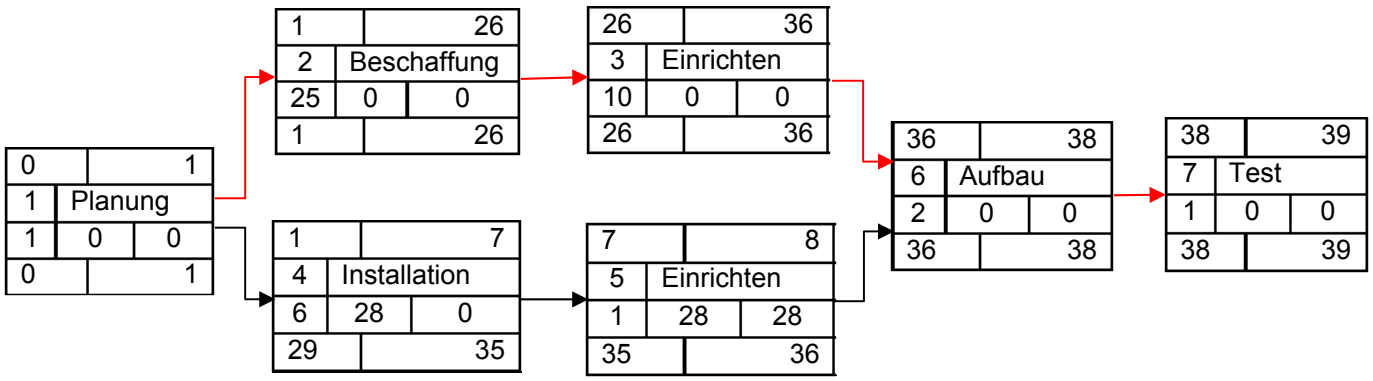


- Dauer (D)
- Frühester Anfangszeitpunkt (FAZ)
- Frühester Endzeitpunkt (FEZ)
- Spätester Anfangszeitpunkt (SAZ)
- Spätester Endzeitpunkt (SEZ)
- Gesamtpuffer (GP)
- Freier Puffer (FP)

Algorithmus
c) Ermittlung der Zeitreserven und des kritischen Pfades

Durchlaufen des Graphen von den Endpunkten → Startpunkten

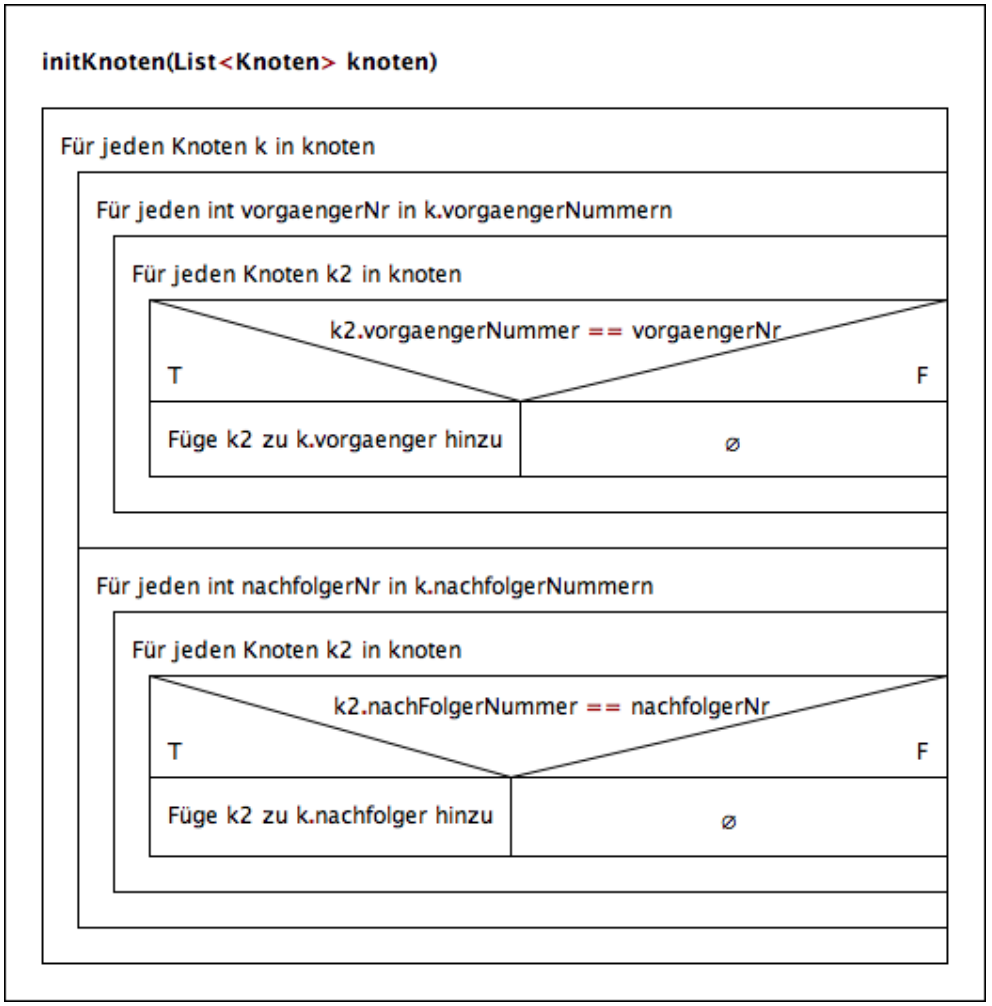
- GP = SAZ – FAZ <=> GP = SEZ – FEZ
- FP = (minimales FAZ der Nachfolger) – FEZ
- Kritischer Pfad:
  - Menge der Knoten mit GP = 0 und FP = 0



