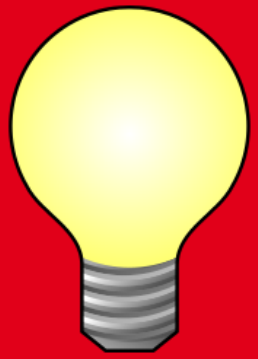
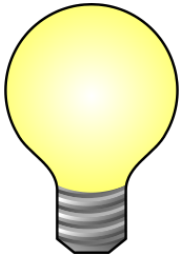
Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

21.01.2021
Programmieren im Grossen IV



Feinentwurf



Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

AGENDA

Einführung ins Thema

Vorgehensweise

Beispiel “Playlistenverwaltung”

Entwurfsmuster

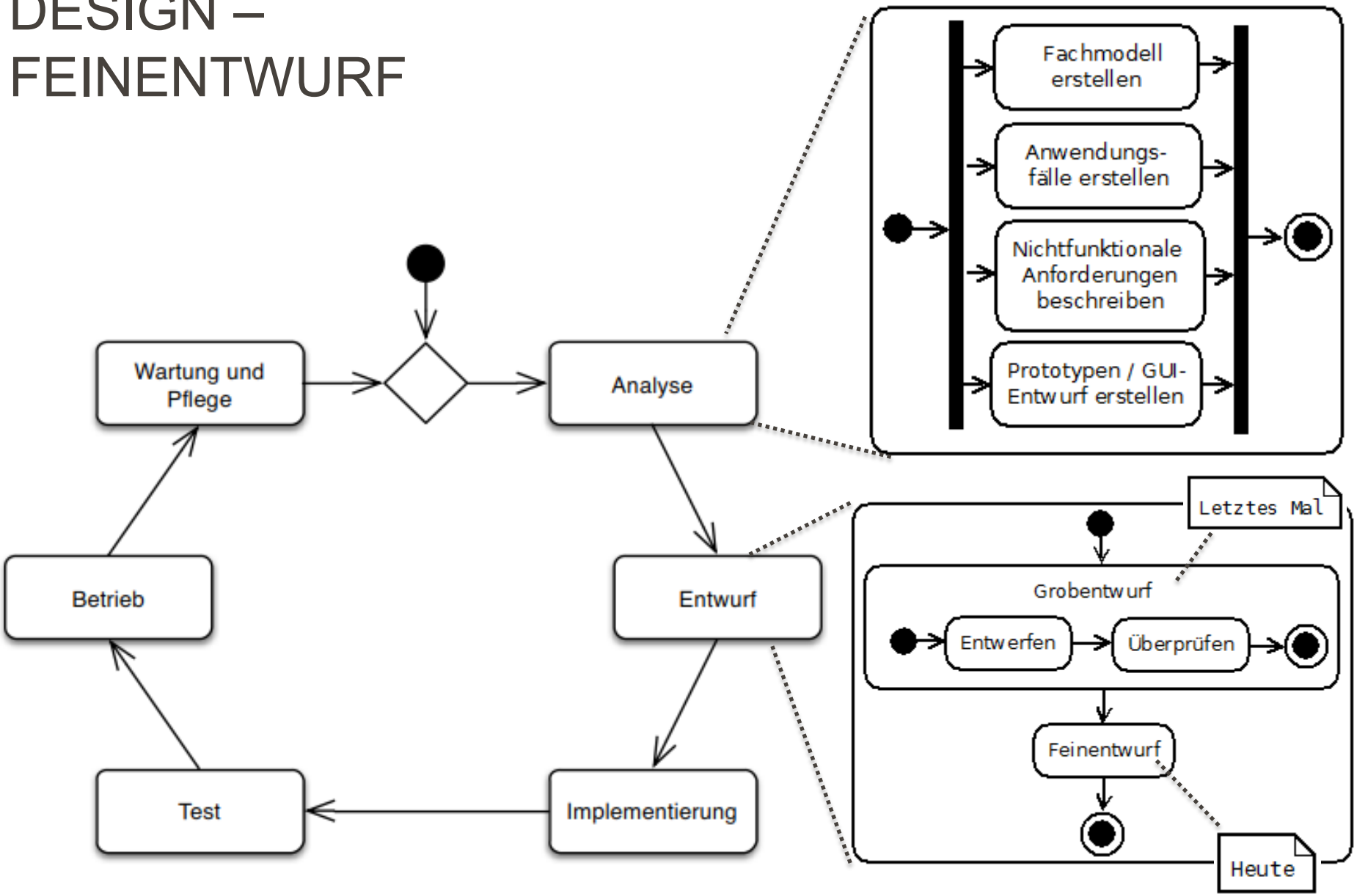
Fazit

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

01
EINFÜHRUNG INS THEMA

Ziel:
Die Eckpunkte des Themas kennenlernen

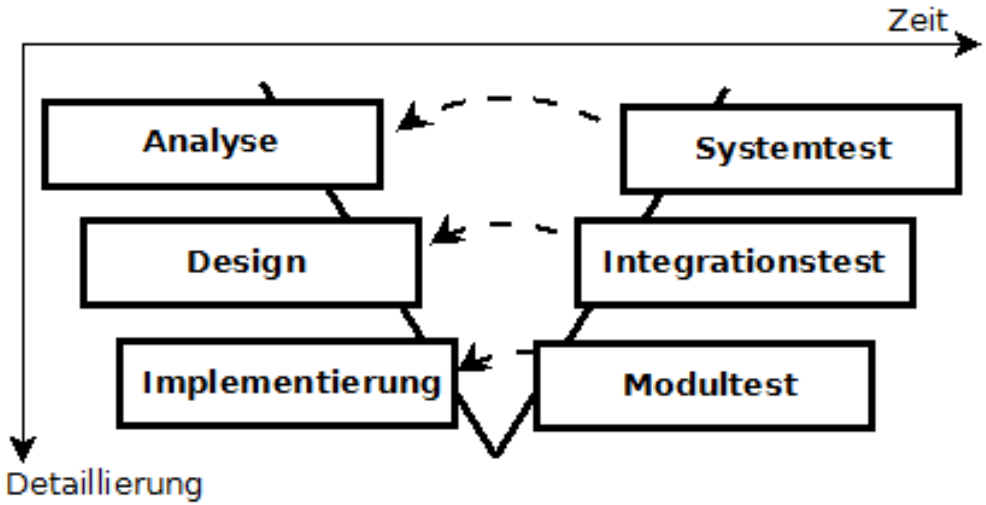
DESIGN – FEINENTWURF



WORUM GEHT‘S?

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Wir brauchen ein Vorgehensmodell
  - Verschiedene Phasen:



- Heute besprechen wir:
  - Design: Feinentwurf

ENTWURF (DESIGN) – WORUM GEHT‘S?

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Grundlegende Frage
  - Wie soll das zu bauende System sein?
- Entwurf (Design)
  - Tätigkeiten
    - Grobentwurf (Architektur) → letztes Mal
    - **Feinentwurf → heute**
  - Sprechweise im (R)UP:
    - „Analysis and Design“
  - Sprechweise im V-Modell:
    - „Design“

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

02
Vorgehensweise

Ziel:
Prinzipielle Vorgehensweise kennenlernen

FEINENTWURF – VORGEHENSWEISE

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Input:
  - Grobentwurf (inkl. GUI-Entwurf)
  - Resultate der Anforderungsanalyse
  - Erfahrung (typische, erprobte Lösungen)
- Tätigkeit: Grobentwurf (schrittweise) verfeinern
- Output:
  - Programmentwurf in Form von Klassendiagrammen
  + Spezifikation (zumindest grob/teilweise)
  + ggfs. Pseudocode
  + ggfs. Sequenz-, Aktivitäts-, . . . -Diagramme
  + ggfs. Datenbank-Tabellen/-Constraints
  + . . .

