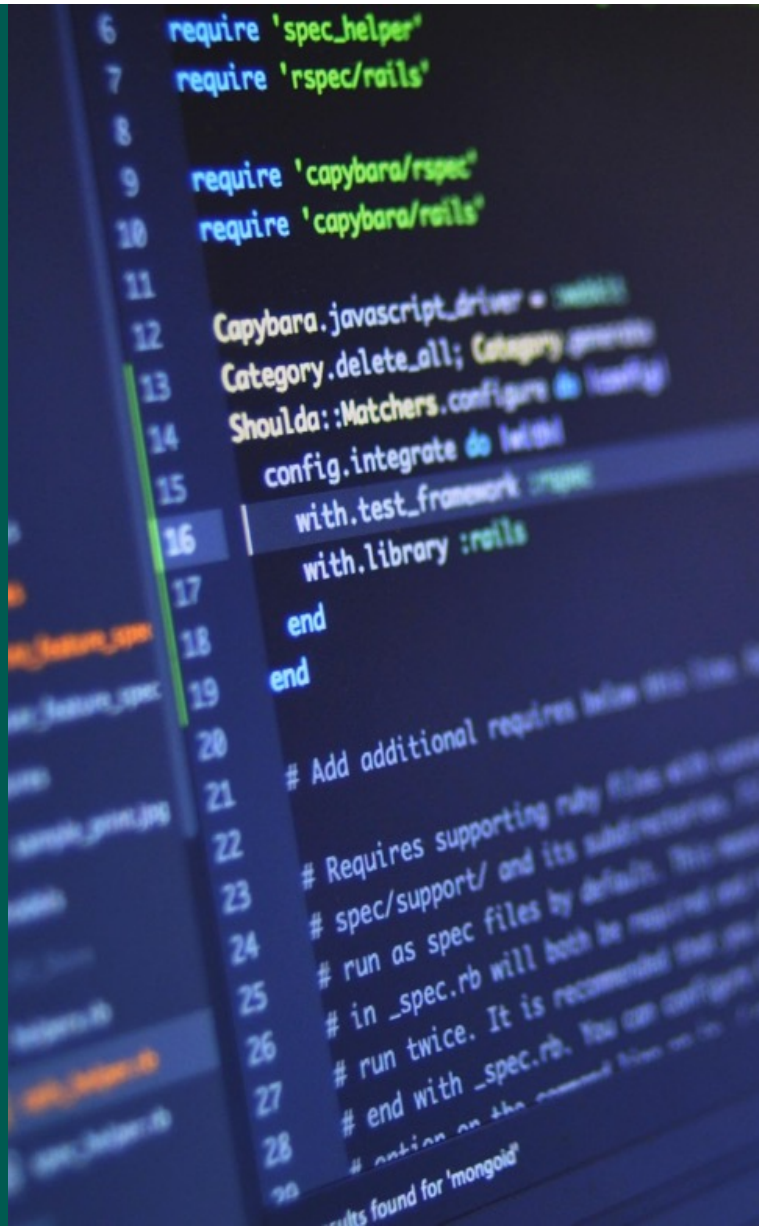
Info² : “Programmieren Best Practices " + Lernpartnerschafts börse"



KIT-Fakultät für Informatik
Informatik Studiengangservice

Dienstag, 20. Mai 2025 um 17:45 Uhr

1. Überblick
2. Best Practices für „Programmieren“
3. Überblick über das zweite Semester
4. Q&A

PBP für Sommersemester 2025

Das Programmieren-Team

Nils Niehues

Timur Sağlam

Haoyu, Liu

Robin Maisch

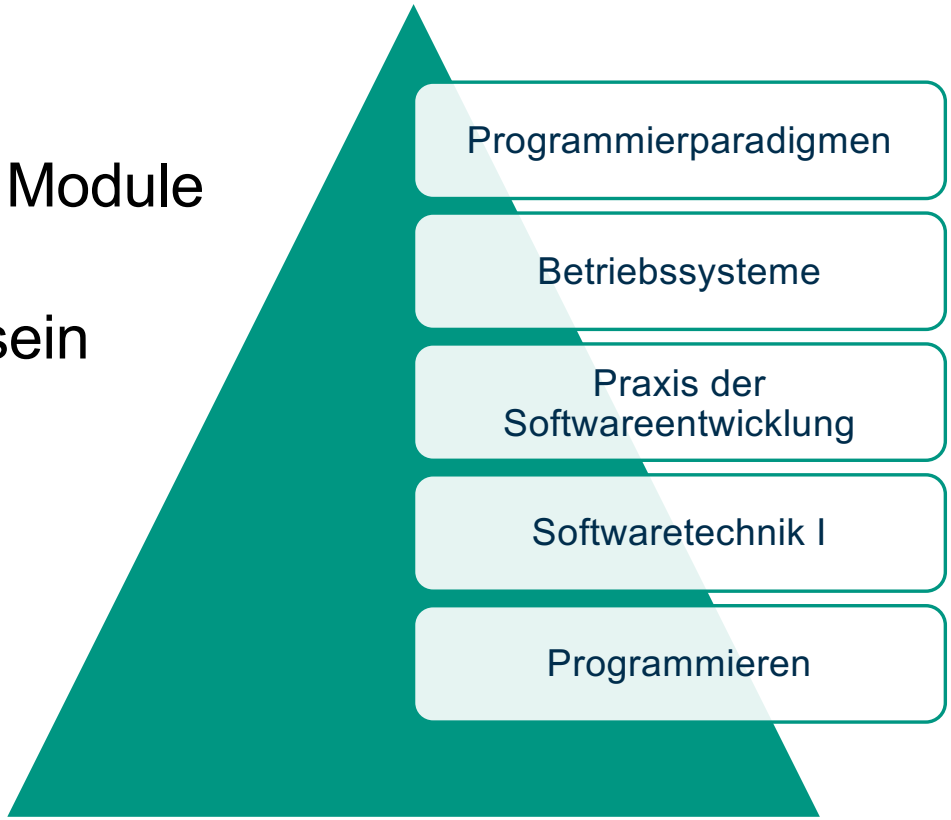
Best Practices für „Programmieren“

- Heutige Ziele:
  - Woran kann es scheitern?
  - Wie Frust sparen?
  - Wie beste Chancen haben

