

# Softwaretechnik 1 - 1. Tutorium

Tutorium 17

Felix Bachmann | 14.05.2019

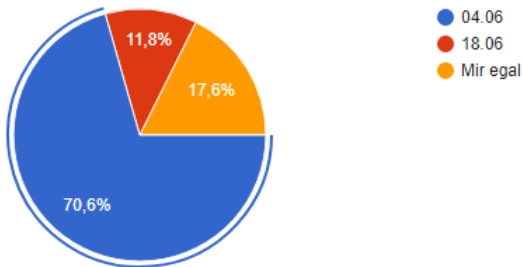
KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)



- 1 Orga
- 2 Wasserfallmodell
- 3 Durchführbarkeitsuntersuchung
- 4 Lastenheft
- 5 Pflichtenheft
- 6 UML-Klassendiagramm
- 7  $\text{\LaTeX}$
- 8 Tipps

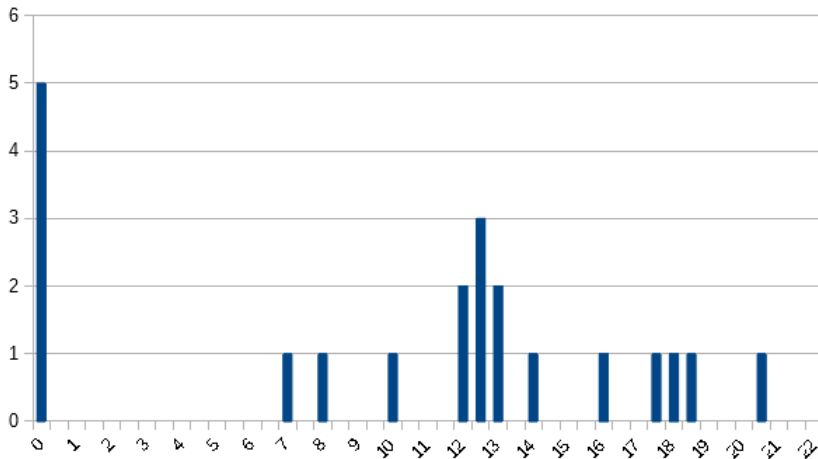
## Wann soll das Ersatztut für den 11.06. stattfinden?

17 Antworten



- Tut von 11.06 wird auf 04.06. verschoben
- ansonsten alles gleich (11:30-13:00, -119)

# 1. Übungsblatt Statistik



- für Feedback zu eurem Code Deckblatt einwerfen!

## Punktevergabe

generell ohne Abzug

- gleiche Abgabe bei allen Aufgaben

- für Feedback zu eurem Code Deckblatt einwerfen!

## Punktevergabe

generell ohne Abzug

- gleiche Abgabe bei allen Aufgaben

generell mit Abzug

- Code nicht CheckStyle-konform (auch Tests!)
  - Save Actions bei Eclipse! (IntelliJ ähnlich)
- JavaDoc nicht (vollständig && sinnvoll)
- Git-Commits nicht (regelmäßig && aussagekräftig)

## Abhängigkeiten deklarieren

- iMage dient dazu jmrst.main zu verpacken
- wenn ihr Abhängigkeiten in iMage-pom hinzufügt funktioniert es bei euch
  - aber es wird jmrst.main-pom in zip gepackt
  - bei mir explodiert es, da mein iMage die Abhängigkeit nicht hat

Lösung: Abhängigkeiten in dem Untermodul, das ihr gerade abgebt, deklarieren

## Abhängigkeiten deklarieren

- iMage dient dazu jmrst.main zu verpacken
- wenn ihr Abhängigkeiten in iMage-pom hinzufügt funktioniert es bei euch
  - aber es wird jmrst.main-pom in zip gepackt
  - bei mir explodiert es, da mein iMage die Abhängigkeit nicht hat

Lösung: Abhängigkeiten in dem Untermodul, das ihr gerade abgebt, deklarieren

## Parent

- als parent von Untermodulen iMage, nicht das remote uebungsparent benutzen!
- als Version des Parents 0.0.1-SNAPSHOT



## Abhängigkeiten deklarieren

- iMage dient dazu jmrst.main zu verpacken
- wenn ihr Abhängigkeiten in iMage-pom hinzufügt funktioniert es bei euch
  - aber es wird jmrst.main-pom in zip gepackt
  - bei mir explodiert es, da mein iMage die Abhängigkeit nicht hat

Lösung: Abhängigkeiten in dem Untermodul, das ihr gerade abgebt, deklarieren

## Parent

- als parent von Untermodulen iMage, nicht das remote uebungsparent benutzen!
- als Version des Parents 0.0.1-SNAPSHOT

ab nächstem ÜB Punktabzug, wenn ich noch irgendwas an der pom ändern muss, damit es bei mir läuft

## Aufgabe 1 (Altsoftware vorbereiten)

- falsche oder keine .gitignore in die zip verpackt
  - Konfiguration in src/assembly/src.xml
  - wenn ihr nur .gitignore hinschreibt, wird .gitignore aus jmjrst-Ordner genommen
  - die existiert bei euch vielleicht garnicht, oder ist default
    - in zip liegt keine oder falsche .gitignore
  - fix: ../../.gitignore statt .gitignore