- Dauer (D)
- Frühester Anfangszeitpunkt (FAZ)
- Frühester Endzeitpunkt (FEZ)
- Spätester Anfangszeitpunkt (SAZ)
- Spätester Endzeitpunkt (SEZ)
- Gesamtpuffer (GP)
- Freier Puffer (FP)

Algorithmus

- Initialisieren des Graphen
  - In allen Knoten Vorgänger und Nachfolger referenzieren

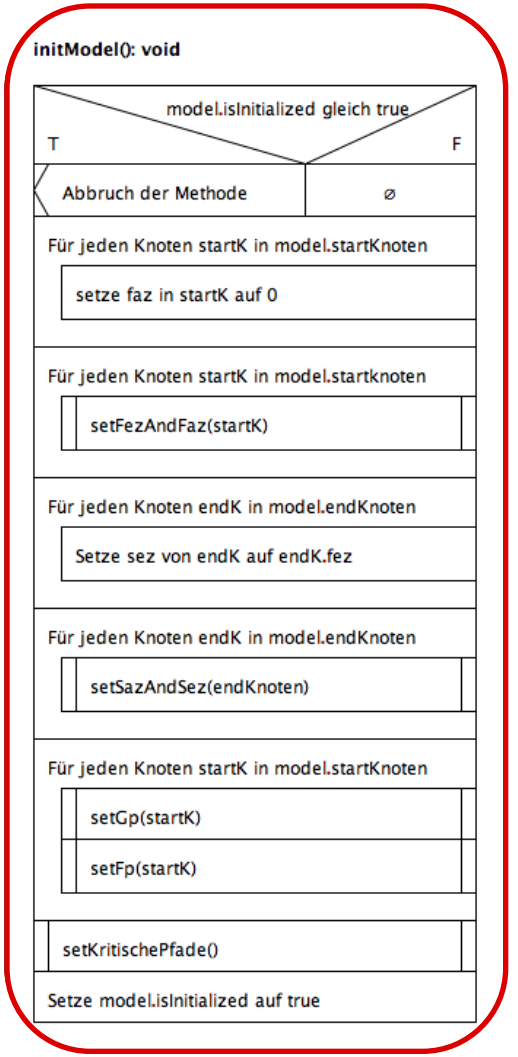


Fraunhofer IPT

WZL RWTH AACHEN

Algorithmus

- Vorwärtsrechnung
  - Frühester Endzeitpunkt (FEZ) und frühesten Anfangszeitpunkt (FAZ) berechnen
    - Rekursiver Baumdurchlauf ausgehend
    - von Startpunkten
- Rückwärtsrechnung
  - Späteste Endzeitpunkte (SEZ) und Späteste Anfangszeitpunkte (SAZ) berechnen
    - Rekursiver Baumdurchlauf ausgehend von Endpunkten
- Ermittlung der Zeitreserven
  - Rekursiver Baumdurchlauf
- Berechnung der Kritischen Pfade
  - Backtracking