- Nicht alle Kriterien werden bei den ersten Blättern anwendbar sein!
  - Es muss nicht alles *sofort* verstanden werden
  - Aber spätestens bei den Abschlussaufgaben

Grundstudium: Praktische Informatik

- Programmieren…
  - …bildet die Grundlage für weitere Module
  - …ist Orientierungsprüfung
  - …sollte vor SWT abgeschlossen sein
- Was kann man mit guten Programmierkenntnissen tun?
  - Studentische Hilfskraft
  - Werkstudent
  - Open-Source-Projekte



Was Sie aus diesem Modul mitnehmen sollten

- Die Grundlagen der objektorientierten Programmierung in Java

- **Abbildung**: Problemmodellierung
  - Wie man alltägliche Probleme in Programmiersprache modelliert
- **Automatisierung**: Lösungsformulierung
  - Entwicklung von Verfahren (Algorithmen) zur Lösung einfacher Probleme
- **Abstraktion**: Problemformulierung
  - Wie man nicht nur eine Probleminstanz löst, sondern eine allgemeine Lösungsmethode findet, die für viele Probleminstanzen funktioniert
- **Sauber programmieren!**
  - Wichtig für Wartbarkeit, Wiederverwendbarkeit und Zusammenarbeit

Allgemeine Tipps zur Studienorganisation

- Termine und Fristen verfolgen
  - Bereits zu Beginn des Semesters veröffentlicht: s.kit.edu/programmieren
  - In den eigenen Kalender mit Erinnerung übernehmen
  - Keine nachträglichen Anmeldungen für Prüfungsleistungen

- Studien- und Prüfungsordnung beachten
  - Arbeit muss selbstständig angefertigt werden
  - Der Übungsschein kann mehrfach wiederholt werden
  - Abschlussaufgaben können nach dem Übungsschein erworben werden
  - Die Abschlussaufgaben können nur einmal wiederholt werden
    - Muss bis zum Ende des dritten Semesters bestanden werden
    - Keine mündliche Nachprüfung

Vorlesung und Übung nachbereiten

- Folien der Vorlesung während des Semesters aktiv nacharbeiten
  - Im Semester nacharbeiten und nicht kurz vor den Abschlussaufgaben
  - Aufzeichnungen der Vorlesung sind auf YouTube verfügbar
  - Vorlesungsfolien sind im ILIAS vorhanden: s.kit.edu/ilias
  - Übungsblätter orientieren sich an der Struktur der Vorlesung

- Beispiellösungen der Übungsblätter nachvollziehen
  - Geben Hinweise zur Bearbeitung der Abschlussaufgaben
  - Nur Beispiele dafür, wie die Aufgabe gelöst werden könnte
  - Verstehen der verwendeten Konzepte in den Beispiellösungen
    - Entwurfsentscheidungen versuchen zu verstehen
    - Das Vorgehen nachvollziehen und verstehen

Unterstützung während des Semesters

Tutorien

- Möglichkeit zum Nachfragen
- Übungsblatt vor- und nacharbeiten
- Vorbereitung auf Präsenzübung
- s.kit.edu/programmieren

Info^2

- Info^2 und Lerngruppenbörse
- Mentoring und Kennenlernen Anderer
- Fachliche und persönliche Unterstützung
- Studienplanung Hilfsmittel

MINT

- Wiederholung des Vorlesungsinhalt
- Begrenzte Teilnehmerzahl
- Aufzeichnungen im ILIAS
- s.kit.edu/mint

Forum-Wi / (Aktive) Fachschaft

- Vielfältigen Angebot auf dem Campus
- Klausuren und Prüfungsprotokollen
- Fachliche und persönliche Unterstützung
- s.kit.edu/fsmi
- s.kit.edu/forum