## Aufgabe 1 (Altsoftware vorbereiten)

- falsche oder keine .gitignore in die zip verpackt
  - Konfiguration in src/assembly/src.xml
  - wenn ihr nur .gitignore hinschreibt, wird .gitignore aus jmjrst-Ordner genommen
  - die existiert bei euch vielleicht garnicht, oder ist default
    - in zip liegt keine oder falsche .gitignore
  - fix: ../../.gitignore statt .gitignore
- LICENSE sollte in das Wurzelverzeichnis
  - siehe <https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html>

## Aufgabe 2 + 3 (Modultests + Testüberdeckung)

- CheckStyle (und JavaDoc) auch in Tests benutzen

## Aufgabe 2 + 3 (Modultests + Testüberdeckung)

- CheckStyle (und JavaDoc) auch in Tests benutzen
- sinnvolle Definition der Gleichheit zweier Bilder
  - gleiche Dimensionen
  - UND gleiche Pixel-Werte

## Aufgabe 2 + 3 (Modultests + Testüberdeckung)

- CheckStyle (und JavaDoc) auch in Tests benutzen
- sinnvolle Definition der Gleichheit zweier Bilder
  - gleiche Dimensionen
  - UND gleiche Pixel-Werte
- auch bei Drehung um  $0^\circ$  ist Überprüfung von Gleichheit nötig

## Aufgabe 2 + 3 (Modultests + Testüberdeckung)

- CheckStyle (und JavaDoc) auch in Tests benutzen
- sinnvolle Definition der Gleichheit zweier Bilder
  - gleiche Dimensionen
  - UND gleiche Pixel-Werte
- auch bei Drehung um  $0^\circ$  ist Überprüfung von Gleichheit nötig
- `assertEquals()` reicht nicht aus, um Gleichheit der Bilder zu prüfen

## Aufgabe 2 + 3 (Modultests + Testüberdeckung)

- CheckStyle (und JavaDoc) auch in Tests benutzen
- sinnvolle Definition der Gleichheit zweier Bilder
  - gleiche Dimensionen
  - UND gleiche Pixel-Werte
- auch bei Drehung um  $0^\circ$  ist Überprüfung von Gleichheit nötig
- `assertEquals()` reicht nicht aus, um Gleichheit der Bilder zu prüfen
- Ordner `target/test` wurde nicht erstellt
  - `new File()` erstellt keine Datei, sondern nur einen "pointer" auf einen Pfad (siehe `File.createNewFile()` oder `File.mkdir()` oder `File.mkdirs()`)



## Aufgabe 2 + 3 (Modultests + Testüberdeckung)

- CheckStyle (und JavaDoc) auch in Tests benutzen
- sinnvolle Definition der Gleichheit zweier Bilder
  - gleiche Dimensionen
  - UND gleiche Pixel-Werte
- auch bei Drehung um  $0^\circ$  ist Überprüfung von Gleichheit nötig
- `assertEquals()` reicht nicht aus, um Gleichheit der Bilder zu prüfen
- Ordner `target/test` wurde nicht erstellt
  - `new File()` erstellt keine Datei, sondern nur einen "pointer" auf einen Pfad (siehe `File.createNewFile()` oder `File.mkdir()` oder `File.mkdirs()`)
- `@Test(expected=XYException.class)` nutzen, wenn Exception erwartet, sonst `@Ignore`

## Aufgabe 2 + 3 (Modultests + Testüberdeckung)

- Stil: Konstanten für Pfade benutzen
  - magic Strings sind ähnlich böse wie magic numbers

## Aufgabe 2 + 3 (Modultests + Testüberdeckung)

- Stil: Konstanten für Pfade benutzen
  - magic Strings sind ähnlich böse wie magic numbers
- Generator soll in @Before-Methode vor jedem Test neu erstellt werden
  - Objekt soll „frisch“ sein
  - Zustand des Objektes kann sich durch Methoden-Aufrufe verändern

## Aufgabe 2 + 3 (Modultests + Testüberdeckung)

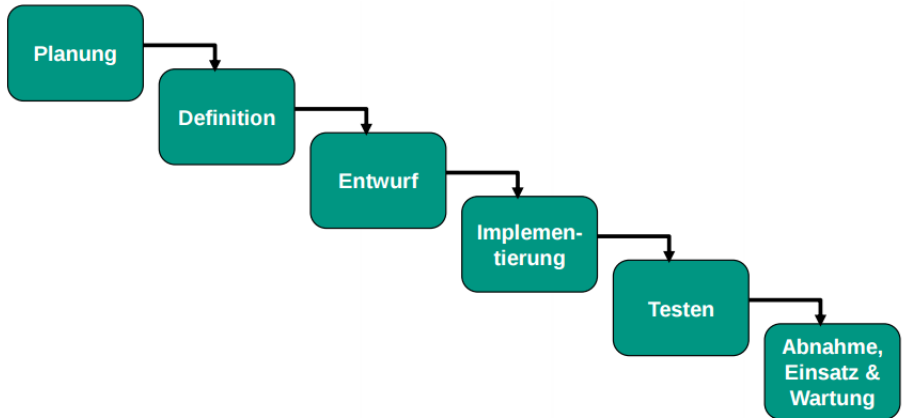
- Stil: Konstanten für Pfade benutzen
  - magic Strings sind ähnlich böse wie magic numbers
- Generator soll in @Before-Methode vor jedem Test neu erstellt werden
  - Objekt soll „frisch“ sein
  - Zustand des Objektes kann sich durch Methoden-Aufrufe verändern
- niemals absolute Pfade verwenden
  - ich habe bei mir keinen Ordner  
C:/Users/Manfred/Desktop/Uni/bloedes\_SWT/eclipseWorkspace/
  - besser: relative Pfade
  - am besten: Ressourcen über eingebaute Methoden suchen
    - `this.getClass().getResource(<Datei-Name>)`
    - ...
    - siehe <https://stackoverflow.com/questions/3861989/preferred-way-of-loading-resources-in-java>

