Lecture 1

SE1 – Moderne Programmierkonzepte

Prof. Dr. Maximilian Scherer

maximilian.scherer@dhbw-mannheim.de

SoSe 2022

Herzlich Willkommen

- Modul Software-Engineering 1

Herzlich Willkommen

- Modul Software-Engineering 1
- Einheit **Moderne Programmierkonzepte**

Herzlich Willkommen

- Modul Software-Engineering 1
- Einheit **Moderne Programmierkonzepte**

Herzlich Willkommen

- Modul Software-Engineering 1
- Einheit **Moderne Programmierkonzepte**

Vorstellungsrunde

Ihr Name und ein uninteressanter Aspekt zu Ihrer Person

Ziel der Veranstaltung

Ziel der Veranstaltung

1. Programmierparadigmen und Programmiersprachen

Ziel der Veranstaltung

1. Programmierparadigmen und Programmiersprachen
2. Funktionale Programmierung

Ziel der Veranstaltung

1. Programmierparadigmen und Programmiersprachen
2. Funktionale Programmierung
3. Stream Programmierung

Ziel der Veranstaltung

1. Programmierparadigmen und Programmiersprachen
2. Funktionale Programmierung
3. Stream Programmierung
4. Introspection / Reflection

Ziel der Veranstaltung

1. Programmierparadigmen und Programmiersprachen
2. Funktionale Programmierung
3. Stream Programmierung
4. Introspection / Reflection
5. GUI Grundlagen

Ziel der Veranstaltung

1. Programmierparadigmen und Programmiersprachen
2. Funktionale Programmierung
3. Stream Programmierung
4. Introspection / Reflection
5. GUI Grundlagen
6. Reguläre Ausdrücke

Ziel der Veranstaltung

1. Programmierparadigmen und Programmiersprachen
2. Funktionale Programmierung
3. Stream Programmierung
4. Introspection / Reflection
5. GUI Grundlagen
6. Reguläre Ausdrücke
7. Visual Programming

Organisatorisches

Für Präsenzveranstaltungen benötigen Sie:

- Laptop

Für Präsenzveranstaltungen benötigen Sie:

- Laptop
- WLAN Zugang

Für Onlineveranstaltungen benötigen Sie:

- Tastatur

Für Onlineveranstaltungen benötigen Sie:

- Tastatur
- Mikrofon

Für Onlineveranstaltungen benötigen Sie:

- Tastatur
- Mikrofon
- ggf. Webcam

Für Onlineveranstaltungen benötigen Sie:

- Tastatur
- Mikrofon
- ggf. Webcam
- stabile Internetverbindung

- Slides kurz vor/nach der Veranstaltung über Moodle

- Slides kurz vor/nach der Veranstaltung über Moodle
- Mitschreiben?

- Slides kurz vor/nach der Veranstaltung über Moodle
- Mitschreiben?
  - Aufmerksamkeit

- Slides kurz vor/nach der Veranstaltung über Moodle
- Mitschreiben?
  - Aufmerksamkeit
  - Notizen machen

- Slides kurz vor/nach der Veranstaltung über Moodle
- Mitschreiben?
  - Aufmerksamkeit
  - Notizen machen
  - Fragen formulieren

- Slides kurz vor/nach der Veranstaltung über Moodle
- Mitschreiben?
  - Aufmerksamkeit
  - Notizen machen
  - Fragen formulieren
- Pausen

Moodle-Raum

- Moodle Kursraum:

  `https://moodle.dhbw-mannheim.de/course/view.php?id=9072`

Moodle-Raum

- Moodle Kursraum:

  `https://moodle.dhbw-mannheim.de/course/view.php?id=9072`

- Schlüssel: **dd83hd3**

- Vorlesung (30 VE)

- Vorlesung (30 VE)
- Selbststudium (45 VE)

- Vorlesung (30 VE)
- Selbststudium (45 VE)
  - Nachbereitung

- Vorlesung (30 VE)
- Selbststudium (45 VE)
  - Nachbereitung
  - Vorbereitung

- Vorlesung (30 VE)
- Selbststudium (45 VE)
  - Nachbereitung
  - Vorbereitung
  - Portfolioprüfung

Portfolioprüfung

- Mini-Projekt mit 60 Punkten

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung
  - One-Pager: Vorlesungsbezug und Referenzen

