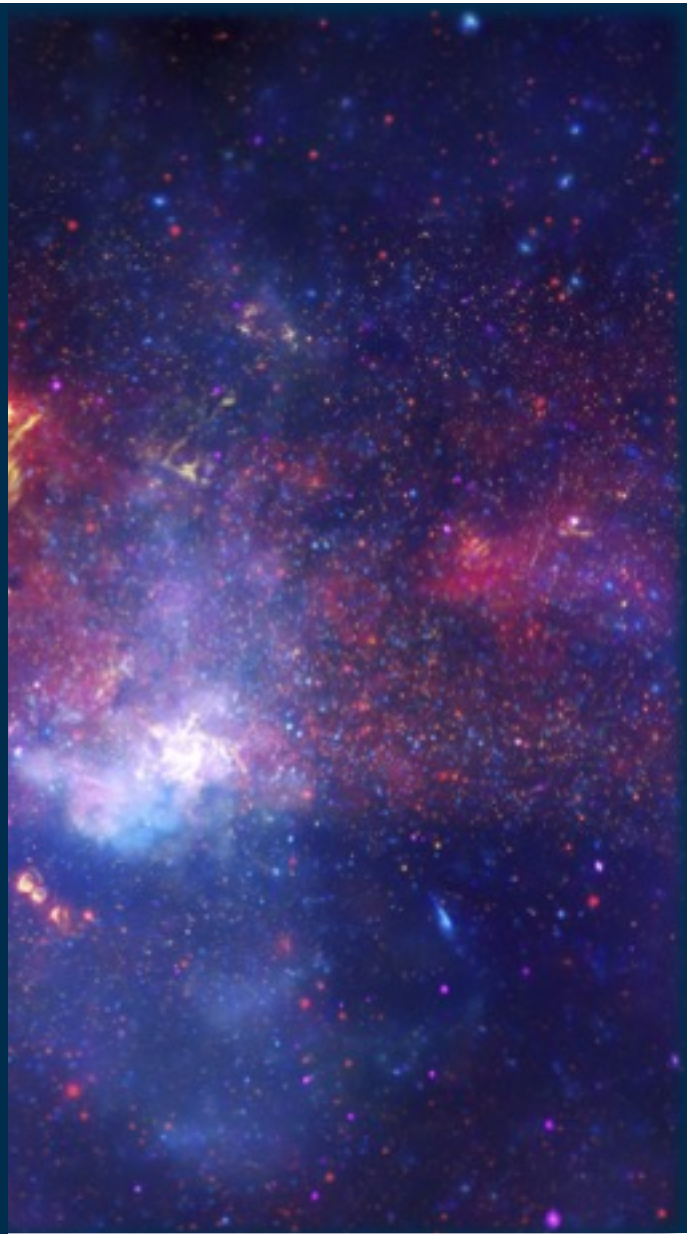
eezi

Eine Einführung zum Informatikstudium

Strategien für Übungsblätter der Orientierungsprüfungen & Programmieren Best Practices



2. Vorlesung – WS 2025-26

25.11.2025

Inhalte

1. Überblick zur Vorlesung
2. Programmieren Best Practices
3. Tipps zu den Übungsblättern
4. Allgemeine Tipps
5. Reflexion
6. Fragen und Antworten
7. Lernpartnerschaftsbörse