## ■ Was ist das?

- dokumentengetriebenes Prozessmodell

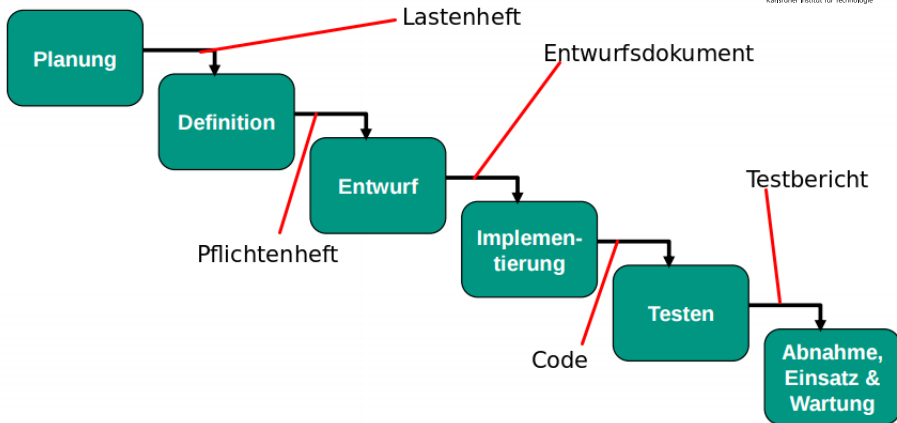
- dokumentengetriebenes Prozessmodell
- mögliche Phasen der Softwareentwicklung

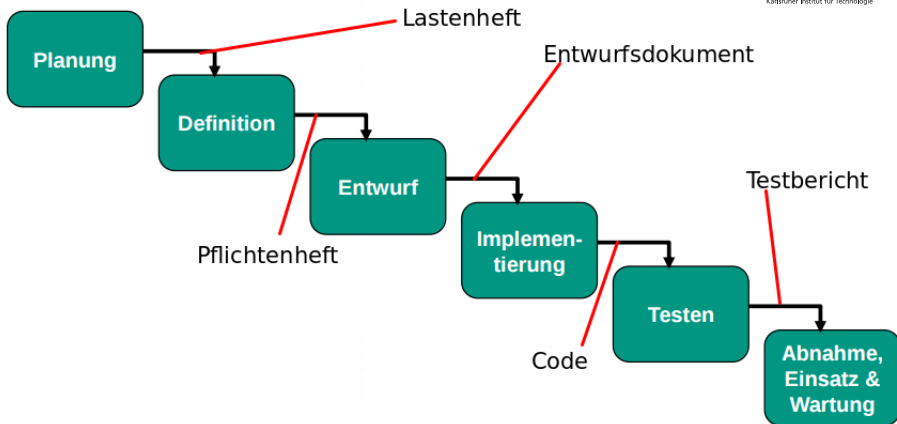
- dokumentengetriebenes Prozessmodell
- mögliche Phasen der Softwareentwicklung





# Wasserfallmodell





Dokumente für das 2. ÜB:

- Lastenheft
- Durchführbarkeitsuntersuchung (weiteres Artefakt der Planung)

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

1 Fachlich

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)
- 3 Personell



## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)
- 3 Personell (genug qualifiziertes Personal?)

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)
- 3 Personell (genug qualifiziertes Personal?)
- 4 Risiken

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)
- 3 Personell (genug qualifiziertes Personal?)
- 4 Risiken (Gibt es Risiken? :D)

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)
- 3 Personell (genug qualifiziertes Personal?)
- 4 Risiken (Gibt es Risiken? :D)
- 5 Ökonomisch

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)
- 3 Personell (genug qualifiziertes Personal?)
- 4 Risiken (Gibt es Risiken? :D)
- 5 Ökonomisch (wirtschaftlich? Termine?)

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)
- 3 Personell (genug qualifiziertes Personal?)
- 4 Risiken (Gibt es Risiken? :D)
- 5 Ökonomisch (wirtschaftlich? Termine?)
- 6 Rechtlich

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)
- 3 Personell (genug qualifiziertes Personal?)
- 4 Risiken (Gibt es Risiken? :D)
- 5 Ökonomisch (wirtschaftlich? Termine?)
- 6 Rechtlich (Datenschutz, Standards)

## Grundlegende Frage

Ist das Projekt in dem jeweiligen Szenario überhaupt durchführbar?

- 1 Fachlich (softwaretechnisch leicht realisierbar?)
- 2 Alternativen (lieber altes Projekt anpassen oder komplett neu entwickeln?)
- 3 Personell (genug qualifiziertes Personal?)
- 4 Risiken (Gibt es Risiken? :D)
- 5 Ökonomisch (wirtschaftlich? Termine?)
- 6 Rechtlich (Datenschutz, Standards)

## Fürs Übungsblatt

Denkt euch was (plausibles) aus!