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung
  - One-Pager: Vorlesungsbezug und Referenzen
  - Vortrag / Demo

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung
  - One-Pager: Vorlesungsbezug und Referenzen
  - Vortrag / Demo
  - etwa 24h pro Person

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung
  - One-Pager: Vorlesungsbezug und Referenzen
  - Vortrag / Demo
  - etwa 24h pro Person
- Gruppenarbeit (2-4 Studierende)

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung
  - One-Pager: Vorlesungsbezug und Referenzen
  - Vortrag / Demo
  - etwa 24h pro Person
- Gruppenarbeit (2-4 Studierende)
- Bewertungskriterien

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung
  - One-Pager: Vorlesungsbezug und Referenzen
  - Vortrag / Demo
  - etwa 24h pro Person
- Gruppenarbeit (2-4 Studierende)
- Bewertungskriterien
  - Funktionierender, kommentierter Code

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung
  - One-Pager: Vorlesungsbezug und Referenzen
  - Vortrag / Demo
  - etwa 24h pro Person
- Gruppenarbeit (2-4 Studierende)
- Bewertungskriterien
  - Funktionierender, kommentierter Code
  - Bezug zur Vorlesung und Referenzen

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung
  - One-Pager: Vorlesungsbezug und Referenzen
  - Vortrag / Demo
  - etwa 24h pro Person
- Gruppenarbeit (2-4 Studierende)
- Bewertungskriterien
  - Funktionierender, kommentierter Code
  - Bezug zur Vorlesung und Referenzen
  - Demo / Vortrag, Fragerunde

Portfolioprüfung

- Mini-Projekt mit 60 Punkten
  - Implementierung
  - One-Pager: Vorlesungsbezug und Referenzen
  - Vortrag / Demo
  - etwa 24h pro Person
- Gruppenarbeit (2-4 Studierende)
- Bewertungskriterien
  - Funktionierender, kommentierter Code
  - Bezug zur Vorlesung und Referenzen
  - Demo / Vortrag, Fragerunde
- Vortrag 26./27.7. und Abgabe 7.8.

Mini-Projekt Themen

Mini-Projekt Themen

- Beliebiges Programmierprojekt mit Fokus auf 1-2 Vorlesungsinhalte

Mini-Projekt Themen

- Beliebiges Programmierprojekt mit Fokus auf 1-2 Vorlesungsinhalte
- Im One-Pager Bezug darstellen

Mini-Projekt Themen

- Beliebiges Programmierprojekt mit Fokus auf 1-2 Vorlesungsinhalte
- Im One-Pager Bezug darstellen
- Eigene Ideen (kurze Absprache) gerne möglich

Mini-Projekt Themen

- Beliebiges Programmierprojekt mit Fokus auf 1-2 Vorlesungsinhalte
- Im One-Pager Bezug darstellen
- Eigene Ideen (kurze Absprache) gerne möglich
- ideenauswahl.txt -> nächstes Mal

Programmiersprachen

"Knowing many languages makes you a better person. Knowing many programming languages makes you a better programmer."

— Bjarne Stroustrup

Brainstorming

Java Eigenschaften

Unterteilung von Programmiersprachen

- Generation

Unterteilung von Programmiersprachen

- Generation
- Programmierparadigma

Unterteilung von Programmiersprachen

- Generation
- Programmierparadigma
- Typisierung

Unterteilung von Programmiersprachen

- Generation
- Programmierparadigma
- Typisierung
- Kompiliert / Interpretiert

Unterteilung von Programmiersprachen

- Generation
- Programmierparadigma
- Typisierung
- Kompiliert / Interpretiert
- Speicherverwaltung

Unterteilung von Programmiersprachen

- Generation
- Programmierparadigma
- Typisierung
- Kompiliert / Interpretiert
- Speicherverwaltung
- Sonstiges

Unterteilung von Programmiersprachen

- Generation
- Programmierparadigma
- Typisierung
- Kompiliert / Interpretiert
- Speicherverwaltung
- Sonstiges
  - Templates / Generics

Unterteilung von Programmiersprachen

- Generation
- Programmierparadigma
- Typisierung
- Kompiliert / Interpretiert
- Speicherverwaltung
- Sonstiges
  - Templates / Generics
  - Metaprogrammierung