Es ist, was es ist,
aber es wird, was du daraus machst

Karlsruher Institut für Technologie

Software Design and Quality

Best Practices für „Programmieren“

Info^2 / eezi

Das Programmieren-Team

Nils Niehues

Haoyu, Liu

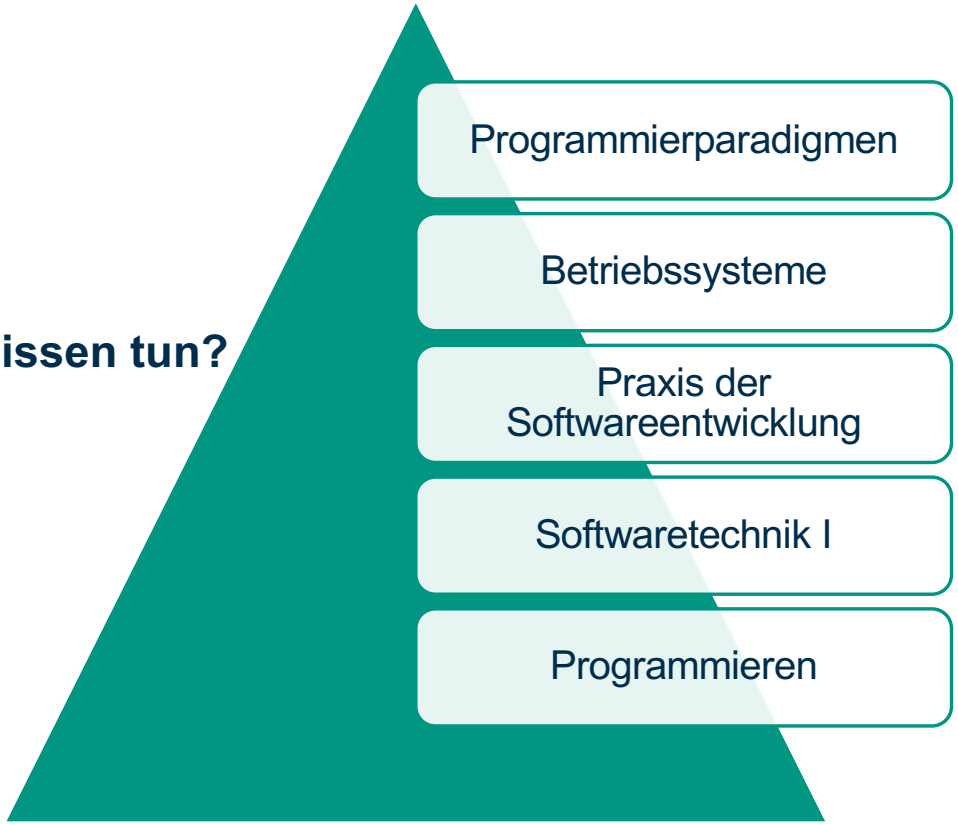
Robin Maisch

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

Unterstützung im Studium

- Info^2, eezi und Lernpartnerschaftsbörse
- 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

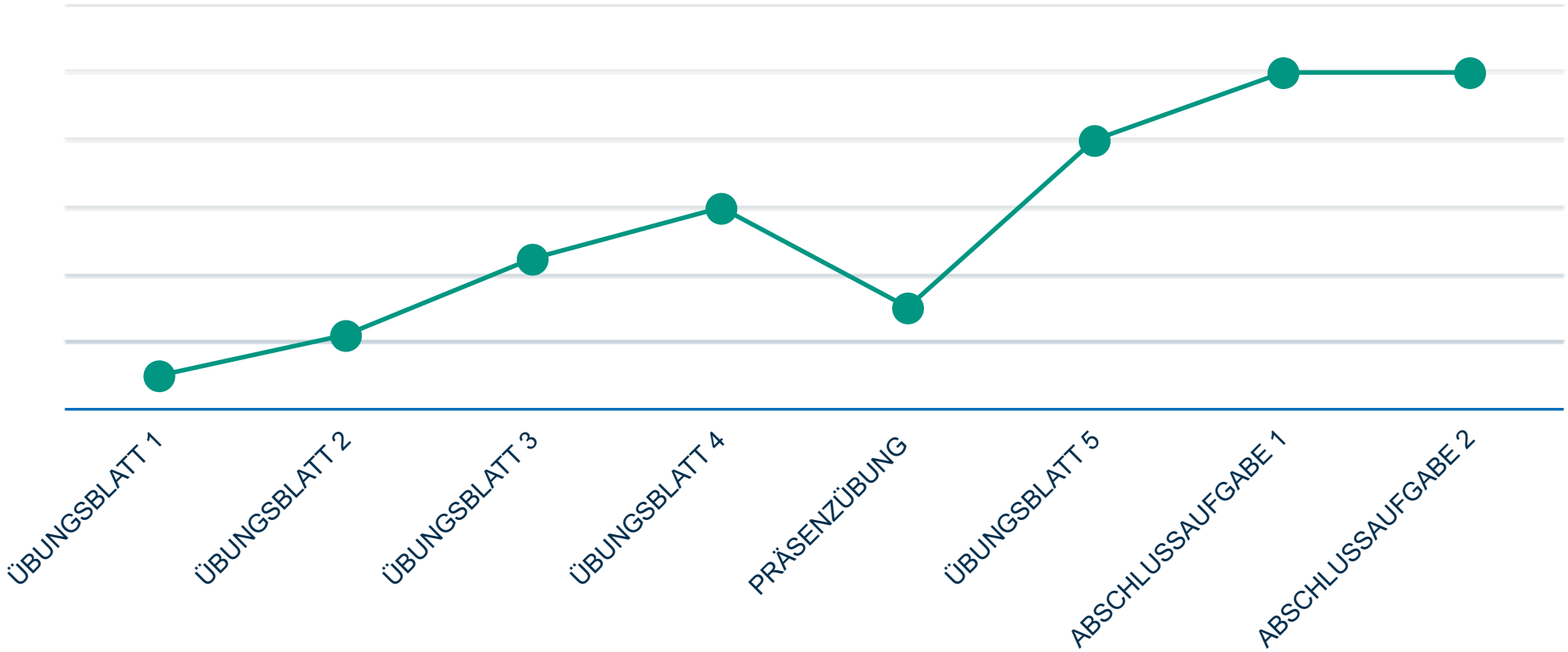
Programmieren Präsenzübung

- **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
  - Tutor 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
- Geeig**nete 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.