## Grundlegende Aufgabe

Das Lastenheft sammelt die Anforderungen des Auftraggebers an den Auftragnehmer. Theoretisch vom Kunden geschrieben.

## Grundlegende Aufgabe

Das Lastenheft sammelt die Anforderungen des Auftraggebers an den Auftragnehmer. Theoretisch vom Kunden geschrieben.

- 1 Zielbestimmung (grobe Beschreibung)

## Grundlegende Aufgabe

Das Lastenheft sammelt die Anforderungen des Auftraggebers an den Auftragnehmer. Theoretisch vom Kunden geschrieben.

- 1 Zielbestimmung (grobe Beschreibung)
- 2 Produkteinsatz (Für wen? Zielgruppe, Anwendungsbereich)

## Grundlegende Aufgabe

Das Lastenheft sammelt die Anforderungen des Auftraggebers an den Auftragnehmer. Theoretisch vom Kunden geschrieben.

- 1 Zielbestimmung (grobe Beschreibung)
- 2 Produkteinsatz (Für wen? Zielgruppe, Anwendungsbereich)
- 3 Funktionale Anforderungen (feingranular: Funktionen des Produkts)

## Grundlegende Aufgabe

Das Lastenheft sammelt die Anforderungen des Auftraggebers an den Auftragnehmer. Theoretisch vom Kunden geschrieben.

- 1 Zielbestimmung (grobe Beschreibung)
- 2 Produkteinsatz (Für wen? Zielgruppe, Anwendungsbereich)
- 3 Funktionale Anforderungen (feingranular: Funktionen des Produkts)
- 4 Produktdaten (Welche Daten speichern?)

## Grundlegende Aufgabe

Das Lastenheft sammelt die Anforderungen des Auftraggebers an den Auftragnehmer. Theoretisch vom Kunden geschrieben.

- 1 Zielbestimmung (grobe Beschreibung)
- 2 Produkteinsatz (Für wen? Zielgruppe, Anwendungsbereich)
- 3 Funktionale Anforderungen (feingranular: Funktionen des Produkts)
- 4 Produktdaten (Welche Daten speichern?)
- 5 Nichtfunktionale Anforderungen (Meta-Anforderungen: Zeit, Zuverlässigkeit)

## Grundlegende Aufgabe

Das Lastenheft sammelt die Anforderungen des Auftraggebers an den Auftragnehmer. Theoretisch vom Kunden geschrieben.

- ① Zielbestimmung (grobe Beschreibung)
- ② Produkteinsatz (Für wen? Zielgruppe, Anwendungsbereich)
- ③ Funktionale Anforderungen (feingranular: Funktionen des Produkts)
- ④ Produktdaten (Welche Daten speichern?)
- ⑤ Nichtfunktionale Anforderungen (Meta-Anforderungen: Zeit, Zuverlässigkeit)
- ⑥ Systemmodelle
  - Szenarien (spezielles Beispiel)
  - Anwendungsfälle (allgemeiner Verwendungszweck)

## Grundlegende Aufgabe

Das Lastenheft sammelt die Anforderungen des Auftraggebers an den Auftragnehmer. Theoretisch vom Kunden geschrieben.

- ➊ Zielbestimmung (grobe Beschreibung)
- ➋ Produkteinsatz (Für wen? Zielgruppe, Anwendungsbereich)
- ➌ Funktionale Anforderungen (feingranular: Funktionen des Produkts)
- ➍ Produktdaten (Welche Daten speichern?)
- ➎ Nichtfunktionale Anforderungen (Meta-Anforderungen: Zeit, Zuverlässigkeit)
- ➏ Systemmodelle
  - Szenarien (spezielles Beispiel)
  - Anwendungsfälle (allgemeiner Verwendungszweck)
- ➐ Glossar (technische Begriffe erklären)



## Zielbestimmung vs. Funktionale Anforderungen

## Zielbestimmung vs. Funktionale Anforderungen

- Zielbestimmung: allgemeine Beschreibung, was das Produkt können soll
- Funktionale Anforderungen: konkrete Auflistung von Funktionen

## Zielbestimmung vs. Funktionale Anforderungen

- Zielbestimmung: allgemeine Beschreibung, was das Produkt können soll
- Funktionale Anforderungen: konkrete Auflistung von Funktionen

## Funktionale Anforderungen vs. Nichtfunktionale Anforderungen

## Zielbestimmung vs. Funktionale Anforderungen

- Zielbestimmung: allgemeine Beschreibung, was das Produkt können soll
- Funktionale Anforderungen: konkrete Auflistung von Funktionen

## Funktionale Anforderungen vs. Nichtfunktionale Anforderungen

- Funktionale Anforderungen: Funktionen des Produkts
- Nichtfunktionale Anforderungen: "Meta"-Eigenschaften des Produkts

## Zielbestimmung vs. Funktionale Anforderungen

- Zielbestimmung: allgemeine Beschreibung, was das Produkt können soll
- Funktionale Anforderungen: konkrete Auflistung von Funktionen

## Funktionale Anforderungen vs. Nichtfunktionale Anforderungen

- Funktionale Anforderungen: Funktionen des Produkts
- Nichtfunktionale Anforderungen: "Meta"-Eigenschaften des Produkts