⇒ Ziel: Programmierer können direkt loslegen

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

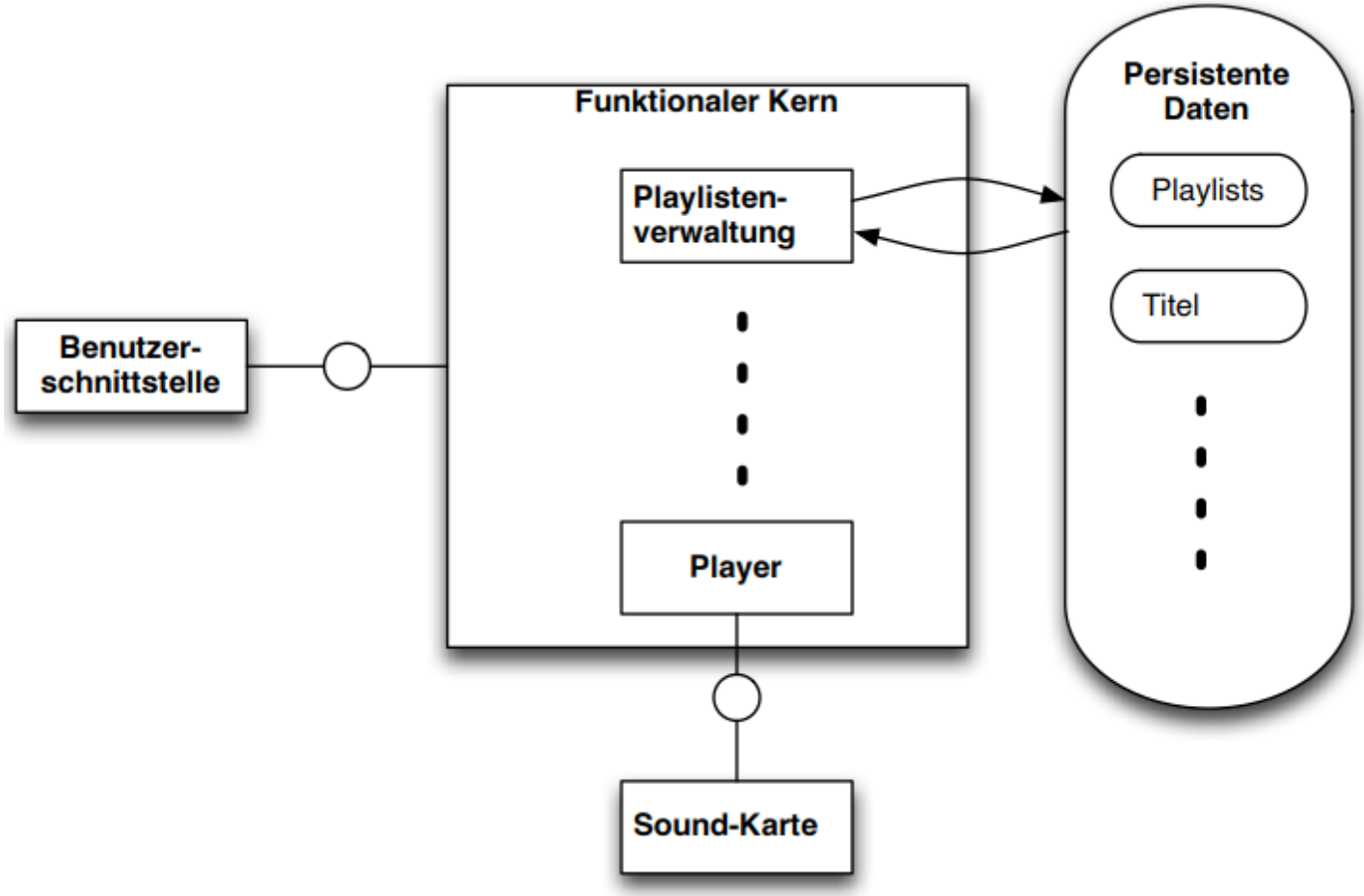
03
Beispiel „Playlistenverwaltung“

Ziel:
Ein konkretes Beispiel

BEISPIEL GROBENTWURF

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- BSP: Grobentwurf MP3-Player mit FMC:

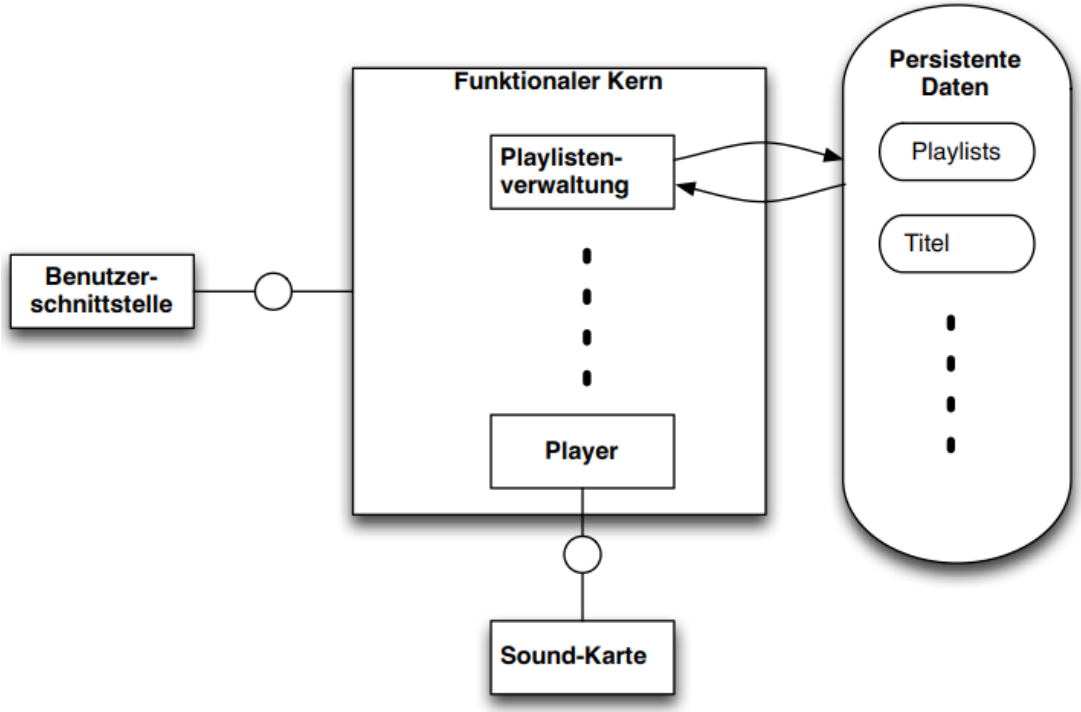


→ Mit Robustness Analysis überprüft

HINWEIS: WIE DEN GROBENTWURF UMSETZEN?

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Wie Grobentwurf mit FMC jetzt in Feinentwurf umsetzen?



→ Mehrere Möglichkeiten für jede FMC-Komponente:

- FMC-Komponente → Klasse(n), Objekt(e)
- FMC-Komponente → Attribut (Variable) einer Klasse
- FMC-Komponente → Methode(n) (Algorithmus), . . .