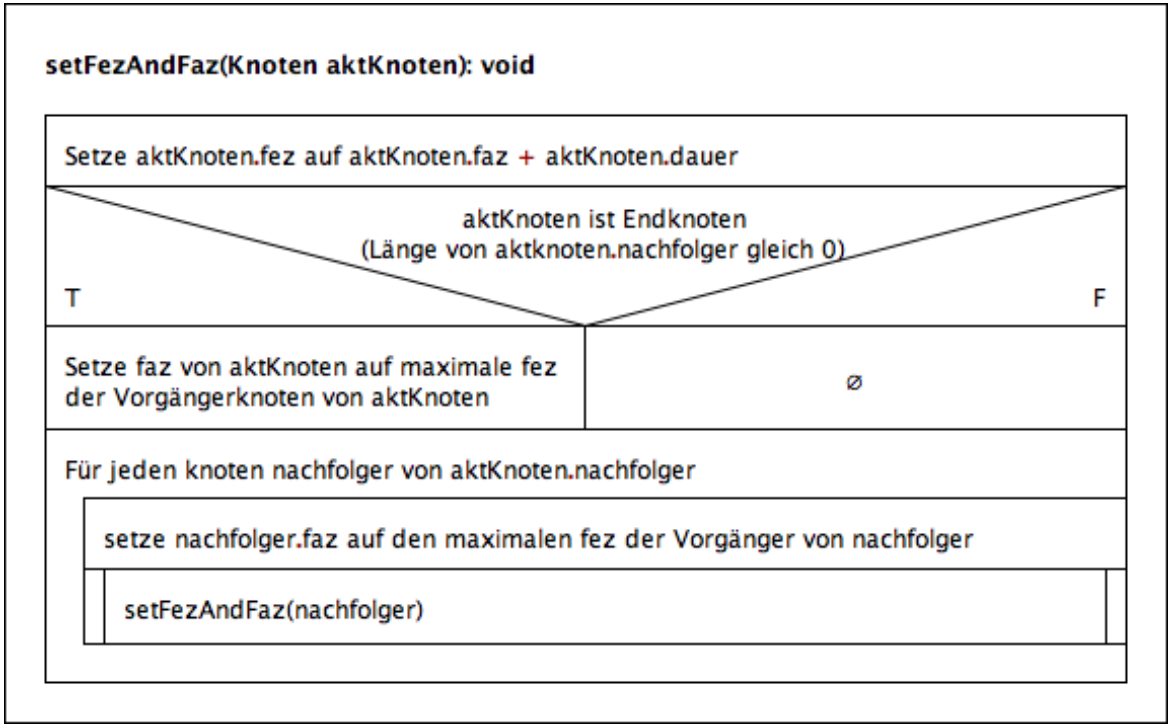
Fraunhofer IPT

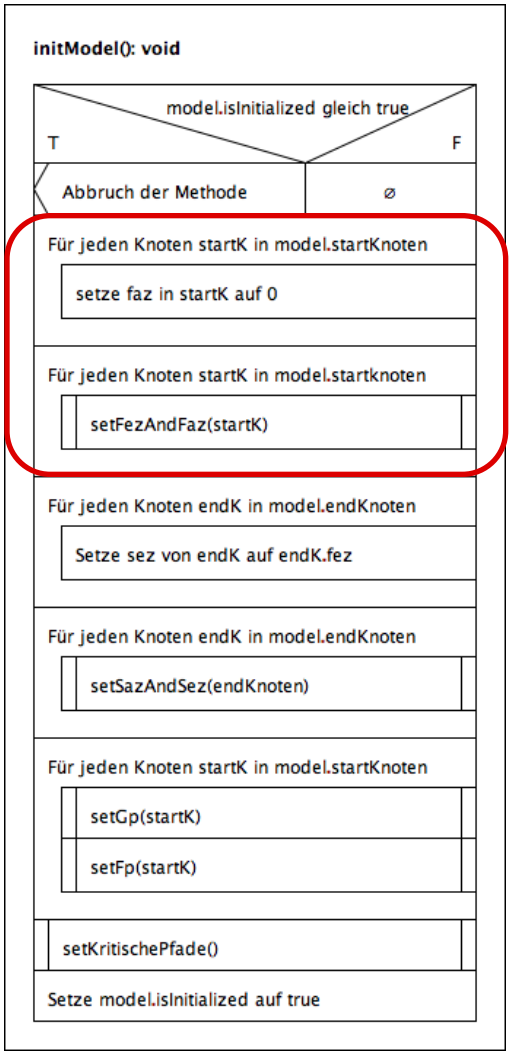
WZL RWTH AACHEN

Algorithmus

- Vorwärtsrechnung
  - Frühester Endzeitpunkt (FEZ) und frühesten Anfangszeitpunkt (FAZ) berechnen
    - Rekursiver Baumdurchlauf ausgehend
    - von Startpunkten

- Startknoten hat FAZ = 0
- FEZ = FAZ + Dauer
- Größter FEZ eines Vorgängers ist FAZ des Nachfolgers