## Zielbestimmung vs. Produkteinsatz

## Zielbestimmung vs. Funktionale Anforderungen

- Zielbestimmung: allgemeine Beschreibung, was das Produkt können soll
- Funktionale Anforderungen: konkrete Auflistung von Funktionen

## Funktionale Anforderungen vs. Nichtfunktionale Anforderungen

- Funktionale Anforderungen: Funktionen des Produkts
- Nichtfunktionale Anforderungen: "Meta"-Eigenschaften des Produkts

## Zielbestimmung vs. Produkteinsatz

- Zielbestimmung: allgemeine Beschreibung, was das Produkt können soll
- Produkteinsatz: Rahmenbedingungen (Zielgruppe, Anwendungsbereiche)

## Grundlegende Aufgabe

Erweiterung des Lastenheftes, sodass exakt abgebildet ist **was** (noch nicht **wie**) zu implementieren ist. Vom Entwickler geschrieben.

## Grundlegende Aufgabe

Erweiterung des Lastenheftes, sodass exakt abgebildet ist **was** (noch nicht **wie**) zu implementieren ist. Vom Entwickler geschrieben.

- man kann es sich so merken
  - erst werden uns „Lasten“ vom Kunden auferlegt
  - daraus generieren wir dann „Pflichten“
  - „Pflichten“ Grundlage für Entwurf



- ① Zielbestimmung
- ② Produkteinsatz
- ③ **Produktumgebung** (Hard-/Software in Einsatzumgebung)
- ④ Funktionale Anforderungen
- ⑤ Produktdaten
- ⑥ Nichtfunktionale Anforderungen
- ⑦ **Globale Testfälle** („zu testende Abläufe“)
- ⑧ Systemmodelle
  - Szenarien
  - Anwendungsfälle
  - **Objektmodelle**  $\implies$  UML-Klassendiagramme (heute)
  - **Dynamische Modelle**  $\implies$  nächstes Mal
  - **Benutzerschnittstelle**  $\implies$  Zeichnungen/Screenshots
- ⑨ Glossar

## Produkteinsatz vs. Produktumgebung

## Produkteinsatz vs. Produktumgebung

- Produkteinsatz: Rahmenbedingungen (Zielgruppe, Anwendungsbereiche)
- Produktumgebung: Rahmenbedingungen bzgl. Software/Hardware

Wahr oder falsch?

- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes.

Wahr oder falsch?

- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes. falsch

Wahr oder falsch?

- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes. falsch
- Das Lastenheft ist das Ergebnis der Planungsphase.

Wahr oder falsch?

- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes. falsch
- Das Lastenheft ist das Ergebnis der Planungsphase. wahr

Wahr oder falsch?

- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes. falsch
- Das Lastenheft ist das Ergebnis der Planungsphase. wahr
- Nicht-funktionale Eigenschaften beschreiben, was das Produkt nicht tun sollte.



Wahr oder falsch?

- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes. falsch
- Das Lastenheft ist das Ergebnis der Planungsphase. wahr
- Nicht-funktionale Eigenschaften beschreiben, was das Produkt nicht tun sollte. falsch

Wahr oder falsch?

- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes. falsch
- Das Lastenheft ist das Ergebnis der Planungsphase. wahr
- Nicht-funktionale Eigenschaften beschreiben, was das Produkt nicht tun sollte. falsch
- Das Pflichtenheft beschreibt nur, was zu implementieren ist und nicht wie.

Wahr oder falsch?

- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes. falsch
- Das Lastenheft ist das Ergebnis der Planungsphase. wahr
- Nicht-funktionale Eigenschaften beschreiben, was das Produkt nicht tun sollte. falsch
- Das Pflichtenheft beschreibt nur, was zu implementieren ist und nicht wie. wahr

Wahr oder falsch?

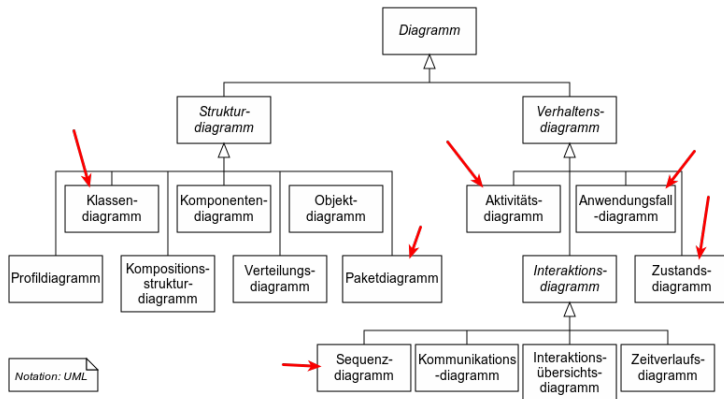
- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes. falsch
- Das Lastenheft ist das Ergebnis der Planungsphase. wahr
- Nicht-funktionale Eigenschaften beschreiben, was das Produkt nicht tun sollte. falsch
- Das Pflichtenheft beschreibt nur, was zu implementieren ist und nicht wie. wahr
- Nicht-funktionale Anforderungen sind sowohl Teil des Pflichtenhefts als auch des Lastenhefts.

Wahr oder falsch?