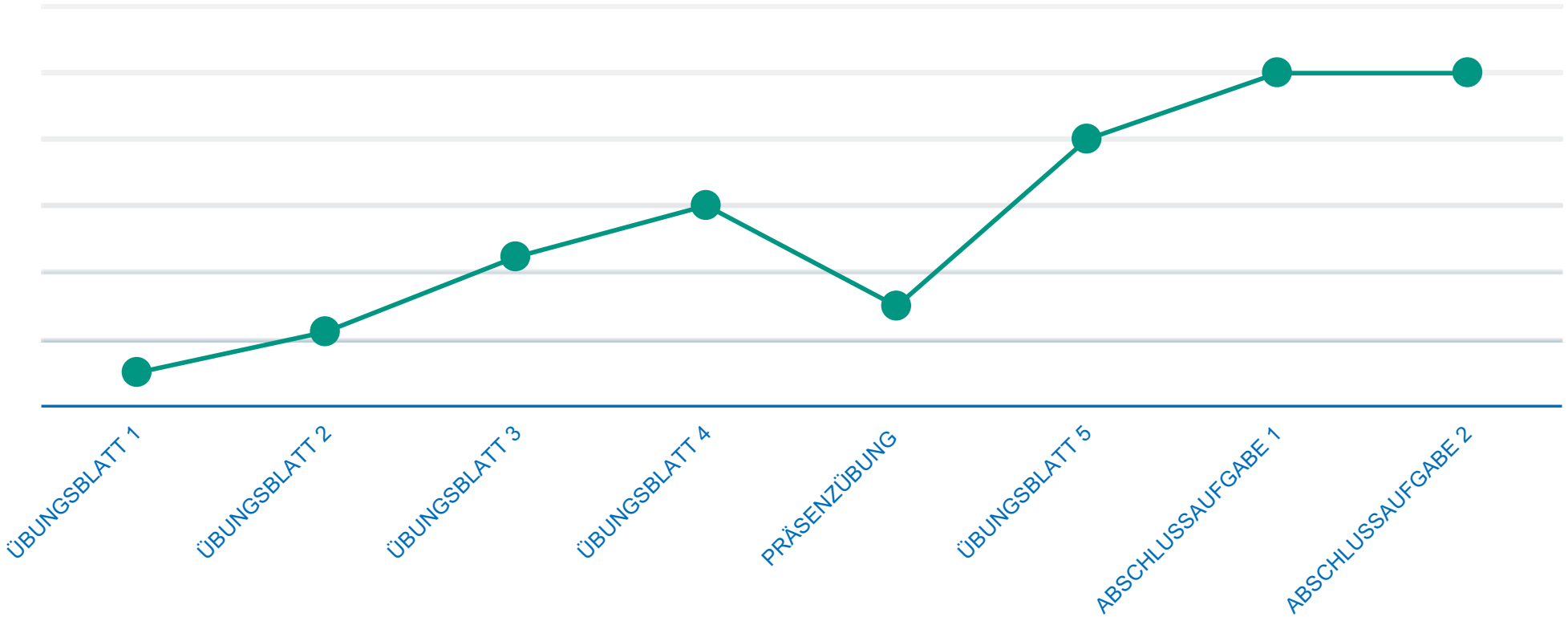
Programmieren Präsenzübung

- Prüft Wissen, welches bei der Bearbeitung der Übungsblätter erworben worden sein sollte
  - Kleine Quelltextausschnitte schreiben oder ergänzen
  - Verhalten von kleinen Quelltextausschnitten analysieren
  - Verbeißen Sie sich nicht zu lange in eine einzige Aufgabe

- Sie sind bereits gut vorbereitet, wenn Sie die Übungsaufgaben gewissenhaft und erfolgreich gelöst haben!
  - Kein besonderes Hintergrundwissen nötig
  - Keine Vorgaben zu Checkstyle-Regeln

- Zur Vorbereitung auf das „Programmieren auf Papier“
  - Frischen Sie die Java-Syntax auf
  - Nicht die Folien auswendig lernen

Subjektive Schwierigkeit



Übungsblätter und Abschlussaufgaben

Übungsschein nicht erworben?

- Vorlesung beginnt bei null
- Übungsblätter orientieren sich an der Vorlesung
- Sehr steile Lernkurve
  - Sehr anspruchsvoll, wenn keine Vorkenntnisse vorhanden sind
  - Fortgeschrittene Studenten sollten für die Methodik am Ball bleiben

Übungsschein bereits erworben?

- Übungsblätter als erneute Vorbereitung nutzen
- Das Abgabesystem anschauen
- Programme selbstständig implementieren
  - Aufgaben aus dem Internet
  - Algorithmen aus den Mathematikvorlesungen

Organisatorische Tipps für die Übungsblätter

- Übungsblätter als Vorbereitung nutzen
  - Durch das eigenständige Bearbeiten lernt man am meisten
  - Praktisch fast kein zusätzlicher Lernaufwand für die Präsenzübung
  - Das letzte Übungsblatt als Vorbereitung auf die Abschlussaufgaben nutzen
- Früh genug anfangen, die Übungsblätter zu bearbeiten
  - Unklarheiten können so frühzeitig beseitigt werden
  - Die Übungsblätter werden schnell deutlich anspruchsvoller
  - Auf den ersten Blättern so viele Punkte wie möglich sammeln
- Früh genug anfangen, die Übungsblätter abzugeben
  - Frühzeitiges erproben von Git und Artemis
  - Lösungen können beliebig häufig hochgeladen werden
  - Tutoren sieht nur die letzte gültige Abgabe

Praktische Tipps für die Aufgaben

- Quelltextklone vermeiden
  - Sich nicht Wiederholen und alles nur einmal im Quelltext implementieren
  - Aus doppelten Quelltextstellen versuchen, Methoden zu extrahieren
- Lange Methoden vermeiden
  - Methoden zerteilen und auf mehrere kleine (private) Hilfsmethoden verteilen
  - Dokumentation der Hilfsmethoden durch Javadoc-Kommentare
- Over-Engineering vermeiden
  - Quelltext so einfach wie möglich halten
  - Keine nicht benötigte Funktionalität
- Trennung der Anliegen
  - Datenstrukturen nicht nach außen geben
  - Trennung von Anwendungslogik und Benutzungsschnittstelle
- Geeignete Datentypen verwenden
  - Enums bei abgeschlossenen Mengen
  - Primitive Datentypen verwenden
  - Modellierung komplexer Typen mit Klassen
- Herangehensweise
  - Aufgabestellung genau lesen
  - Exakt die erwartete Ausgabe liefern
  - Bei Unklarheiten sofort nachfragen

Methodik Abzüge

- Neben Funktionalität wird auch Methodik geprüft
- Großer Teil der Bewertung von Übungsblättern und Abschlussaufgaben
- Alle Bewertungskriterien sind im Wiki: sdq.kastel.kit.edu/programmieren

Ilias-Forum

Materialsammlung

Vorlesungsmaterial
Foliensätze und Aufzeichnungen der Vorlesung.