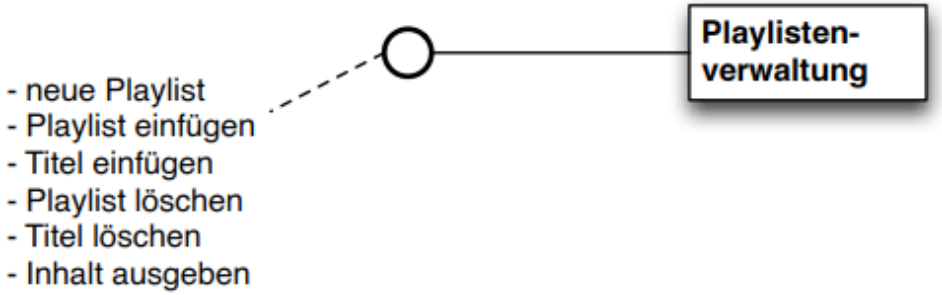
BEISPIEL FEINENTWURF „PLAYLISTENVERWALTUNG“

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

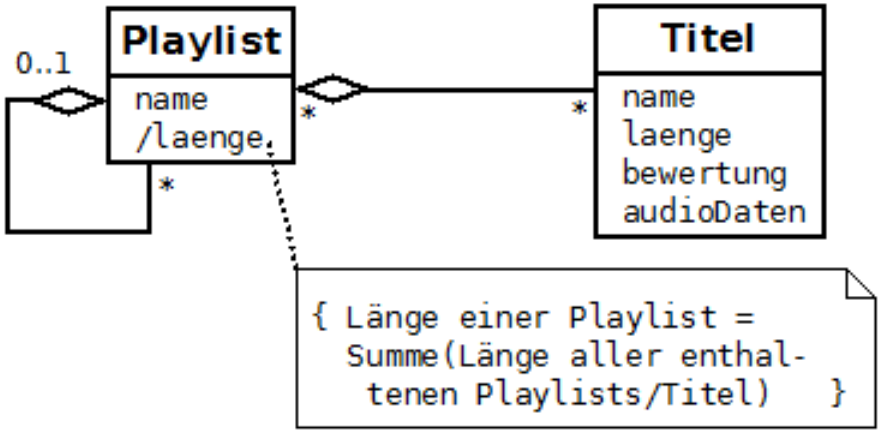
INPUT:

- Passender Auszug aus dem Grobentwurf

  → Grobe Schnittstellenbeschreibung für „Playlistenverw.“:



- Passender Auszug aus dem Fachmodell zur „Playlistenverw.“



- ggfs. weitere passende Auszüge aus den Anforderungen

BEISPIEL FEINENTWURF „PLAYLISTENVERWALTUNG“

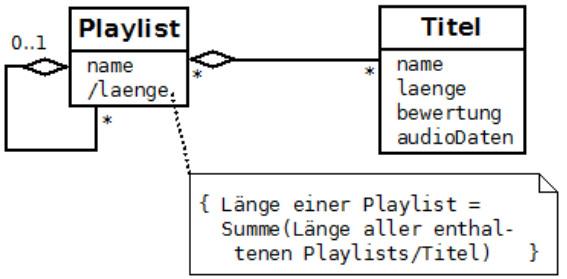
Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

OUTPUT: Detailliertere Schnittstelle zur „Playlistenverwaltung“

Was benötigen wir?

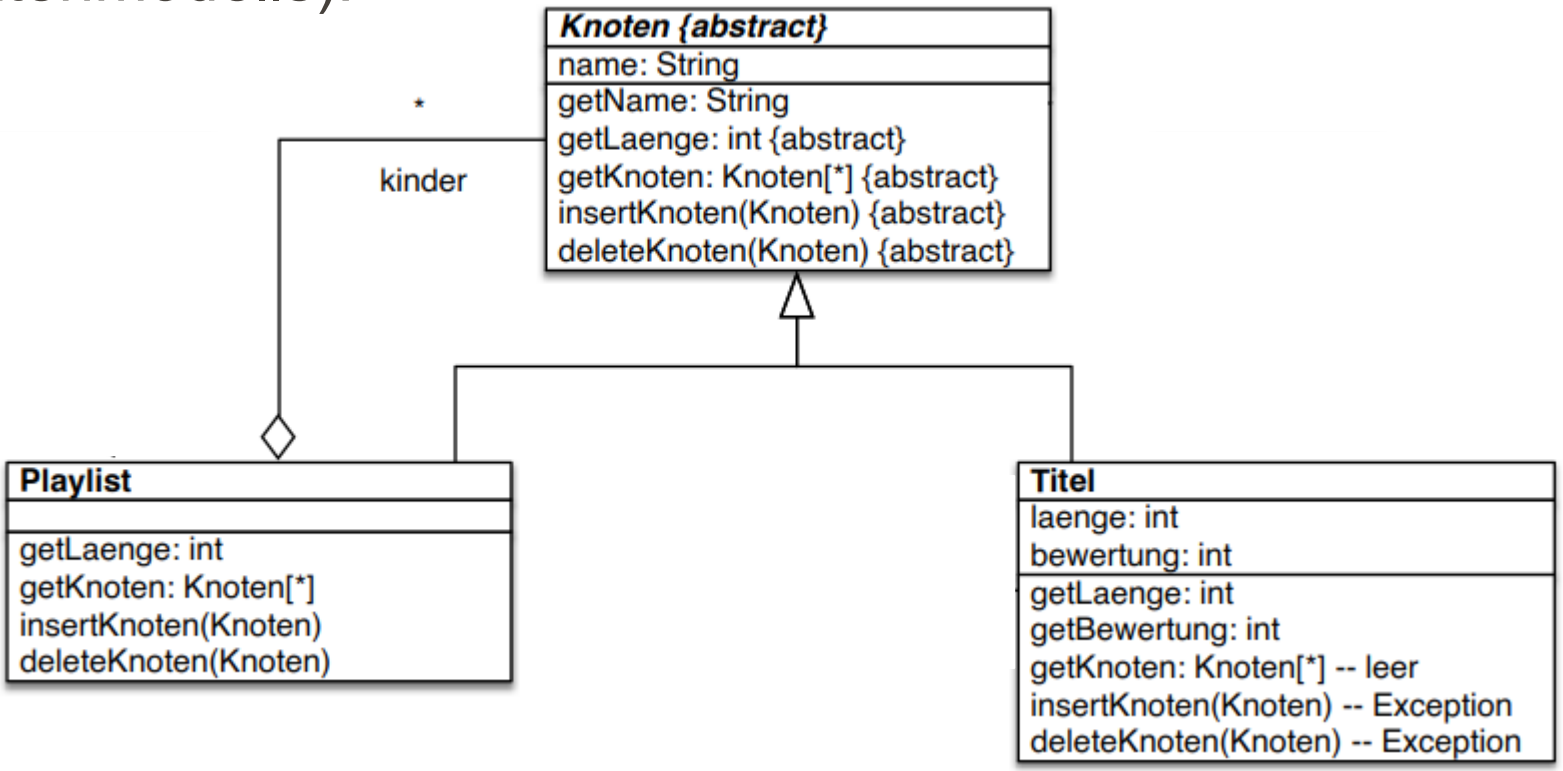
- Baum-Struktur mit verschiedenartigen Knoten
  - Playlist
  - Titel
- Navigation, zumindest: Von oben nach unten
- Verwaltung, zumindest:
  - Knoten einfügen
  - Knoten löschen
- Delegieren von Aufgaben nach unten
  - Obere Knoten (Elternknoten) erledigen Aufgaben mit Hilfe der untergeordneten Knoten
  - Letzte Knoten (Blätter) müssen Aufgaben alleine erledigen