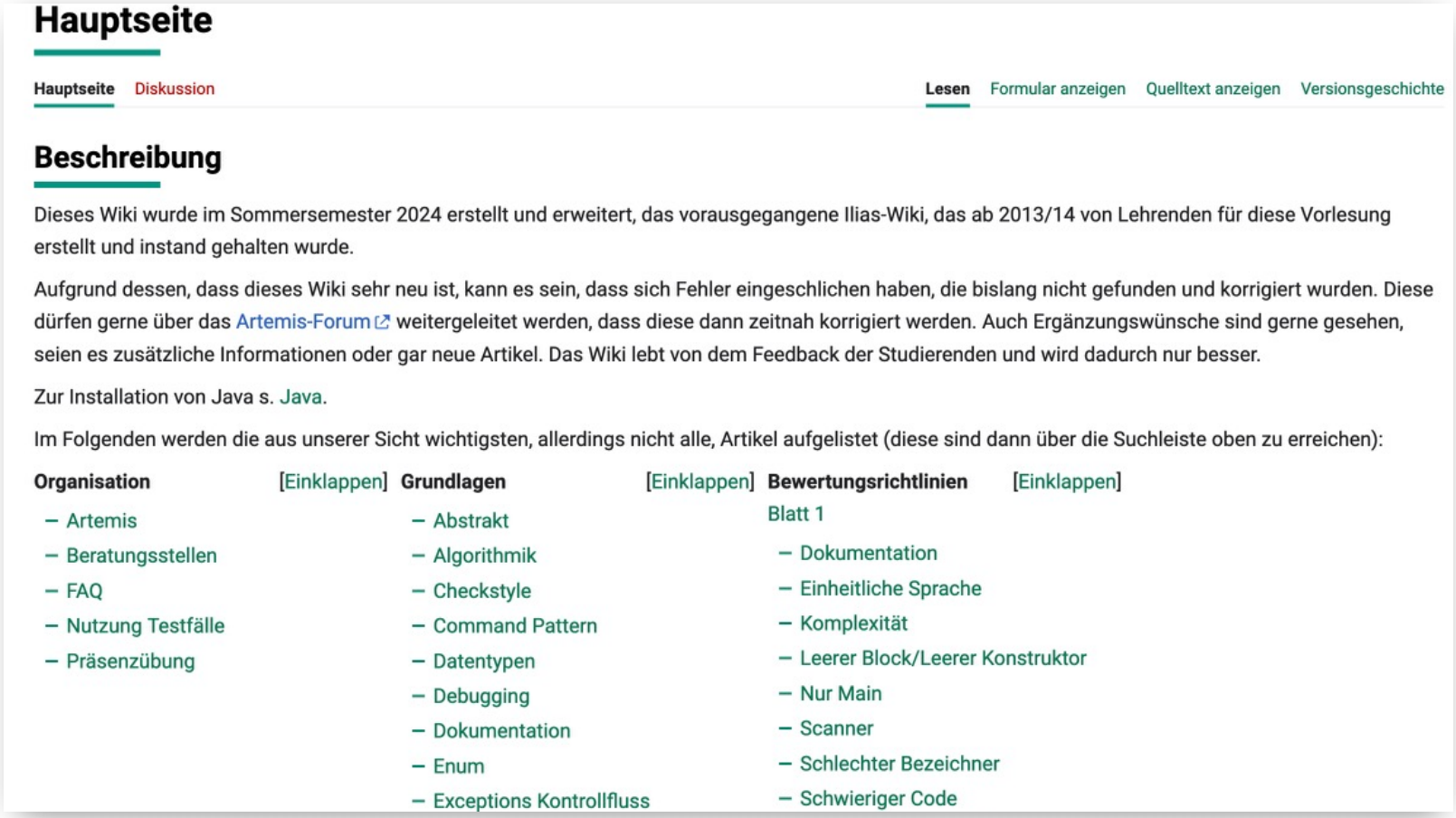
Tutoriumsmaterial
Inhalte aus den Tutorien.

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

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

Wiki: Bewertungsrichtlinien

Bewertungsrichtlinien [Einklappen]

Blatt 1 Abzüge

- Einheitliche Sprache
- Systemabhängiger Zeilenumbruch

Blatt 2 Abzüge

- Dokumentation
- Instanceof außerhalb der equals-Methode
- Komplexität
- Leerer Block/Leerer Konstruktor
- Nur Main
- Schlechter Bezeichner
- Schwieriger Code
- Unbenutztes Element