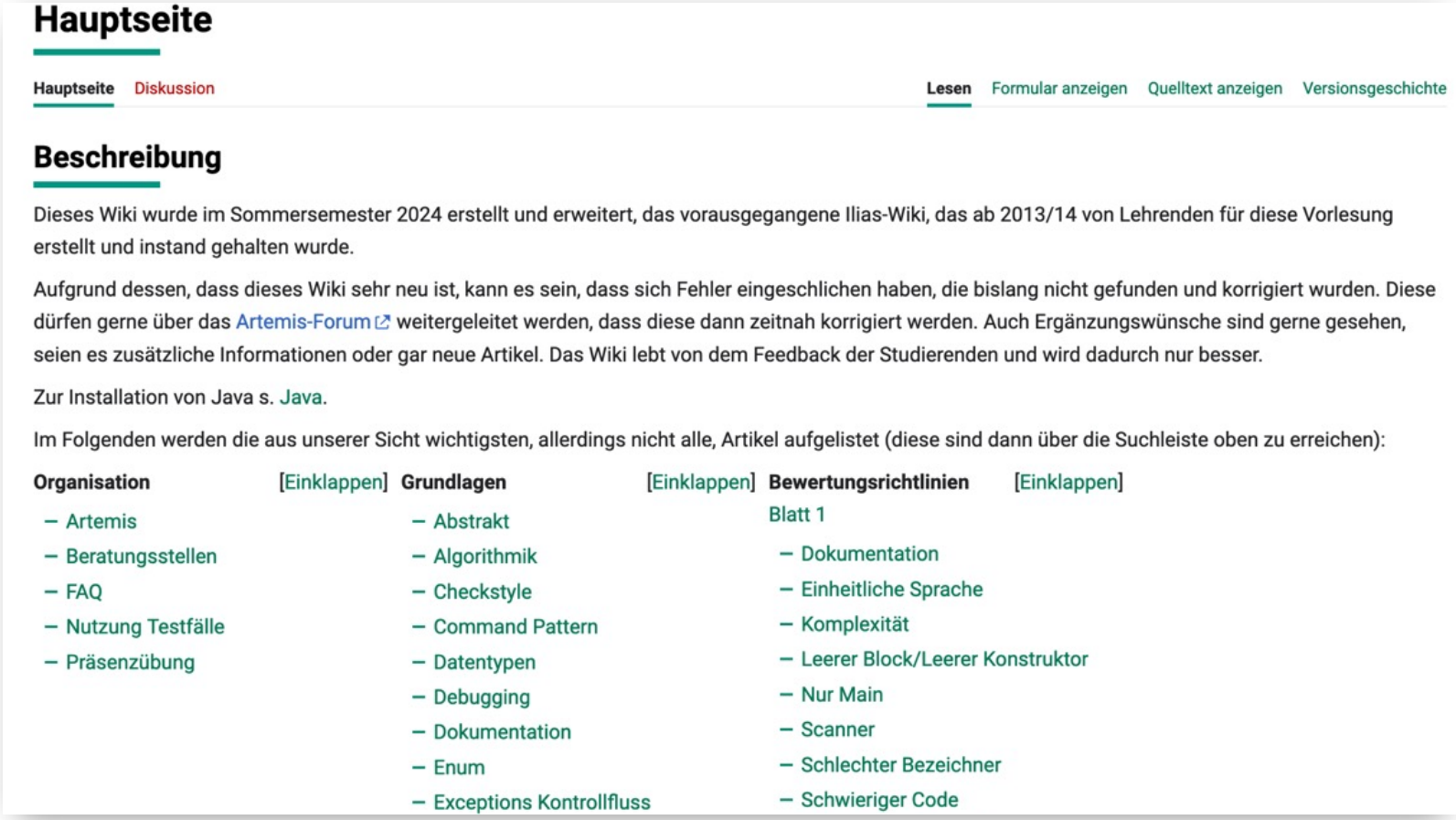
Tutoriumsmaterial
Inhalte aus den Tutorien.

Übungsblätter
Aufgabenstellungen, Abgabe und weitere Materialien zu den Übungsblättern.

Programmieren-Wiki
Erste Schritte, Methodik und Bewertungsrichtlinien.

Vorlesungsaufzeichnungen WiSe 24/25
Kursnummer: WS242424004 ; Ilias ID: 2476137 ; Semester: WS 24/25 ; CampusID

Ilias-Wiki: Übersicht



Hauptseite

Hauptseite | Diskussion | Lesen | Formular anzeigen | Quelltext anzeigen | Versionsgeschichte

Beschreibung

Dieses Wiki wurde im Sommersemester 2024 erstellt und erweitert, das vorausgegangene Ilias-Wiki, das ab 2013/14 von Lehrenden für diese Vorlesung erstellt und instand gehalten wurde.

Aufgrund dessen, dass dieses Wiki sehr neu ist, kann es sein, dass sich Fehler eingeschlichen haben, die bislang nicht gefunden und korrigiert wurden. Diese dürfen gerne über das Artemis-Forum weitergeleitet werden, dass diese dann zeitnah korrigiert werden. Auch Ergänzungswünsche sind gerne gesehen, seien es zusätzliche Informationen oder gar neue Artikel. Das Wiki lebt von dem Feedback der Studierenden und wird dadurch nur besser.

Zur Installation von Java s. Java.

Im Folgenden werden die aus unserer Sicht wichtigsten, allerdings nicht alle, Artikel aufgelistet (diese sind dann über die Suchleiste oben zu erreichen):

Organisation [Einklappen]

- Artemis
- Beratungsstellen
- FAQ
- Nutzung Testfälle
- Präsenzübung

Grundlagen [Einklappen]

- Abstrakt
- Algorithmik
- Checkstyle
- Command Pattern
- Datentypen
- Debugging
- Dokumentation
- Enum
- Exceptions Kontrollfluss

Bewertungsrichtlinien [Einklappen]

Blatt 1

- Dokumentation
- Einheitliche Sprache
- Komplexität
- Leerer Block/Leerer Konstruktor
- Nur Main
- Scanner
- Schlechter Bezeichner
- Schwieriger Code

Ilias-Wiki: Bewertungsrichtlinien

Bewertungsrichtlinien [Einklappen]

Blatt 1

- Dokumentation
- Einheitliche Sprache
- Komplexität
- Leerer Block/Leerer Konstruktor
- Nur Main
- Scanner
- Schlechter Bezeichner
- Schwieriger Code
- Unbenutztes Element
- Verwendung von instanceOf