- Das Lastenheft ist eine Verfeinerung des Pflichtenheftes. falsch
- Das Lastenheft ist das Ergebnis der Planungsphase. wahr
- Nicht-funktionale Eigenschaften beschreiben, was das Produkt nicht tun sollte. falsch
- Das Pflichtenheft beschreibt nur, was zu implementieren ist und nicht wie. wahr
- Nicht-funktionale Anforderungen sind sowohl Teil des Pflichtenhefts als auch des Lastenhefts. wahr

# UML? Kann man das essen?

- UML = **U**nified **M**odeling **L**anguage
- grafische Modellierungssprache, strenge Syntax



- **Klassenname**
  - keine spezielle Syntax
  - einfach Namen hinschreiben
- **Attribute**
  - `<modifier><name>:<type>`
- **Methoden**
  - `<modifier><name>(<parameters>):<type>`
  - `<parameters>`
    - kann leer sein
    - oder komma-getrennte Liste von `<name>:<type>`
  - falls Rückgabe void, `:<type>` weglassen
- **statische Methoden und Attribute unterstreichen**

<b>Name der Klasse</b>
Attribute
Methoden

## ■ UML

### ■ -

- private
- von Instanzen derselben Klasse sichtbar (**aber von allen!**)

### ■ #

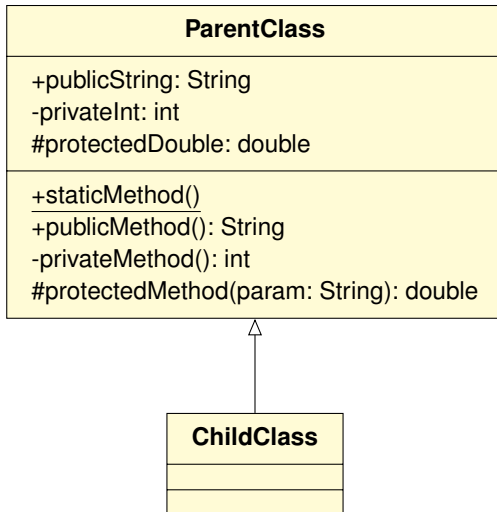
- protected (wie in Java)
- von Instanzen derselben Klasse, aller Unterklassen und Instanzen aus dem gleichen Paket sichtbar

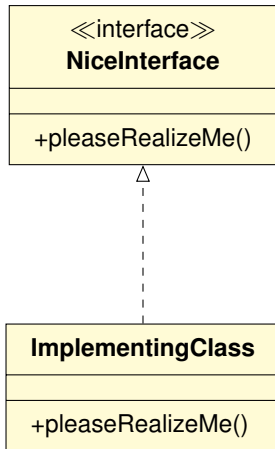
### ■ +

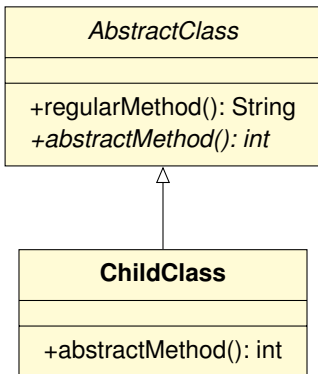
- public (wie in Java)
  - von Instanzen jeder Klasse sichtbar
- falls nichts angegeben implizit public

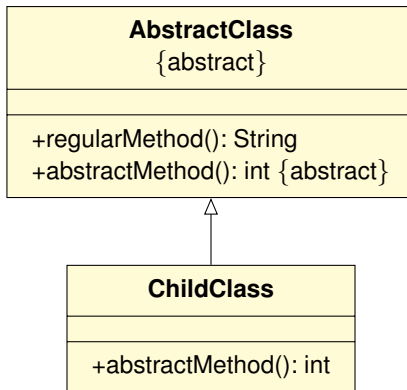


Hund
<ul style="list-style-type: none"><li>- name: String</li><li>- rasse: Rasse</li><li>- gewicht: int</li></ul>
<ul style="list-style-type: none"><li>+ wiegen(): int</li><li>+ streicheln()</li><li>+ streicheln(intensität: int, ausruf: String)</li><li>+ füttern(ration: Nahrung)</li></ul>

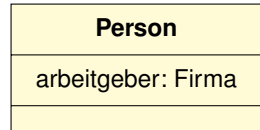
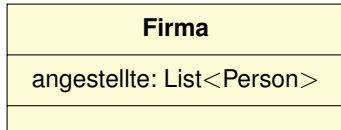








- für Übungsblätter und Klausur
  - kursiv nicht erkennbar, stattdessen **{abstract}** verwenden
  - laut VL unter Klassenname, hinter Methode



Firma
angestellte: List<Person>

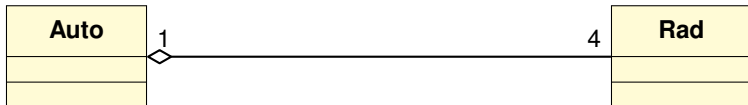
Person
arbeitgeber: Firma

## Probleme

- List<X> ist Java-Syntax und schreibt Datenstruktur vor
- Beziehungen sollen direkt ersichtlich werden
- Faustregel: nur primitive Typen als Attribute hinschreiben

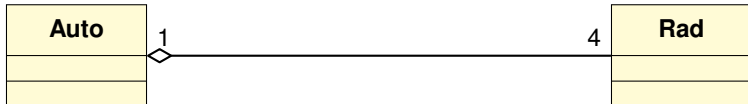


- Aggregation = Teil-Ganzes-Beziehung

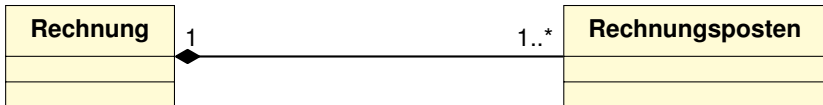




- Aggregation = Teil-Ganzes-Beziehung



- Komposition: Aggregation, aber Teil kann ohne Ganzes nicht existieren
  - wenn ganzes gelöscht wird, dann auch Teile!