Blatt 3 Abzüge

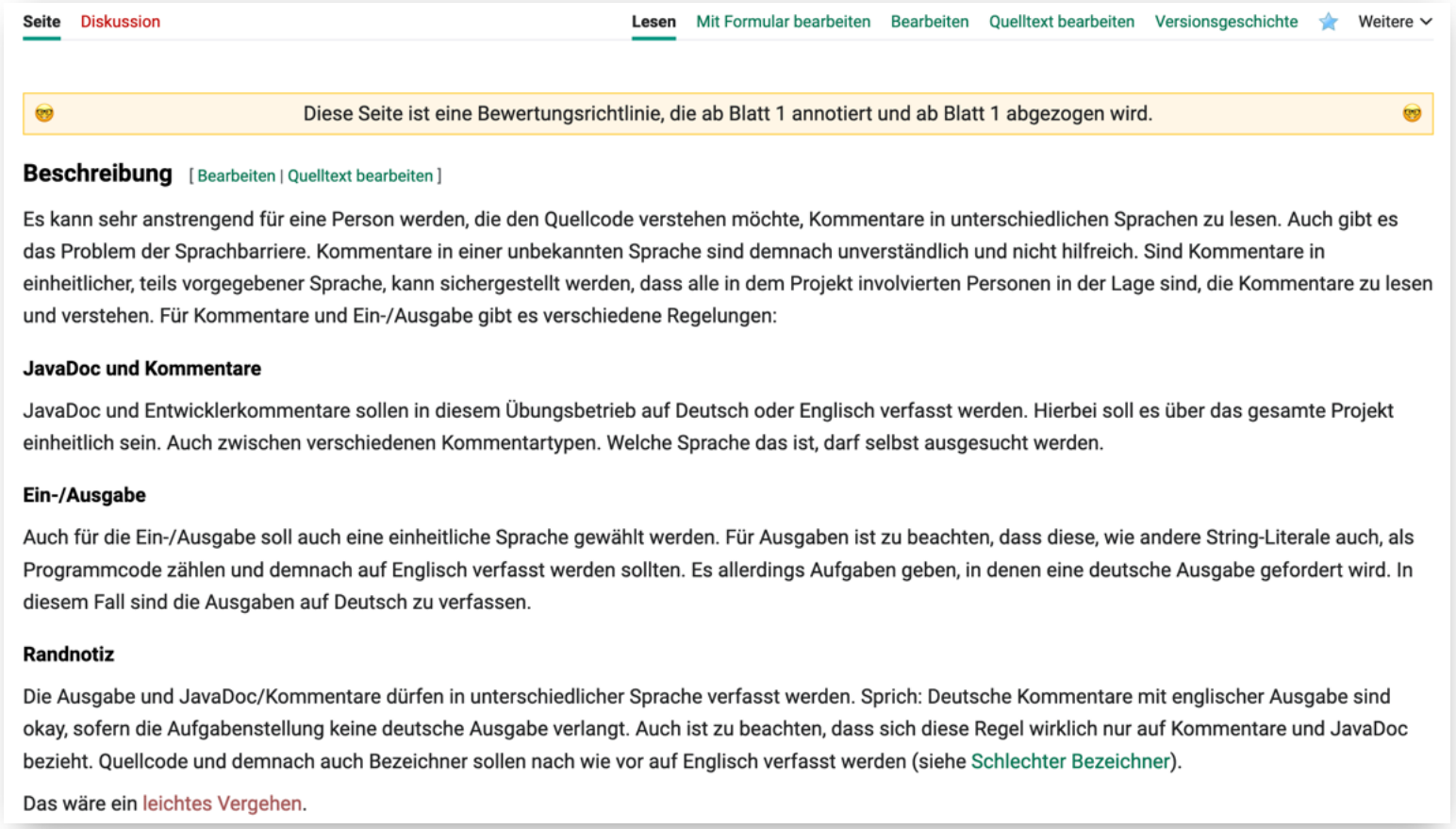
- Bedeutungslose Konstanten
- Datenkapselung
- Duplikate
- Enum
- Fehlermeldungen
- Final
- Hilfsklasse
- JavaDoc
- Konstanten-Klasse
- Magic Literal
- Object statt konkreter Klasse
- Pakete
- Polymorphie
- Raw Types
- Reimplementierung
- Runtime Exceptions
- Sichtbarkeit
- Stringreferenzen
- Ungeeigneter Schleifentyp

Blatt 4 Abzüge

- Cast außerhalb der equals-Methode
- Exceptions Kontrollfluss
- Große Try-Catch Blöcke
- Interface statt konkreter Klasse
- Statische Methoden und Attribute
- Verweigerte Vererbung

Alle gelten für die Abschlussaufgaben!

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.

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");
    }
}
```

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

```java
boolean isValid() {
    if (this.sold == false && !(this.price <= 0)) {
        return true;
    } else {
        return false;
    }
}
```

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

Magic Numbers/Strings

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

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