Blatt 2

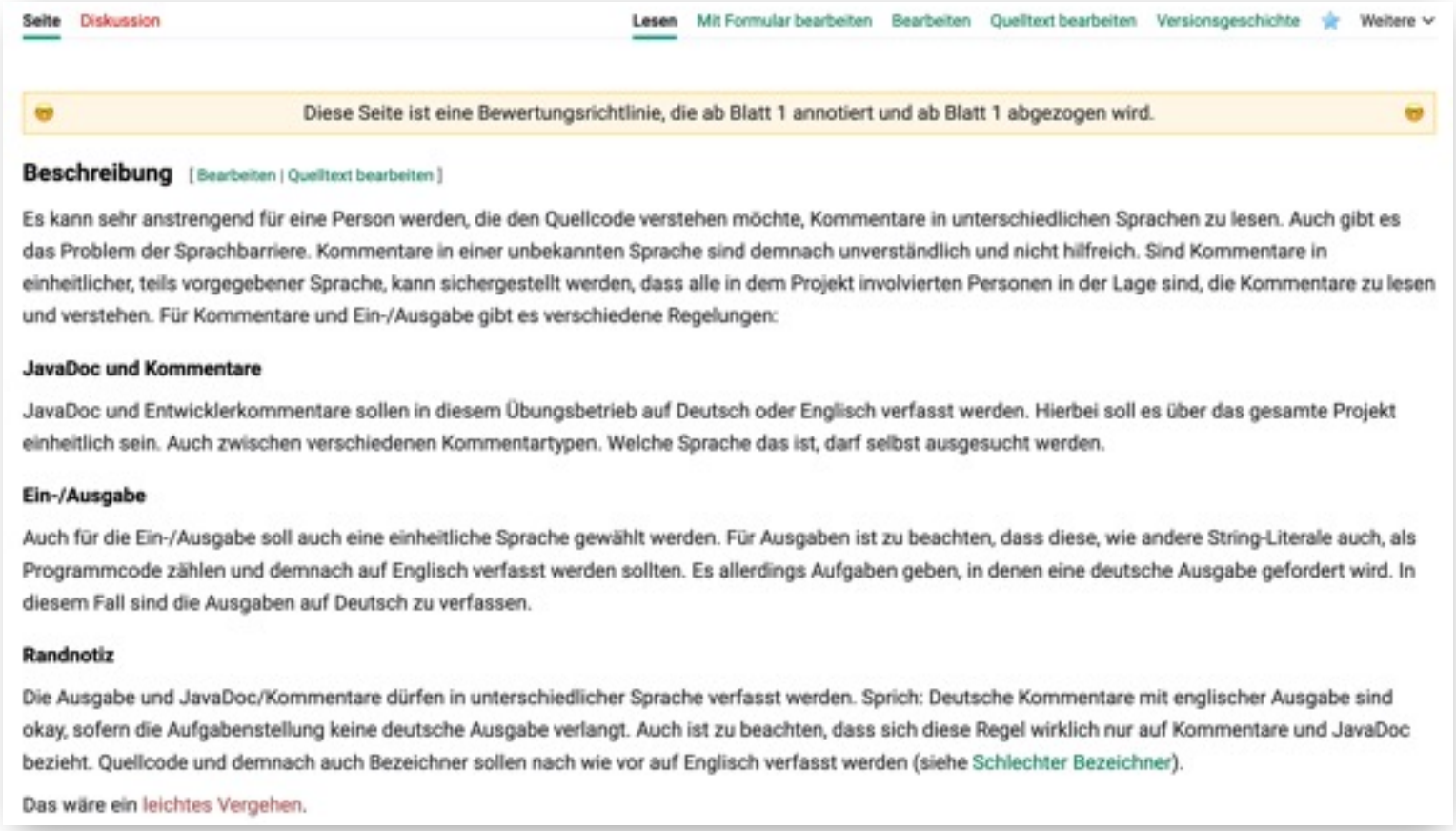
- Assertions
- Bedeutungslose Konstanten
- Enum
- Fehlermeldungen
- Final
- Getter / Setter für Listen
- Hartcodieren
- Hilfsklasse
- IO/UI
- JavaDoc
- JavaDoc Trivial
- Konstanten-Klasse
- Magic Number
- Object statt konkreter Klasse
- Pakete
- Polymorphie
- Raw Types
- Reimplementierung
- Runtime Exceptions
- Sichtbarkeit
- Stringreferenzen
- Ungeeigneter Schleifentyp
- Zeilenumbrüche

Blatt 3

- Duplikate
- Exceptions Kontrollfluss
- Große Try-Catch Blöcke
- Interface statt konkreter Klasse
- Statische Methoden und Attribute
- Verweigerte Vererbung

} Alle gelten für die Abschlussaufgaben!

Ilias-Wiki: Einheitliche Sprache



Seite Diskussion Lesen Mit Formular bearbeiten Bearbeiten Quelltext bearbeiten Versionsgeschichte Weitere

Diese Seite ist eine Bewertungsrichtlinie, die ab Blatt 1 annotiert und ab Blatt 1 abgezogen wird.

Beschreibung [Bearbeiten | Quelltext bearbeiten]

Es kann sehr anstrengend für eine Person werden, die den Quellcode verstehen möchte, Kommentare in unterschiedlichen Sprachen zu lesen. Auch gibt es das Problem der Sprachbarriere. Kommentare in einer unbekannten Sprache sind demnach unverständlich und nicht hilfreich. Sind Kommentare in einheitlicher, teils vorgegebener Sprache, kann sichergestellt werden, dass alle in dem Projekt involvierten Personen in der Lage sind, die Kommentare zu lesen und verstehen. Für Kommentare und Ein-/Ausgabe gibt es verschiedene Regelungen:

JavaDoc und Kommentare

JavaDoc und Entwicklerkommentare sollen in diesem Übungsbetrieb auf Deutsch oder Englisch verfasst werden. Hierbei soll es über das gesamte Projekt einheitlich sein. Auch zwischen verschiedenen Kommentartypen. Welche Sprache das ist, darf selbst ausgesucht werden.