## Text $\Rightarrow$ UML-Klassendiagramm

Jeder Student hat eine Matrikelnummer und einen Namen. Ein fauler Student ist ein Student, der schlafen kann. Er hat dazu ein Bett. Ein fleißiger Student hingegen, kann lernen und hat dazu einen Computer, der aus Bauteilen besteht.

## Text $\implies$ UML-Klassendiagramm

Jeder Student hat eine Matrikelnummer und einen Namen. Ein fauler Student ist ein Student, der schlafen kann. Er hat dazu ein Bett. Ein fleißiger Student hingegen, kann lernen und hat dazu einen Computer, der aus Bauteilen besteht.

## UML-Diagramm?

## Text $\Rightarrow$ UML-Klassendiagramm

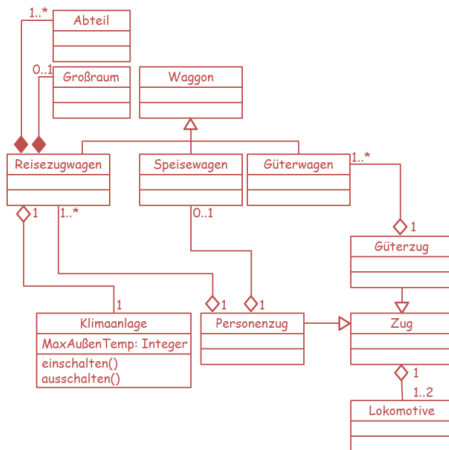
Jeder **Student** hat eine Matrikelnummer und einen Namen. Ein fauler Student **ist ein** Student, der schlafen kann. Er **hat** dazu ein Bett. Ein fleißiger Student hingegen, kann lernen und **hat** dazu einen **Computer**, der aus **Bauteilen besteht**.

**Schlüsselwörter!**

*Modellieren Sie das Szenario möglichst vollständig als UML-Klassendiagramm. Geben Sie Methoden, Attribute, Multiplizitäten, Restriktionen, Assoziationsnamen, Aggregationen und Kompositionen sowie Rollen an.*

## Szenario

Ein Güterzug ist ein Zug, dessen Waggon ausschließlich Güterwagen sind. Die Waggon eines Personenzugs sind mindestens ein Reisezugwagen und höchstens ein Speisewagen. Jeder Zug hat eine oder zwei Lokomotiven. Reisezugwagen setzen sich aus bis zu einem Großraum sowie einem oder mehreren Abteilen zusammen. Jeder Reisezugwagen hat eine Klimaanlage, die ein- und ausgeschaltet werden kann. Jede Klimaanlage darf nur bis zu einer bestimmten maximalen Außentemperatur betrieben werden.

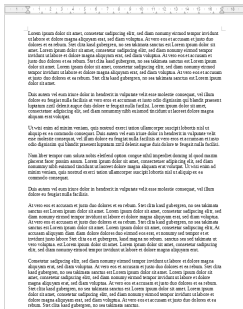


Klassen	11x	0,5 P
Methoden/Attribute "Klimaanlage"	1x	0,5 P
Aggregationen/Kompositionen	7x	0,5 P
Vererbung (alle 3 zusammen)	1x	0,5 P

- auf dem Blatt müsst ihr L<sup>A</sup>T<sub>E</sub>X für die Dokumente benutzen
- wird euch an der Uni immer wieder begegnen, manchmal Pflicht

- auf dem Blatt müsst ihr L<sup>A</sup>T<sub>E</sub>X für die Dokumente benutzen
- wird euch an der Uni immer wieder begegnen, manchmal Pflicht
- nicht WYSIWYG
  - What You See Is What You Get
  - Paradigma von Word etc.

## (a) What You See



## (b) What You Get





- stattdessen WYSIWYAF bzw. WYSIWYM
  - What You See Is What You Ask For / Mean
  - Paradigma von L<sup>A</sup>T<sub>E</sub>X  $\approx$  HTML und CSS


## (a) What You See

```
\subsection{Pflichtenheft - Gliederung}
\begin{frame}{Pflichtenheft - Gliederung}
\begin{enumerate}
\item Zielbestimmung
\item Produkteinsatz
\item \underline{\textbf{Produktumgebung}} (Hard-/Software in \textit{Einsatzumgebung})
\item Funktionale Anforderungen
\item Produktdaten
\item Nichtfunktionale Anforderungen
\item \underline{\textbf{Globale Testfälle}} (\enquote{zu testende Abläufe})
\item Systemmodelle
\begin{itemize}
\item Szenarien
\item Anwendungsfälle
\item \underline{\textbf{Objektmodelle}} $\implies$ UML-Klassendiagramme (heute)
\item \underline{\textbf{Dynamische Modelle}} $\implies$ nächstes Mal
\item \underline{\textbf{Benutzerschnittstelle}} $\implies$ Zeichnungen/Screenshots
\end{itemize}
\item Glossar
\end{enumerate}
\end{frame}
```