```java
final String password = "Test";
final int minimalPasswordLength = 8;
if (password.length() < minimalPasswordLength) {
    System.err.println(PASSWORD_ERROR_MESSAGE);
}
```

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

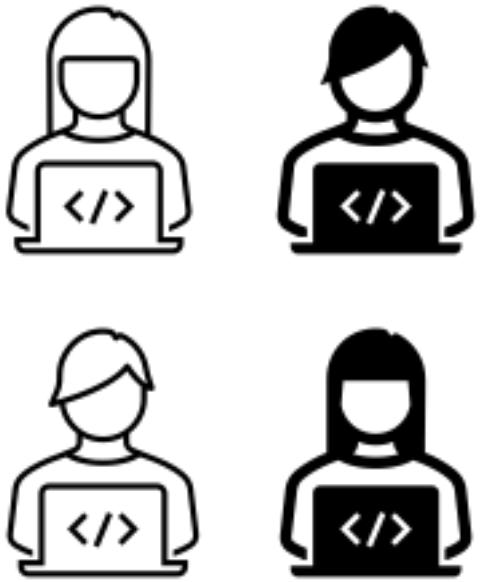
- **Fragen und Unklarheiten können** immer auftreten
  - Wenn ihr eine Frage habt, haben andere vielleicht die gleiche Frage
  - Wir bieten viele Möglichkeiten, Fragen zu stellen
- **Bei Rückfragen werden Aufgaben gegebenenfalls aktualisiert**
  - Lediglich minimale Aktualisierungen oder weitere Klarstellungen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



Tipps zu den Übungsblättern

3

Übungsscheine - Übersicht

Modul	Übungsschein notwendig für Prüfung	Semester
Grundbegriffe der Informatik	Nein, aber um Modul zu bestehen	WS
Programmieren	Ja	WS + SS
Lineare Algebra I (für Info)	Ja	WS + SS: LA II Übungsschein
Lineare Algebra I (Mathe)	Ja	WS
Höhere Mathematik I	Ja	WS + SS: HM II Übungsschein
Analysis I	Ja	WS
Mathe I	Nein, aber um Modul zu bestehen	WS
Wirtschaftsinformatik I	Nein, nur Klausurbonus	WS

Grundbegriffe der Informatik

GBI: Übungsschein

- **Zum Bestehen notwendig:**
  - Mindestens 50% der Punkte
  - Mindestens 20% der Punkte auf jedes Aufgabenblatt (bis auf 2)

- **Nur im WS machbar**
  - Erst im 3. Semester nachholbar!

- **Übungsschein ist notwendig, um das Modul zu bestehen**

- **Prüfung lässt sich aber auch ohne Übungsschein schreiben**

GBI:
Tipps

- **Tutorium bereitet auf neues Übungsblatt vor**

- **Definitionen bereithalten (z.B. aus Vorlesung / Übung)**

- **Nach ähnlichen Aufgaben und Lösungen suchen**
  - In der Übung
  - In Übungsblättern der letzten Jahre
  - In Altklausuren
  - Illias, (Fachschaft)

GBI:
Hilfreiche Seiten

- **Youtube**
  - NLogSpace
  - Morpheus Tutorials

Mathematik

Lineare Algebra I: Übungsschein

- **Zum Bestehen notwendig:**
  - Mindestens 40% der Punkte jeweils in Übungsblättern 1-7 und 8-14
- **Wird nur im WS angeboten**
  - Alternative im Sommersemester Lineare Algebra II Übungsschein
  - MathematikerInnen brauchen beide Scheine
- **Für das Orientierungsmodul notwendig**
- **Für die Prüfung notwendig**

Höhere Mathematik I: Übungsschein

- **Zum Bestehen notwendig:**
  - Mindestens 50% der Punkte jeweils in Übungsblättern 1-7 und 8-13
- **Wird nur im WS angeboten**
  - Alternative: Höhere Mathematik II Übungsschein im Sommersemester
- **Für die Prüfung notwendig**

Analysis I: Übungsschein

- **Zum Bestehen notwendig:**