BEISPIEL FEINENTWURF „PLAYLISTENVERWALTUNG“



OUTPUT: Detailliertere Schnittstelle zur „Playlistenverwaltung“

→ (Unvollständiger) Feinentwurf des Domänenmodells (Datenmodells):

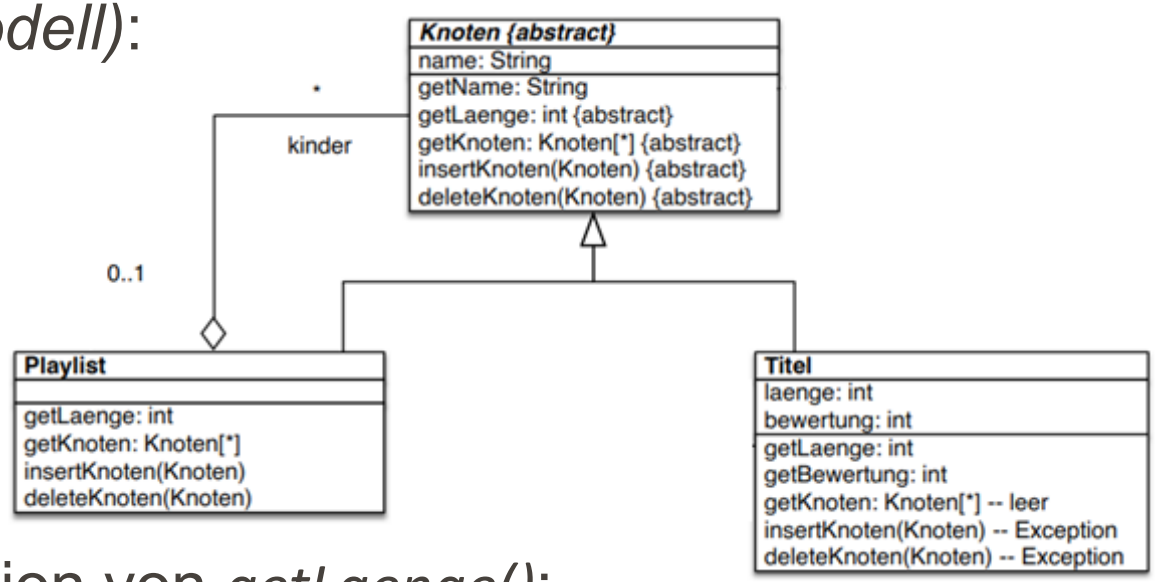


BEISPIEL FEINENTWURF „PLAYLISTENVERWALTUNG“

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

OUTPUT: Detailliertere Schnittstelle zur „Playlistenverwaltung“

- (Unvollständiger) Feinentwurf des *Domänenmodells (Datenmodell)*:



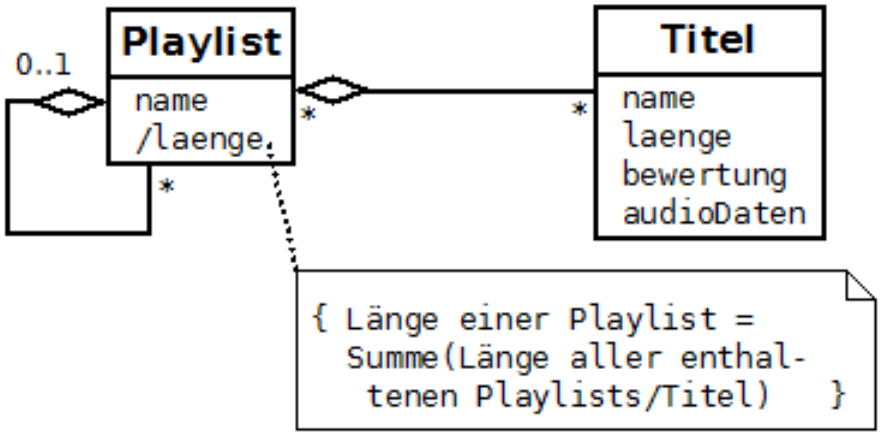
- Spezifikation von *getLaenge()*:
  - *Titel.getLaenge()*: gibt Länge des Titels zurück
  - *Playlist.getLaenge()*: gibt Summe von *getLaenge()* aller Unterknoten zurück (leere Summe=0)
- . . .