## (b) What You Mean

### Pflichtenheft - Gliederung

- ① Zielbestimmung
- ② Produkteinsatz
- ③ **Produktumgebung** (Hard-/Software in Einsatzumgebung)
- ④ Funktionale Anforderungen
- ⑤ Produktdaten
- ⑥ Nichtfunktionale Anforderungen
- ⑦ **Globale Testfälle** („zu testende Abläufe“)
- ⑧ Systemmodelle
  - Szenarien
  - Anwendungsfälle
  - **Objektmodelle**  $\implies$  UML-Klassendiagramme (heute)
  - **Dynamische Modelle**  $\implies$  nächstes Mal
  - **Benutzerschnittstelle**  $\implies$  Zeichnungen/Screenshots
- ⑨ Glossar



Orga	Wasserfallmodell	Durchführbarkeitsuntersuchung	Lastenheft	Pflichtenheft	UML-Klassendiagramm	ITgX	Tipp
0000000	000	0	00	000	0000000000000000	0000000000000000	000
Felix Bachmann – SWT1						14.05.2019	15:38

## ■ Vorteile

- gut versionierbar
  - „Quellcode“ ist normales Textdokument
  - Word etc. verwenden oft XML-Formate mit Metadaten
- leicht Formeln erstellbar
- nach Eingewöhnung recht intuitiv
- multifunktional (Bücher, Dokumente, Präsentationen, ...)
- open source, kostet nix :)
- viele Erweiterungen, Pakete, ...

## ■ Nachteile

- Einarbeitung notwendig :(

Installation einer Distribution notwendig, z.B.:

- MiKTeX für Windows
- TeX Live für Linux, Mac, Windows

Installation einer Distribution notwendig, z.B.:

- MikTeX für Windows
- TeX Live für Linux, Mac, Windows

Editoren machen das Schreiben von L<sup>A</sup>T<sub>E</sub>X-Dokumenten angenehmer

- Texmaker
- TeXstudio (erweiterter Texmaker, mein Favorit)
- TeXclipse (Plugin für Eclipse)
- Plugins für Visual Studio Code
- ...

## Präambel: Pakete laden, Dokumenttyp festlegen

```
\documentclass{Klasse}  
\usepackage[option1,option2,...]{Paket}
```

- nützliche Klassen: book, beamer, scrartcl
- nützliches Paket z.B. csquotes (ermöglicht `\enquote{...}`)

## Präambel: Pakete laden, Dokumenttyp festlegen

```
\documentclass{Klasse}  
\usepackage[option1,option2,...]{Paket}
```

- nützliche Klassen: book, beamer, scrartcl
- nützliches Paket z.B. csquotes (ermöglicht `\enquote{...}`)

## Inhalt: Text setzen, Bilder, Graphiken, Formeln,...

```
% preamble  
\begin{document}  
    content  
\end{document}
```

- Struktur: part, (chapter), section, subsection, subsubsection
- Auflistungen: `\begin{itemize}` `\item Hello World!` `\end{itemize}`
- Bilder: `\includegraphics[scale = 0.8]{PfadZumBild}`

# L<sup>A</sup>T<sub>E</sub>X-Beispiel!

## Aufgabe 1 + 3: Lastenheft + Durchführbarkeitsuntersuchung

- lasst euch was (sinnvolles) einfallen
  - 6+3 Punkte für  $< 5$  Seiten sinnvollen Text!
    - besser wirds nicht
    - aber: nicht zu allgemein werden, Bezug zu Szenario, Form
- Anwendungsfalldiagramm: Syntax beachten
- Durchführbarkeitsuntersuchung: Fragen beantworten, nicht stellen!



## Aufgabe 1 + 3: Lastenheft + Durchführbarkeitsuntersuchung

- lasst euch was (sinnvolles) einfallen
  - 6+3 Punkte für < 5 Seiten sinnvollen Text!
    - besser wirds nicht
    - aber: nicht zu allgemein werden, Bezug zu Szenario, Form
- Anwendungsfalldiagramm: Syntax beachten
- Durchführbarkeitsuntersuchung: Fragen beantworten, nicht stellen!

## Aufgabe 2: Klassendiagramme

- Form, Syntax
- achtet auf Schlüsselwörter ("ist ein", "enthält ein", "besteht aus", ...)

## Aufgabe 4 + 5: HDrize

- HDR implementieren
  - aus Belichtungsreihe
- Zusammenhang der Klassen unklar? Vielleicht hilft Diagramm

## Aufgabe 4 + 5: HDrize

- HDR implementieren
  - aus Belichtungsreihe
- Zusammenhang der Klassen unklar? Vielleicht hilft Diagramm

## Aufgabe 6: Schnittstellen für Filter

- nur Schnittstellen definieren
- JavaDoc!

## Abgabe

- Deadline am 22.5 um 12:00
- Dokumente ausdrucken
- Klassendiagramme handschriftlich
- Deckblatt!



DEVELOPERS