Ein-/Ausgabe

Auch für die Ein-/Ausgabe soll auch eine einheitliche Sprache gewählt werden. Für Ausgaben ist zu beachten, dass diese, wie andere String-Literale auch, als Programmcode zählen und demnach auf Englisch verfasst werden sollten. Es allerdings Aufgaben geben, in denen eine deutsche Ausgabe gefordert wird. In diesem Fall sind die Ausgaben auf Deutsch zu verfassen.

Randnotiz

Die Ausgabe und JavaDoc/Kommentare dürfen in unterschiedlicher Sprache verfasst werden. Sprich: Deutsche Kommentare mit englischer Ausgabe sind okay, sofern die Aufgabenstellung keine deutsche Ausgabe verlangt. Auch ist zu beachten, dass sich diese Regel wirklich nur auf Kommentare und JavaDoc bezieht. Quellcode und demnach auch Bezeichner sollen nach wie vor auf Englisch verfasst werden (siehe Schlechter Bezeichner).

Das wäre ein leichtes Vergehen.

Ilias-Wiki: Einheitliche Sprache

1 Negativbeispiel

```java
/**
* Prints the status of the player to the console.
* @param player The id of the player
*/
public void printPlayerStatus(int playerId) {
    // Falls der Spieler unbekannt ist wird null zurückgegeben
    Player player = findPlayerById(playerId);

    if (player == null) {
        System.err.println("Error, unbekannter Spieler");
    } else {
        System.out.println("Player " + playerId + ": " + player.getHP() + "HP");
    }
}
```

Ilias-Wiki: Einheitliche Sprache

2 Positivbeispiel

```java
/**
 * Prints the status of the player to the console.
 * @param player The id of the player
 */
public void printPlayerStatus(int playerId) {
    // Returns null if the player is unknown
    Player player = findPlayerById(playerId);

    if (player == null) {
        System.err.println("Error, unknown player");
    } else {
        System.out.println("Player " + playerId + ": " + player.getHP() + "HP");
    }
}
```

Schlechte Bezeichner

- Bezeichner sollten *sprechend* sein
- Keine Abkürzungen, konform zu Konventionen
- Ausnahmen: i, j, x, y, it

```java
public static int s(final int a, final int b) {
    return a - b;
}
```

```java
public static int subtract(final int minuend, final int subtrahend) {
    return minuend - subtrahend;
}
```

Final

- Attribute sollten immer, wenn möglich, final sein
- Verhindert unabsichtliche Änderungen der Attribute (nicht ihrer Zustände)

```
class Event {
    private static int BASE_FEE = 1000;
    private List<String> participants;

    public Event() {
        participants = new ArrayList<>();
    }

    public void addParticipant(String name) {
        participants.add(name);
    }

    public int calculateProfit(int ticketPrice, int venueCost) {
        int fixedCost = BASE_FEE + venueCost;
        return participants.size() * ticketPrice - fixedCost;
    }
}
```

Final

- Attribute sollten immer, wenn möglich, final sein
- Verhindert unabsichtliche Änderungen der Attribute (nicht ihrer Zustände)

```java
class Event {
    private final static int BASE_FEE = 1000; // constants should be final
    private final List<String> participants; // fields should be final if possible

    public Event() {
        participants = new ArrayList<>();
    }

    public void addParticipant(String name) { // not required for parameters or local variables
        participants.add(name); // (list is final, but not the content)
    }

    public int calculateProfit(int ticketPrice, int venueCost) {
        int fixedCost = BASE_FEE + venueCost;
        return participants.size() * ticketPrice - fixedCost;
    }
}
```

Ungeeigneter Schleifentyp

- Boolscher-Ausdruck => While-Schleife, Do-While-Schleife
- Einfache Iteration => For-Each-Schleife
- Index benötigt => For-Schleife

```java
public void printList(List<String> list){
    for(int i = 0; i<list.size(); i++){
        System.out.println(list.get(i));
    }
}
```

```java
public void printList(List<String> list){
    for(String item : list){
        System.out.println(item);
    }
}
```

Unnötige Komplexität

- Redundanz/Komplexität erhöht visuelles Rauschen
  => einfachste Lösung wählen

```
1  boolean isValid() {
2      if (this.sold == false && !(this.price <= 0)) {
3          return true;
4      } else {
5          return false;
6      }
7  }
```

```
1  boolean isValid() {
2      return !this.sold && this.price > 0;
3  }
```

Magic Numbers/Strings

- Alleinstehende Zahlen im Quelltext sind oft schwer zu verstehen
- Variablen erlauben Benennung
- Konstanten erlauben Wiederverwendung

```
1  final String password = "Test";
2  if (password.length() < 8) {
3      System.err.println("Password too short, please choose a longer password!");
4  }
```

```
1  final String password = "Test";
2  final int minimalPasswordLength = 8;
3  if (password.length() < minimalPasswordLength) {
4      System.err.println(PASSWORD_ERROR_MESSAGE);
5  }
```

Sichtbarkeit und Datenkapselung

- Die Sichtbarkeit sollte so restriktiv wie möglich gesetzt werden
- Interner Zustand soll nur, falls wirklich nötig, herausgegeben werden
- Die Möglichkeit zu ungeplanten Zugriff führt zu Bugs

```
public final class Book {
    public String title;
    long isbn;

    protected Book(String title, long isbn) { // protected sinnlos, da Klasse final
        if (!isIsbnValid(isbn)) {
            throw new InvalidIsbnException("The isbn is not valid");
        }
        this.title = title;
        this.isbn = isbn;
    }

    public boolean isIsbnValid(long isbn) { // interne Hilfsmethode
        // ...
    }
}
```