(Siehe delegieren von Aufgaben zwei Folien voher)

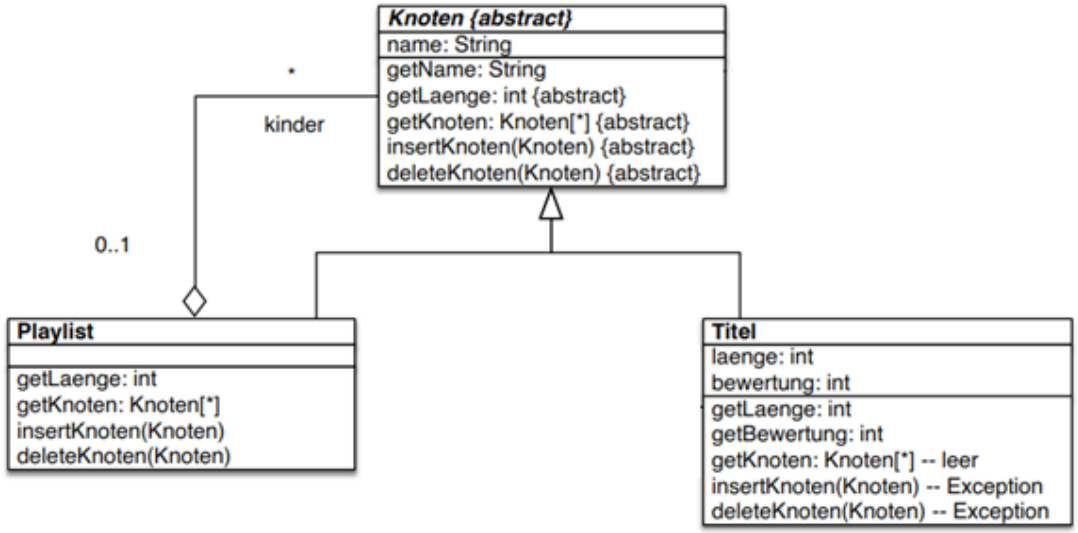
NOCHMALS ZUM VERGLEICH

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- *Fachmodell* aus der Analyse (rein aus Anforderungen):



- *Domänenmodell* (Datenmodell) im Feinentwurf:



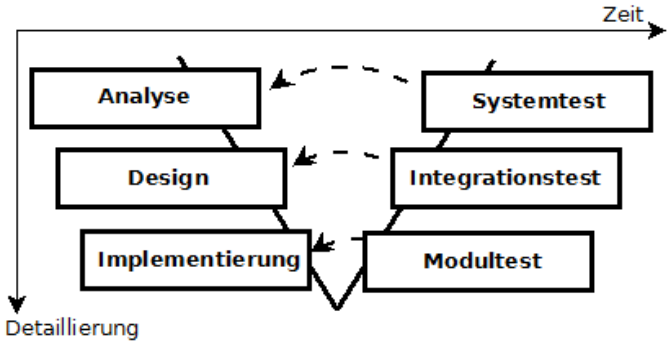
WEITERE BEMERKUNGEN ZUM FEINENTWURF

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Wichtige Frage: Wann hört man auf?
  - D.h.: Was modelliert man? ↔ Was programmiert man?
- Dazu gibt es leider keine einfache Antwort
  - Auch etwas eine Idealvorstellung der durchgängigen Modellierung
  - Hängt davon ab wie durchgängig die Werkzeugkette ist
    - Wenn enge Verzahnung vorhanden, kann man sehr detailliert durchmodellieren
    - Ansonsten Gefahr, dass Modell und Code schnell auseinanderdriften
  - In vielen Projekten: Eher rudimentäre Modellierung der Eckdaten → Rest wird programmiert

→ Jedes Projekt muss hier selbst die richtige Dosis finden

TESTVORBEREITUNG IM FEINENTWURF



- Vorbereitung des Integrationstests:
  - Input: Feinentwurf
    - Welche Klassen gibt es?
    - Wie kommunizieren diese miteinander?
  - Output:
    - Ansatz (bottom-up, top-down, . . .)
    - Mindestens ein Testfall für jede Verbindung

- Vorbereitung des Modultests:
  - Black-Box-Testfälle für die Module (Klassen)

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

04
Entwurfsmuster

Ziel:
Lösungen für wiederkehrende Probleme kennen

WIEDERKEHRENDE AUFGABEN-STELLUNGEN IM FEINENTWURF

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

Aus unserem vorherigen Beispiel:

Wir benötigen:

- Baum-Struktur mit verschiedenartigen Knoten
- Navigation, zumindest: von oben nach unten
- Verwaltung, zumindest: Knoten einfügen, löschen
- Delegieren von Aufgaben nach unten