Unterteilung von Programmiersprachen

- Generation
- Programmierparadigma
- Typisierung
- Kompiliert / Interpretiert
- Speicherverwaltung
- Sonstiges
  - Templates / Generics
  - Metaprogrammierung
  - Introspection / Reflection

Unterteilung von Programmiersprachen

- Generation
- Programmierparadigma
- Typisierung
- Kompiliert / Interpretiert
- Speicherverwaltung
- Sonstiges
  - Templates / Generics
  - Metaprogrammierung
  - Introspection / Reflection
  - Syntax

Java Eigenschaften

- high-level Sprache, C++ like Syntax

Java Eigenschaften

- high-level Sprache, C++ like Syntax
- Compiling in Bytecode

Java Eigenschaften

- high-level Sprache, C++ like Syntax
- Compiling in Bytecode
- Intepretation des Bytecode

Java Eigenschaften

- high-level Sprache, C++ like Syntax
- Compiling in Bytecode
- Intepretation des Bytecode
- statisch typisiert

Java Eigenschaften

- high-level Sprache, C++ like Syntax
- Compiling in Bytecode
- Intepretation des Bytecode
- statisch typisiert
- rein objekt-orientiert

Java Eigenschaften

- high-level Sprache, C++ like Syntax
- Compiling in Bytecode
- Intepretation des Bytecode
- statisch typisiert
- rein objekt-orientiert
- garbage collected / automatic memory management

Java Eigenschaften

- high-level Sprache, C++ like Syntax
- Compiling in Bytecode
- Intepretation des Bytecode
- statisch typisiert
- rein objekt-orientiert
- garbage collected / automatic memory management
- Plattform unabhängig über JVM

Python Eigenschaften

- high-level Sprache, “lesbarer” englischer Syntax

Python Eigenschaften

- high-level Sprache, “lesbarer” englischer Syntax
- intepretiert

Python Eigenschaften

- high-level Sprache, “lesbarer” englischer Syntax
- intepretiert
- dynamisch typisiert

Python Eigenschaften

- high-level Sprache, “lesbarer” englischer Syntax
- intepretiert
- dynamisch typisiert
- gemischt objekt-orientiert / funktional

Python Eigenschaften

- high-level Sprache, “lesbarer” englischer Syntax
- intepretiert
- dynamisch typisiert
- gemischt objekt-orientiert / funktional
- garbage collected / automatic memory management

Python Eigenschaften

- high-level Sprache, “lesbarer” englischer Syntax
- intepretiert
- dynamisch typisiert
- gemischt objekt-orientiert / funktional
- garbage collected / automatic memory management
- Plattform unabhängig

C++ Eigenschaften

- “Mid-Level” Language

C++ Eigenschaften

- "Mid-Level" Language
- Write once, compile anywhere

C++ Eigenschaften

- "Mid-Level" Language
- Write once, compile anywhere
- Preprocessor

C++ Eigenschaften

- "Mid-Level" Language
- Write once, compile anywhere
- Preprocessor
- Unmanaged (new / delete)

C++ Eigenschaften

- "Mid-Level" Language
- Write once, compile anywhere
- Preprocessor
- Unmanaged (new / delete)
- References

C++ Eigenschaften

- "Mid-Level" Language
- Write once, compile anywhere
- Preprocessor
- Unmanaged (new / delete)
- References
- statisch Typisiert

C++ Eigenschaften

- "Mid-Level" Language
- Write once, compile anywhere
- Preprocessor
- Unmanaged (new / delete)
- References
- statisch Typisiert
- Mehrfachvererbung

C++ Eigenschaften

- "Mid-Level" Language
- Write once, compile anywhere
- Preprocessor
- Unmanaged (new / delete)
- References
- statisch Typisiert
- Mehrfachvererbung
- Compile-time templates

C++ Eigenschaften

- "Mid-Level" Language
- Write once, compile anywhere
- Preprocessor
- Unmanaged (new / delete)
- References
- statisch Typisiert
- Mehrfachvererbung
- Compile-time templates
- Unterschiedliche Standards (98,03,07,11,14,17,20)