  - Mindestens 40% der Punkte jeweils in Übungsblättern 1-7 und 8-14

- **Wird nur im WS angeboten**
  - Alternative: Teilnahme an Scheinklausur, diese findet jedoch parallel zur Hauptklausur Analysis I statt

- **Für die Prüfung notwendig**

Äquivalenz von Übungsscheinen

- **Die Übungsscheine von LA1 für Informatiker und LA1 sind gleichwertig und ein Wechsel ist möglich**
- **Die Übungsscheine von Analysis 1 und Höhere Mathematik 1 sind gleichwertig und ein Wechsel ist möglich**
- **Genauere Informationen findet man im Modulhandbuch oder im ISS FAQ**

Mathe I:
Übungsschein

- **Zum Bestehen notwendig:**
  - Mindestens 50% der Punkte jeweils in Übungsblättern 1-7 und 8-15
- **Wird nur im WS angeboten**
  - am besten jetzt versuchen, da O-Prüfung
- **Zum Bestehen des Moduls notwendig**
- **Teilnahme an Prüfung auch ohne Übungsschein möglich**

Mathematik: Tipps

- **Sich nicht daran aufhalten, es sich vorstellen zu wollen**
  - Mathematik an Uni ist sehr abstrakt
  - Versuchen es einfach mal nur stur anzuwenden

- **Ähnliche Aufgaben mit Lösungen Googlen, erfragen, …**
  - Lösungen aber nicht abschreiben, sondern nachvollziehen und auf Aufgabenstellung übertragen

Mathematik: Hinweise

- **Prüfungsaufgaben ähnlich zu Übungsblättern**
  - Rechenaufgaben genau anschauen
  - Sehr viel üben!

Mathematik

- **Youtube**
  - 3Blue1Brown (LA, HM, Mathe I, Analysis I)
  - Mathepeter (LA, HM, Mathe I, Analysis I)
  - blackpenredpen (HM, Mathe I, Analysis I)
  - Mathe by Daniel Jung (LA, HM, Mathe I, Analysis I)

Wirtschaftsinformatik I

Wirtschaftsinformatik I: Übungsschein

- **Zum Bestehen notwendig:**
  - Mindestens **75%** der Punkte
  - Alle 5 Übungsblätter müssen bearbeitet werden
- **Wird nur im WS angeboten**
- **Erfolgreiches bestehen des Übungsscheins gibt einen Notenbonus**

Wirtschaftsinformatik I: Tipps

- **Notenbonus mitnehmen**
- **Auswendig lernen!!**

Allgemeine Tipps

4

Übungblätter Management

- **Effektivität**
  - Die richtigen Ziele haben
  - Die Dinge machen, die deinen Zielen wirklich dienen
- **Effizienz**
  - Aufgaben bündeln, Tag in Sessions einteilen, …
  - Hilfsmittel nutzen, andere um Rat fragen
- **Vereinfachen (Priorisieren)**: mental aufräumen
  - Eisenhower-Prinzip
  - Pareto-Prinzip (80/20)

Lerntechniken

Lerntechnik	Modul	Beispiel
Karteikarten (z.B. Anki)	Alle	Definitionen lernen
Pomodoro	Alle	Hilft gegen Prokrastination
Deep Work	Mathe, Programmieren	Tief in Themen eintauchen
Feynman-Technik	Alle	Wissenslücken erkennen, Themen zusammenfassen