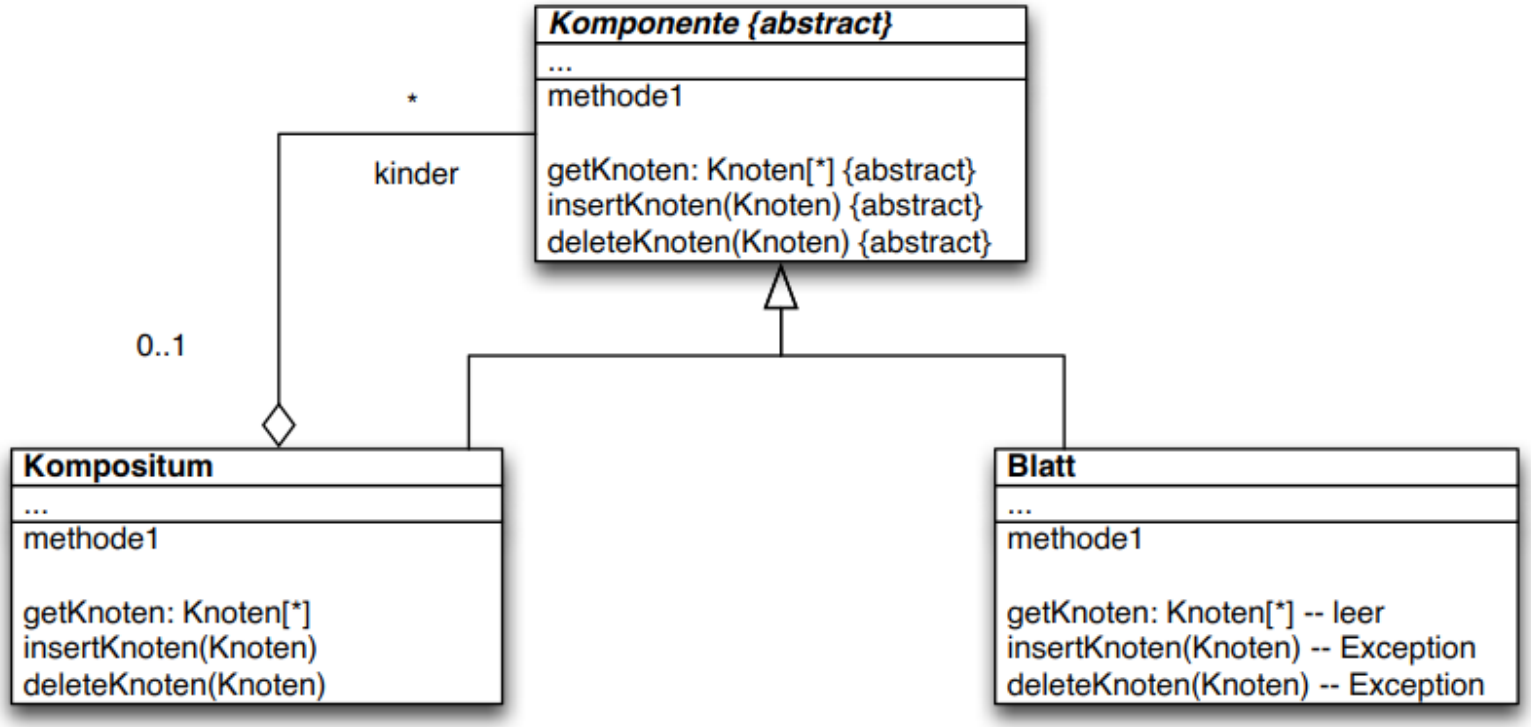
→ Solche oder sehr ähnliche Aufgaben werden in vielen Situationen gebraucht – Z.B.:

- Dateisystem,
- Evtl.: Unsere Programmieraufgabe im 2. Semester

ZUGEHÖRIGE, TYPISCHE, ERPROBTE LÖSUNG:

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Wir setzen das im allgemeinen Fall so um:



Kompositum.methode1: delegiert Aufgabe an Unterknoten

→ Diese immer wiederkehrende Aufgabenstellung + diese typische, erprobte Lösung (+ zusätzliche Beschreibungen)

wird *Entwurfsmuster „Kompositum“* genannt

DEFINITION & ARTEN VON MUSTERN

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Muster (in der Software-Entwicklung)
  = typische, erprobte Lösung für ein immer wiederkehrendes Problem in einem bestimmten Kontext
- Analysemuster (analysis pattern)
  = typische, erprobte Lösung für ein immer wiederkehrendes Problem beim Erstellen von Fachmodellen, etc.
- Architekturmuster (architectural patten)
  = typische, erprobte Lösung für ein immer wiederkehrendes Problem beim Grobentwurf
- **Entwurfsmuster (design pattern)**
  = typische, erprobte Lösung für ein immer wiederkehrendes Problem beim Feinentwurf
- …