Weitere Sprachen

- Assembly

Weitere Sprachen

- Assembly
- C#

Weitere Sprachen

- Assembly
- C#
- Javascript / Typescript

Weitere Sprachen

- Assembly
- C#
- Javascript / Typescript
- Go

Weitere Sprachen

- Assembly
- C#
- Javascript / Typescript
- Go
- Rust

Weitere Sprachen

- Assembly
- C#
- Javascript / Typescript
- Go
- Rust
- C

Weitere Sprachen

- Assembly
- C#
- Javascript / Typescript
- Go
- Rust
- C
- FORTRAN

Weitere Sprachen

- Assembly
- C#
- Javascript / Typescript
- Go
- Rust
- C
- FORTRAN
- R

Weitere Sprachen

- Assembly
- C#
- Javascript / Typescript
- Go
- Rust
- C
- FORTRAN
- R
- Matlab

Weitere Sprachen

- Assembly
- C#
- Javascript / Typescript
- Go
- Rust
- C
- FORTRAN
- R
- Matlab
- ...

Programmierparadigmen

- Was ist ein Programmierparadigma? Welche kennen Sie?

- Was ist ein Programmierparadigma? Welche kennen Sie?
  - Imperative (structured, procedural)

- Was ist ein Programmierparadigma? Welche kennen Sie?
  - Imperative (structured, procedural)
  - Object-oriented

- Was ist ein Programmierparadigma? Welche kennen Sie?
  - Imperative (structured, procedural)
  - Object-oriented
  - Declarative / Functional

- Was ist ein Programmierparadigma? Welche kennen Sie?
  - Imperative (structured, procedural)
  - Object-oriented
  - Declarative / Functional
  - Event-driven

- Was ist ein Programmierparadigma? Welche kennen Sie?
  - Imperative (structured, procedural)
  - Object-oriented
  - Declarative / Functional
  - Event-driven
  - Data-driven

- Was ist ein Programmierparadigma? Welche kennen Sie?
  - Imperative (structured, procedural)
  - Object-oriented
  - Declarative / Functional
  - Event-driven
  - Data-driven
  - `https://cs.lmu.edu/~ray/notes/paradigms/`

- Was ist ein Programmierparadigma? Welche kennen Sie?
  - Imperative (structured, procedural)
  - Object-oriented
  - Declarative / Functional
  - Event-driven
  - Data-driven
  - `https://cs.lmu.edu/~ray/notes/paradigms/`
- Fast alle modernen Programmiersprachen unterstützen mehrere Paradigmen

- Was ist ein Programmierparadigma? Welche kennen Sie?
  - Imperative (structured, procedural)
  - Object-oriented
  - Declarative / Functional
  - Event-driven
  - Data-driven
  - `https://cs.lmu.edu/~ray/notes/paradigms/`
- Fast alle modernen Programmiersprachen unterstützen mehrere Paradigmen
- objekt-orientierte Programmierung vs. funktionale Programmierung

- Was ist ein Programmierparadigma? Welche kennen Sie?
  - Imperative (structured, procedural)
  - Object-oriented
  - Declarative / Functional
  - Event-driven
  - Data-driven
  - https://cs.lmu.edu/~ray/notes/paradigms/
- Fast alle modernen Programmiersprachen unterstützen mehrere Paradigmen
- objekt-orientierte Programmierung vs. funktionale Programmierung
- https://www.youtube.com/watch?v=LnX3B9oaKzw (funcprog.mp4)

Functional Programming & Haskell - Computerphile

OOP vs. FP

- primäre Einheit

OOP vs. FP

- primäre Einheit
  - OOP: Dinge

OOP vs. FP

- primäre Einheit
  - OOP: Dinge
  - FP: Transformationen / Abbildungen

OOP vs. FP

- primäre Einheit
  - OOP: Dinge
  - FP: Transformationen / Abbildungen
- stateful (OOP) / stateless (FP)

OOP vs. FP

- primäre Einheit
  - OOP: Dinge
  - FP: Transformationen / Abbildungen
- stateful (OOP) / stateless (FP)
- side-effects (OOP) / no side-effects (FP)

OOP vs. FP

```
FUNC get_api_dat(api_key, auth_url, dat_url) {
  oauth2_token = web_request(auth_url,header=api_key);
  return web_request(data_url,header=oauth2_token);
}
```