Spaced Repetition	Alle	Wissen verfestigen
Visualisieren (Mindmapping, Übersichten)	Winfo, GBI, Mathe	Überblick bekommen und Zusammenhänge verstehen
Distraction Delay	Alle	Fokussiert bleiben
Üben, üben, üben	Alle	Verständnis entwickeln

Exkurs: Stay Focused App

- **Ablenkungen reduzieren: blockiert Social Media & andere Apps während Lernzeiten**
- **Fokuszeit planen: Zeitfenster einstellen, in denen die App aktiv ist**

HoC - Lern LABOR:
Lern- und Arbeitstechniken

Organisieren und Planen	Recherche und Textbearbeitung	Struktur und Gedächtnis	Motivation und Selbstregulation	Entspannung und Bewegung
To Do-Liste, Tagesplan, Masterplan erstellen	Internetrecherche (Suchmaschinen)	Mind-Mapping	Selbstmotivierung	Bewegung und Lernen
Meilensteinplanung	Kooperatives Lesen	Strukturlegetechnik	Pomodoro Technik	Bewegungspausen gestalten
Planung mit der ALPEN-Methode	Markierungstechniken	Lernposter erstellen	Prokrastination Selbsttest	Jonglage
Planung mit dem Kanban-Kalender	Lesetechniken	Mnemotechnik: Schlüsselwort-Methode	Aufschiebetagebuch führen	Stressbewältigung
Prioritäten setzen mit der Eisenhower-Matrix	Lesemethode PQ4R (für komplexe Texte)	Mnemotechnik: Zahlentafel/Gedächtnistafel	Umgang mit Lampenfieber	Entspannungsübungen
SMARTe Lernziele setzen	Speed Reading/ Schnelles Lesen (Selbsttest)	Mnemotechnik: Akronyme	Survival Guide Prüfungsangst	Achtsamkeit
Pausenplanung		Mnemotechnik: Loci-Methode	Survival Guide Gruppenarbeit	Powernapping
Lernplan erstellen		Mnemotechnik: Karteikarten-Methode	Umgang mit Erfolg und Misserfolg	
Lerntagebuch schreiben		Vorlesungsmitschrieb, Vor- und Nachbereitung	Emotionsregulation: Das ABC-Schema	
Der häusliche Arbeitsplatz		Vorlesungsmitschrieb: Cornell Methode	Ressourcen aktivieren	
Strukturiert im Homeoffice		Deep Work	Selbstregulierung mit der Affektbilanz	
Survival Guide Prüfungsphase		Hubschrauber-Landkarten-Methode	Selbststeuerung mit dem Wenn-Dann-Plan	

- Karteikarten

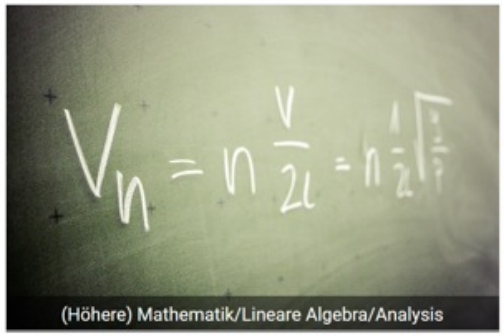
Materialsammlungen

- **Zusammenfassungen von Studierenden aus höheren** Semestern
  - Studydrive
  - KIT Mathe Info Discord

- **Vorlesung verpasst?**
  - Lineare Algebra I Videos im Ilias von WS 20/21
  - Grundbegriffe der Informatik Videos im Ilias
  - Programmieren Videos im Ilias

MINT-Kolleg

MINT-Kolleg: ***Semesterkurse*****, Helpdesk,** ***Aufbaukurse***



- **Bietet Semesterkurse** (kostenlos) **an zu**
  - Höhere Mathematik I
  - Lineare Algebra I
  - Programmieren