MUSTER-BESCHREIBUNGEN

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

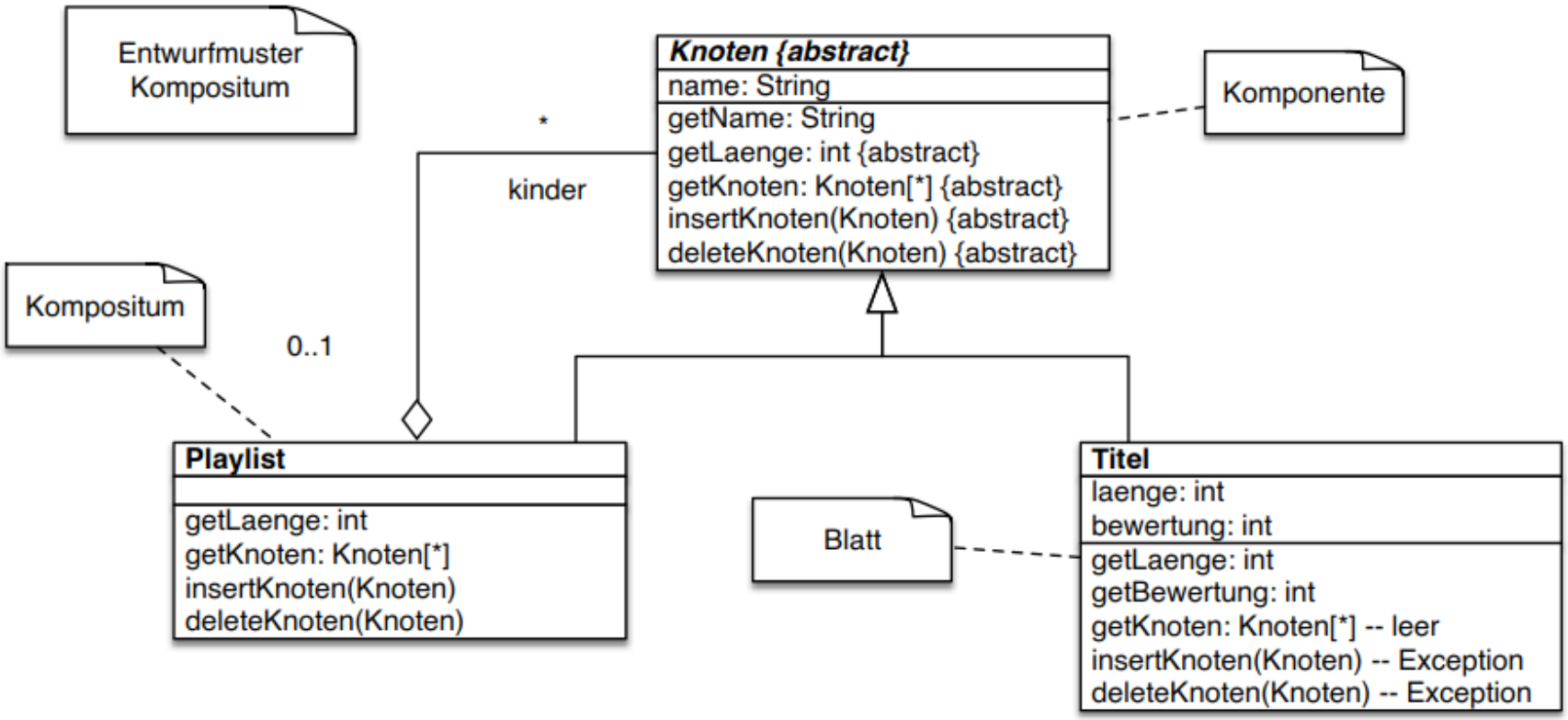
- Übliche Form für die Beschreibung eines Entwurfsmusters:
  1. Name + andere Namen
  2. Beispiel
  3. Kontext
  4. Problem
  5. Lösung
  6. Vor- und Nachteile
  7. Verwandte Muster
- Musterkataloge
  - Sammlung von Mustern + deren Zusammenwirken (Unterstützend, Widersprechend, Neutral)
  - BSP: Entwurfsmuster von Gamma et al. (→ Literaturverzeichnis) „Gang of Four (GoF)-Muster“

KENNZEICHNUNG EINES VER-WENDETEN ENTWURFSMUSTERS

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

BSP: „Kompositum“-Muster in unserem Domänenmodell

- Kennzeichnung durch Kommentare:



- BEM: Entwurfsmuster und die Rollen der beteiligten Klassen kann man in UML auch mit dem Modell-Element „Kollaboration“ kennzeichnen.

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

06
Fazit

Ziel:
Was haben wir damit gewonnen?

WAS HABEN WIR GELERNT?

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Feinentwurf
  - Detaillierung des Grobentwurfs
  - Besonders wichtig: Das Domänenmodell
- Entwurfsmuster
  - Bieten bewährte Lösungen für wiederkehrende Probleme
  - Heute: Kompositum

  → Nächste Woche: Die Musteridee allgemein

WEITERFÜHRENDE LITERATUR

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Object-Oriented Analysis and Design:
  - Kleuker: Grundkurs Software-Engineering mit UML [http://dx.doi.org/10.1007/978-3-8348-9843-2]
  - Zuser et al: Software-Engineering mit UML und dem Unified Process [BF 500 92]
  - C. Larman: Applying UML and Patterns [30 BF 500 78]

- Entwurfsmuster:
  - Gamma et al.: Entwurfsmuster [30 BF 500 40].
    → Der Klassiker aus den 90-igern. Ganz gut zum Nachschlagen. Trocken zu lesen.
  - Freeman et al: Entwurfsmuster von Kopf bis Fuß [30 BF 500 98]
    → Schöne, umfassende Einführung. Witzig geschrieben.

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

AUF GEHT‘S!!

SELBER MACHEN UND LERNEN!!