Sichtbarkeit und Datenkapselung

- Die Sichtbarkeit sollte so restriktiv wie möglich gesetzt werden
- Interner Zustand soll nur, falls wirklich nötig, herausgegeben werden
- Die Möglichkeit zu ungeplanten Zugriff führt zu Bugs

```java
public final class Book {
    private String title;
    private long isbn;

    public Book(String title, long isbn) {
        if (!isIsbnValid(isbn)) {
            throw new InvalidIsbnException("The isbn is not valid");
        }
        this.title = title;
        this.isbn = isbn;
    }

    public String getTitle() { return this.title; }
    public long getIsbn() { return this.isbn;}
    private boolean isIsbnValid(long isbn) {
        // ...
    }
}
```

Testen

- Testen hilft, Fehler zu finden!
  - Vergleich von tatsächlichem und erwünschtem Verhalten
  - Modularer Quelltext hilft beim Testen
- Ausführliches Testen vordem Hochladen
  - Nicht auf den Public-Test verlassen
  - Den Public-Test lokal ausführen
- Selber Tests schreiben
  - Für die Kommandozeileninteraktion
  - Über eigene Main-Methode
  - JUnit-Tests schreiben

Teststrategien

- Datenbasiert
  - Beispiele auf Aufgabenblättern
- Kontrollflussbasiert
  - Suche in Datenstrukturen
  - Alle Entscheidungen einmal treffen
- Grenzwertbasiert
  - Wertebereich bei Berechnungen
  - Off-by-one-Error

Rückfragen

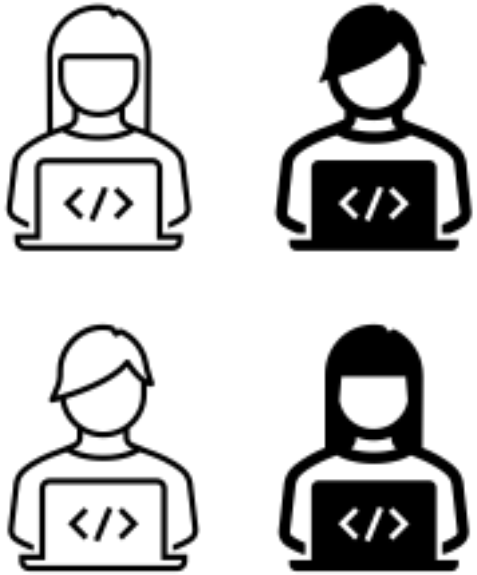
- Fragen und Unklarheiten können immer auftreten
  - Wenn ihr eine Frage habt, haben andere vielleicht die gleiche Frage
  - Wir bieten viele Möglichkeiten, Fragen zu stellen
- Bei Rückfragen werden Aufgaben gegebenenfalls aktualisiert
  - Lediglich minimale Aktualisierungen oder weitere Klarstellungen
  - Schaut immer mal wieder, ob die aktuelle Aufgabe aktualisiert wurde

Reihenfolge bei Fragen

1. Websuche
2. FAQ / Programmieren-Wiki prüfen
   - *s.kit.edu/faq*
   - *sdq.kastel.kit.edu/programmieren*
3. Diskussionsforum prüfen
   - Suchfunktion verwenden
4. Im Diskussionsforum nachfragen
   - Aussagekräftigen Titel verwenden
5. Eure Tutoren fragen
6. Übungsleitung fragen
   - *programmieren-vorlesung@cs.kit.edu*

Zusammenfassung

- Nehmen Sie sich **ausreichend Zeit** für die Übungsblätter
- Fangen Sie **rechtzeitig** mit der Bearbeitung an
- Beachten Sie die **Bewertungskriterien** im Ilias-Wiki
- Verwenden Sie früh **Checkstyle**
- Besuchen Sie die **Tutorien** und nehmen Sie aktiv Teil
- Üben Sie für die Präsenzübung **auf Papier**



Überblick über das zweite Semester

Wie geht das Studium weiter?

Module im 2. Semester: Informatik

- LA2
- HM2
- Softwaretechnik 1
- Digitaltechnik und Entwurfsverfahren
- Algorithmen 1

- Die Module sind eine Empfehlung vom Modulhandbuch, aber keine Pflicht!

Info²: “Programmieren Best Practices " + Lernpartnerschaftsbörse" (**Sommersemester 2025**)
https://www.informatik.kit.edu/imstudium.php

Module im 2. Semester: Winfo

- Wirtschaftsinformatik 2
- Algorithmen 1
- Mathe 2
- Statistik 1
- Einführung in das Operations Research (1)
- Recht (BGB für Fortgeschrittene)

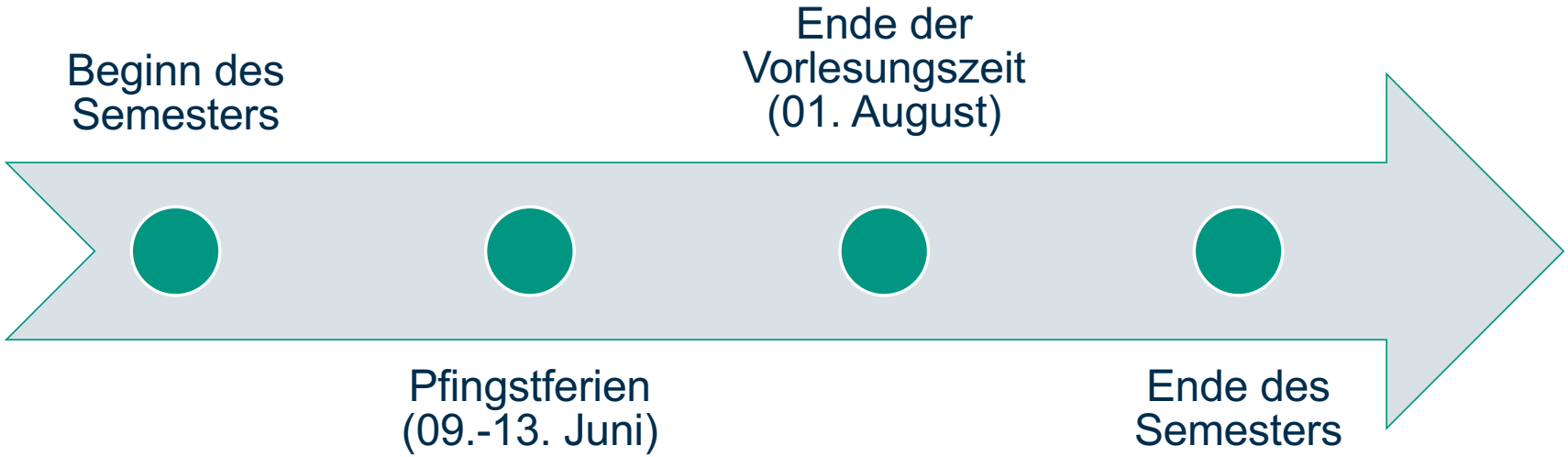
Info²: “Programmieren Best Practices " + Lernpartnerschaftsbörse" (**Sommersemester 2025**)
https://www.informatik.kit.edu/imstudium.php