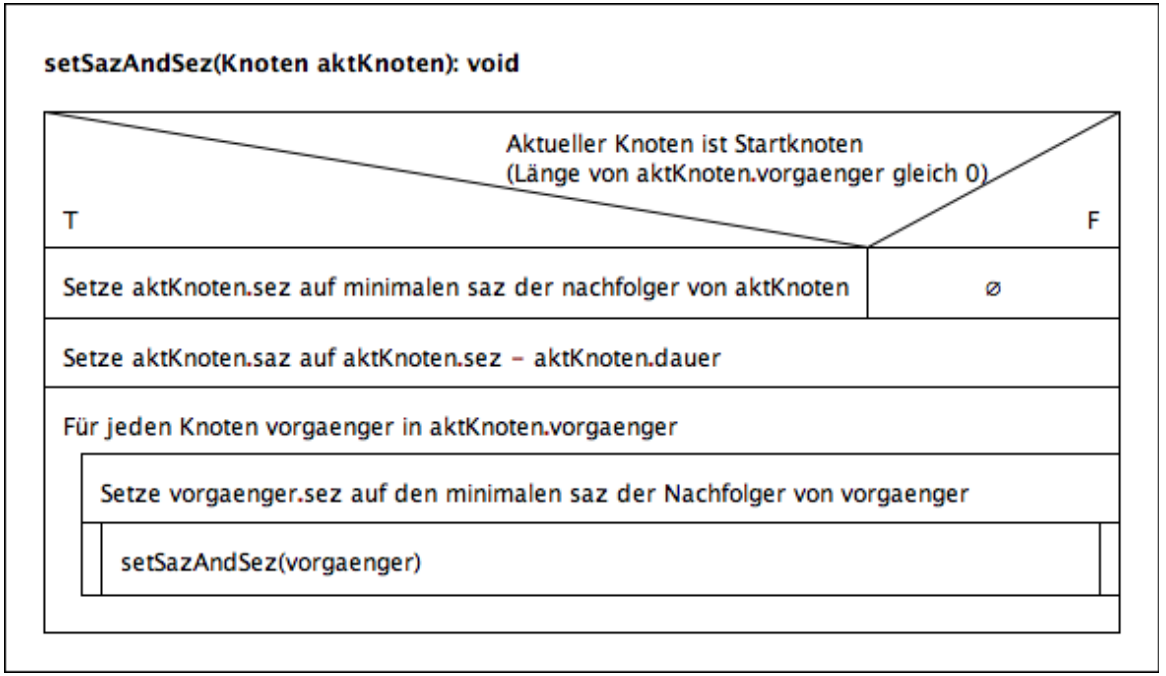


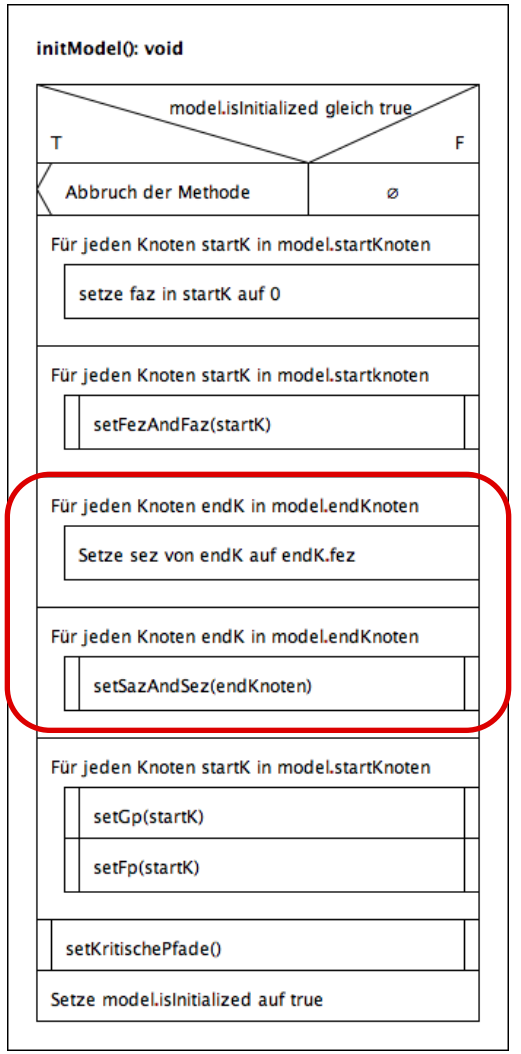


Algorithmus

- Rückwärtsrechnung
  - Späteste Endzeitpunkte (SEZ) und Späteste Anfangszeitpunkte (SAZ) berechnen
    - Rekursiver Baumdurchlauf ausgehend von Endpunkten

- Endknoten haben SEZ = FEZ
- SAZ = SEZ – Dauer
- SEZ eines Knotens ist das minimale SAZ der Nachfolgeknoten





Algorithmus

- Ermittlung der Zeitreserven
  - Rekursiver Baumdurchlauf

GP = SAZ – FAZ

FP = (minimales FAZ der Nachfolger) – FEZ

setGp(Knoten aktKnoten): void

Setze aktKnoten.gp auf aktKnote.saz – aktKnoten.faz

Für jeden Knoten nachfolger von aktKnoten.nachfolger

setGp(nachfolger)

setFp(Knoten aktKnoten): void