OOP vs. FP

```
FUNC get_api_dat(api_key, auth_url, dat_url) {
  oauth2_token = web_request(auth_url,header=api_key);
  return web_request(data_url,header=oauth2_token);
}

CLASS Api {
  PRIVATE oauth2_token;
  PUBLIC Api(api_key,auth_url) {
    oauth2_token = Web.request(auth_url,header=api_key);
  }
  PUBLIC get_api_dat(dat_url) {
    return Web.request(dat_url,header=THIS.oauth2_token);
  }
}
```

Mini-lecture: Lambda-Ausdrücke

- λ-Kalkül

- λ-Kalkül
- `https://en.wikipedia.org/wiki/Lambda_calculus`

- λ-Kalkül
- https://en.wikipedia.org/wiki/Lambda_calculus
- Deklaration einer anonymen Funktion

- λ-Kalkül
- `https://en.wikipedia.org/wiki/Lambda_calculus`
- Deklaration einer anonymen Funktion
- `() -> 5`

- λ-Kalkül
- https://en.wikipedia.org/wiki/Lambda_calculus
- Deklaration einer anonymen Funktion
- () -> 5
- (x) -> x+5

- λ-Kalkül
- https://en.wikipedia.org/wiki/Lambda_calculus
- Deklaration einer anonymen Funktion
- () -> 5
- (x) -> x+5
- (x,y) -> x*y

Lambda – Beispiele

```
1 var arr = new ArrayList<Integer>(Arrays.asList(1,2,3))↩
    ;
2 //foreach arr element, execute function:
3 arr.forEach((x) -> {
4   System.out.println(x);
5 });
```

Lambda – Beispiele

```
1 var arr = new ArrayList<Integer>(Arrays.asList(1,2,3))↩
    ;
2 //foreach arr element, execute function:
3 arr.forEach((x) -> {
4   x += 1;
5   print(x);//?
6 });
```

Lambda – Beispiele

```
1 interface Ifunc
2 {
3   int twoparams(int a, int b);
4 }
5 Ifunc f = (int a, int b) -> {a*b};
6 f.twoparams(2,3);//result?
```

Lambda – Beispiele

```
1 //bind button click event
2 button.onClick(() -> {
3   System.out.printf("user clicked on " + button.name);
4 });
```

Lambda – Beispiele

```
1 var line1 = ConveyorBelt.Find("line1");
2 //bind changed event
3 line1.onIsTransportingChanged(changed -> {
4     if (changed.NewVal==true)
5       line1.setTargetBeltSpeed(3); //go
6     else
7       line1.setTargetBeltSpeed(0); //stop
8   System.out.printf("%s changed from %g -> %g\n", ↩
       changed.PropName, changed.OldVal, changed.NewVal)↩
       ;
9 });
```

Event-basierte Programmierung

- Programmier-Paradigma

Event-basierte Programmierung

- Programmier-Paradigma
- Kontrollfluss durch Events / Ereignisse gesteuert

Event-basierte Programmierung

- Programmier-Paradigma
- Kontrollfluss durch Events / Ereignisse gesteuert
- Events

Event-basierte Programmierung

- Programmier-Paradigma
- Kontrollfluss durch Events / Ereignisse gesteuert
- Events
  - Nutzer-Eingaben (insb. GUI)

Event-basierte Programmierung

- Programmier-Paradigma
- Kontrollfluss durch Events / Ereignisse gesteuert
- Events
  - Nutzer-Eingaben (insb. GUI)
  - Sensorwerte / Interrupts

Event-basierte Programmierung

- Programmier-Paradigma
- Kontrollfluss durch Events / Ereignisse gesteuert
- Events
  - Nutzer-Eingaben (insb. GUI)
  - Sensorwerte / Interrupts
  - Message-Passing

Event-basierte Programmierung

- Programmier-Paradigma
- Kontrollfluss durch Events / Ereignisse gesteuert
- Events
  - Nutzer-Eingaben (insb. GUI)
  - Sensorwerte / Interrupts
  - Message-Passing
  - Callbacks

Event-basierte Programmierung

- Programmier-Paradigma
- Kontrollfluss durch Events / Ereignisse gesteuert
- Events
  - Nutzer-Eingaben (insb. GUI)
  - Sensorwerte / Interrupts
  - Message-Passing
  - Callbacks
- Event-handler

Literaturverzeichnis

Literaturverzeichnis I