Übersicht über die Übungsscheine

Modul	Übungsschein notwendig für Prüfung?	Semester
Programmieren	Ja	WS + SS
Softwaretechnik	Nein, aber um Modul zu bestehen	SS
Lineare Algebra II (für Info)	Ja	SS
Lineare Algebra II (Mathe)	Ja	SS
Höhere Mathematik II	Ja	SS
Analysis II	Ja	SS
Mathe II	Nein, aber um Modul zu bestehen	SS
Wirtschaftsinformatik II	Nein, nur Klausurbonus	SS

Info²: "Programmieren Best Practices " + Lernpartnerschaftsbörse" (**Sommersemester 2025**)
https://www.informatik.kit.edu/imstudium.php

Zeit zu reflektieren



- Gut Reflektiert ist halb Gewonnen
  - „Fühlt sich das Studium für mich weiterhin richtig an?“
  - „Welche Fehler habe ich im ersten Semester gemacht und mache ich diese gerade wieder?“
- Ferien: Nimm dir Zeit für dich!

- Wieder zur Uni
  - Stundenplan überdenken
  - Stoff wieder einarbeiten
- Prüfungsvorbereitung beginnen

Info²: “Programmieren Best Practices " + Lernpartnerschaftsbörse" (**Sommersemester 2025**)
https://www.informatik.kit.edu/imstudium.php

Ressourcen zur Reflexion

- Zentrale Studienberatung (ZSB)
- Angebote der Fakultät für Informatik
  - Informatik Studiengangservice (ISS)
  - eezi-Beratungs Gepräch
  - Fachschaft (FSMI) / Wi-Forum
- Mit Studierenden oder Tutoren aus höheren Fachsemestern reden
- Mit sozialem Umkreis und Familie reden

Info²: "Programmieren Best Practices " + Lernpartnerschaftsbörse" (**Sommersemester 2025**)
https://www.informatik.kit.edu/imstudium.php

Prüfungstermine in VL-FZ

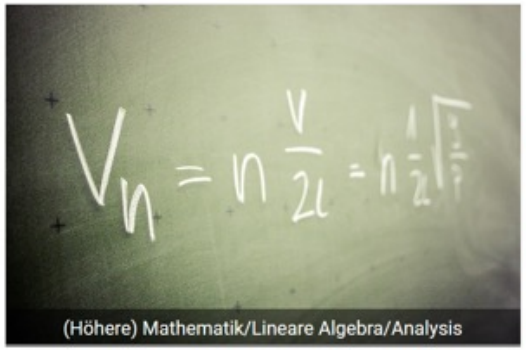
Programmieren (NT)	AA1: 13.08. - 11.09.2025 AA2: 27.08. - 25.09.2025	jeweils 13:00 - 06:00
Statistik 1	02.08.2025	08:00 - 10:30
Softwaretechnik 1	04.08.2025	08:00 - 10:00
Wirtschaftsinformatik 2	15.08.2025	14:00 - 15:00
GBI (NT)	19.08.2025	08:00 - 10:00
Algorithmen I	25.08.2025	08:00 - 10:00
Wirtschaftsinformatik 1 (NT)	29.08.2025	08:00 - 09:00
Analysis 2	02.09.2025	08:00 - 10:00
Analysis 1 (NT)	03.09.2025	14:00 - 16:00
Höhere Mathematik I & II	03.09.2025	08:00 - 12:00
LA 1 (NT) & 2	16.09.2025	LA1: 12:00 - 14:00 LA2: 15:00 - 17:00
Mathematik 1 (NT)	25.09.2025	08:00 - 09:30
Mathematik 2	29.09.2025	11:00 - 12:30

MINT-Kolleg

MINT-Kolleg: Vorkurse, Semesterkurse, Helpdesk, Aufbaukurse

Bietet Semesterkurse an zu

- Höhere Mathematik I
- Lineare Algebra I
- Programmieren



→ Im Ilias-Kurs können Ressourcen genutzt werden, auch wenn man keinen Kursplatz hat

Info²: "Programmieren Best Practices " + Lernpartnerschaftsbörse" (**Sommersemester 2025**)
https://www.informatik.kit.edu/imstudium.php

Ausgleichsmöglichkeiten

- Hochschulsport

- KIT ***Sprachenzentrum*** :Sprachkurse
  - 1 Kostenfreikurs pro Semester

- Hochschulgruppen

Info²: "Programmieren Best Practices " + Lernpartnerschaftsbörse" (**Sommersemester 2025**)
https://www.informatik.kit.edu/imstudium.php

Weitere Informationen

- Programmieren Übung SS25
- MINT-Kolleg
- Modulhandbücher
- Informatik Studiengangsservice
- Unterstützung im Studium – Website
- Info2 - Youtube Playlist
- while (true) - Youtube Podcasts eines Alumni

Und jetzt:

Zeit für eure Fragen!

- Unten „Unterstützung im Studium“ Youtube Play List

  sind Videos auf Deutsch und Englisch. In der Playlist finden Sie dieses Thema sowie weitere für Ihr Studium relevante Themen