Setze aktKnoten.fp auf (minimales faz der Nachfolger von aktKnoten) – aktKnoten.fez

Für jeden Knoten nachfolger in aktKnoten.nachfolger

setFp(nachfolger)

initModel(): void

model.isInitialized gleich true

T | F

Abbruch der Methode | ø

Für jeden Knoten startK in model.startKnoten

setze faz in startK auf 0

Für jeden Knoten startK in model.startknoten

setFezAndFaz(startK)

Für jeden Knoten endK in model.endKnoten

Setze sez von endK auf endK.fez

Für jeden Knoten endK in model.endKnoten

setSazAndSez(endKnoten)

Für jeden Knoten startK in model.startKnoten

setGp(startK)

setFp(startK)

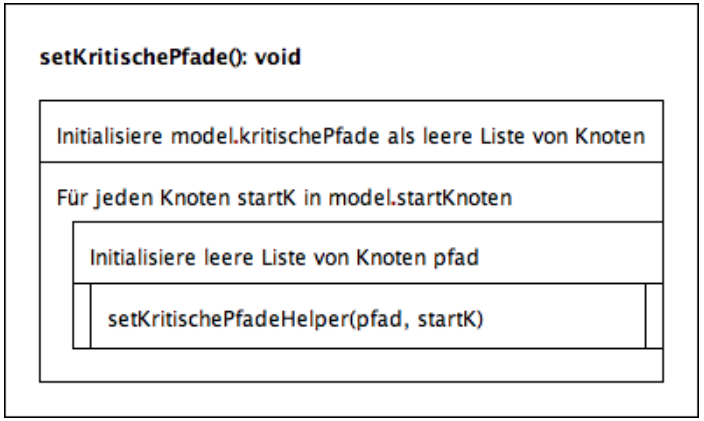
setKritischePfade()