- **Im Ilias Kurs können Ressourcen genutzt werden, auch wenn man keinen Kursplatz hat**
- **Auch zwischen dem Ende der Winter-Vorlesungzeit und dem Beginn der Sommer-Vorlesungzeit bieten sie Kurse an,** wie zum Beispiel:
  - Lineare Algebra I
  - Programmieren

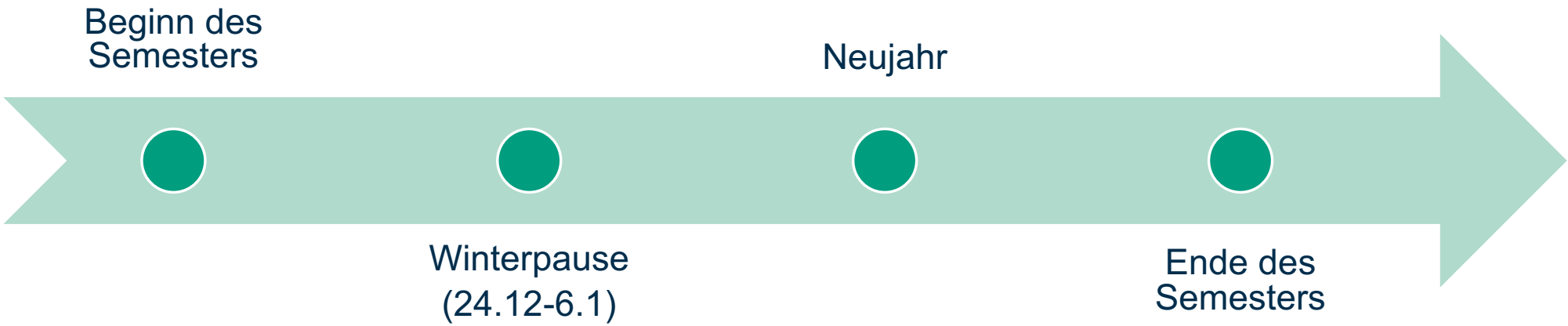
Fazit

- **Probiert verschiedene Lernmethoden aus!**
- **Jeder Mensch lernt anders – findet heraus, was für euch am besten funktioniert.**
- **Es muss nicht von Anfang an perfekt sein. Verbessert euch Schritt für Schritt.**

Reflexion

5

Reflexion und Motivation



Gut Reflektiert ist halb Gewonnen

- „Passt ein Studium zu mir?“
- „Passt der Studiengang zu mir?“
- „Was habe ich bisher erreicht?“

Ferien: Nimm dir Zeit für dich!

Wieder zur Uni

- Stundenplan überdenken
- Stoff wieder einarbeiten

Prüfungsvorbereitung beginnen

Reflexion: Ressourcen

Student Advisory Services (ZSB)

- **Zentrale Studienberatung (ZSB)**
  - Zweifel am Studium, Studiengang- /Hochschulwechsel, Studienabbruch, usw.
  - Clearing Sprechstunde

- **Angebote der Fakultät für Informatik**
  - Informatik Studiengangservice (ISS)
  - eezi-Beratungs Gepräch
  - Fachschaft (FSMI) / Wi-Forum

Informatics Study Program Services (ISS)

eezi-Gepräch

FSMI

Wi-Forum

- **Mit Studierenden oder Tutoren aus höheren Fachsemestern reden**

- **Mit sozialem Umkreis und Familie reden**

Letzter Abschnitt

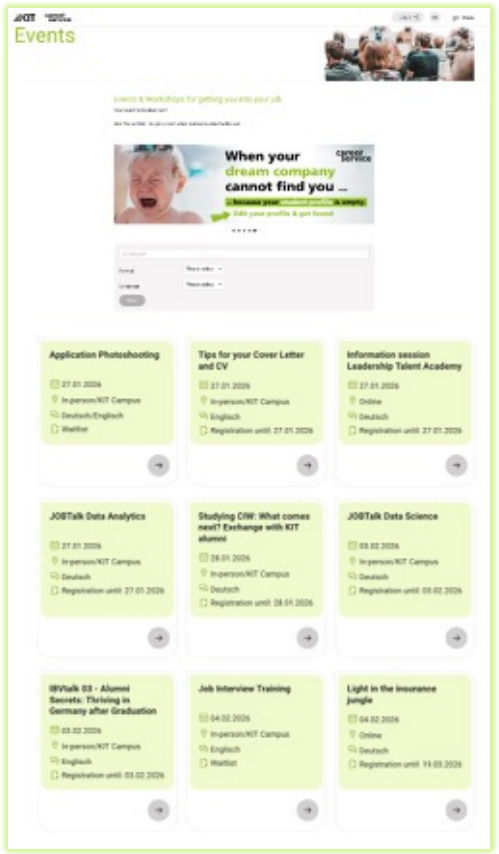
6

3. eezi-Vorlesung: „Orientierungsprüfungsvorbereitung“

- **Themen**
  - Vorbereitungsstrategie für Prüfungen
  - Fachliche Tipps zu Orientierungsprüfung
  - Anschließende Lernpartnerschaftsbörse / Studygroups für Prüfungausurenphase
- **Wann:** am **27.01.26,** um **17:30 Uhr**
- **Wo: Gebäude 50.34, Raum - 101** und auf Zoom


Mentoring
Get-Together
Game night
hangout and chill
with other students
16. Dezember 2025
at 5:45pm
in Building 50.34, Room 131
© Violka.art

Career Service Veranstaltungen



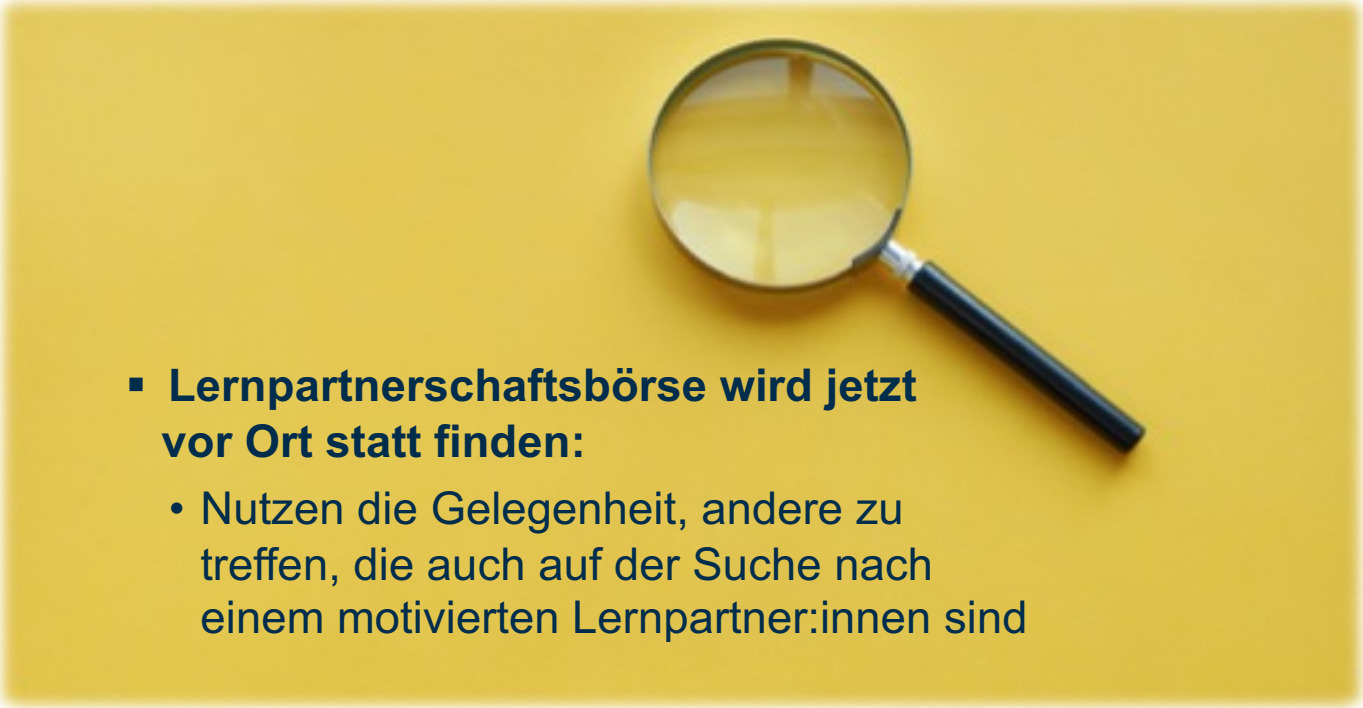
JOBTalks

- 26.11.2025 – Automation and autonomous machines / Automatisierung und autonome Maschinen
- 20.01.2026 – User Experience und Usability Engineering
- 27.01.2026 – Data Analytics
- 03.02.2026 – Data Science

Eure Fragen – unsere Antworten

- Fragen können auf Deutsch oder Englisch

Danke für eure Aufmerksamkeit!



- **Lernpartnerschaftsbörse wird jetzt vor Ort statt finden:**
  - Nutzen die Gelegenheit, andere zu treffen, die auch auf der Suche nach einem motivierten Lernpartner:innen sind

- **Ressourcen**

eezi Ilias-Kurs

Youtube-Kanal