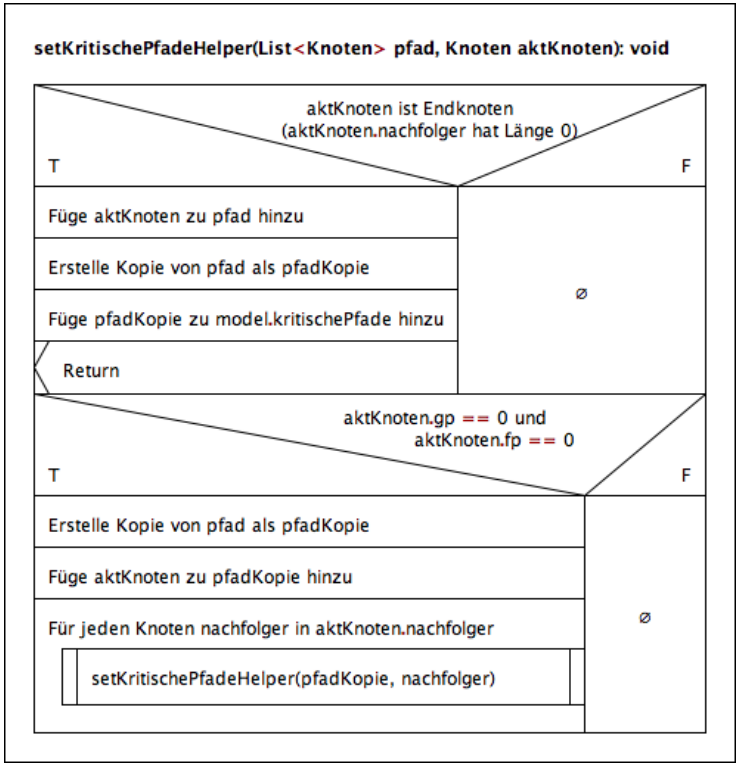
Setze model.isInitialized auf true

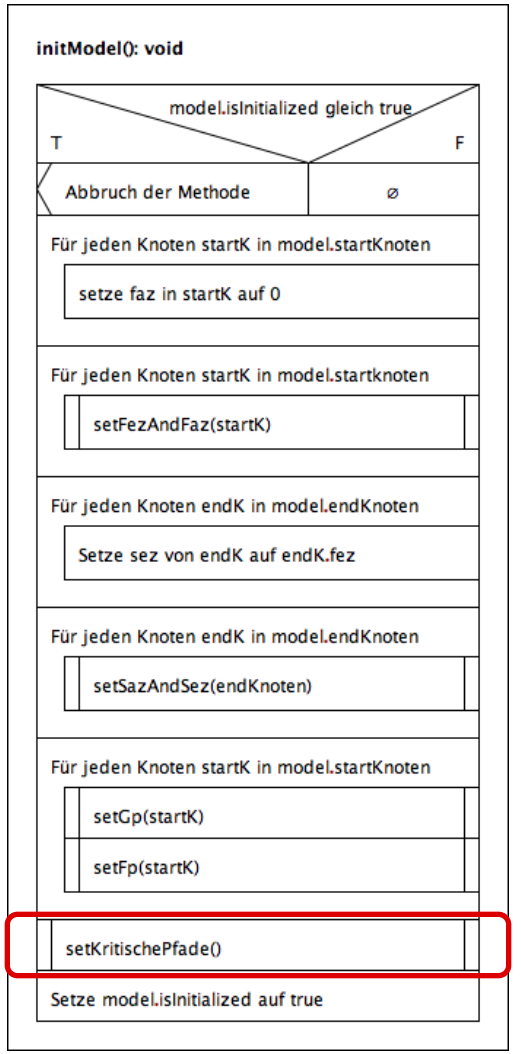
Algorithmus

- Berechnung der Kritischen Pfade
  - Backtracking
    - Abbruchkriterium: Endknoten erreicht

Kritischer Pfad:
Menge der Knoten mit
GP = 0 und FP = 0







Implementation

- Sprache Java
- Plattformübergreifend
- Shellscripte zum automatisierten öffnen der Tests

Fraunhofer IPT

WZL RWTH AACHEN