Karlsruher Institut für Technologie

“Programmieren Best Practices“

mit Lernpartnerschaftsbörse

Dienstag, 14.05.2024, um 17:45 Uhr

https://www.informatik.kit.edu/imstudium.php

Agenda

- Einführung
- „Programmieren Best Practices“ mit Timur Sağlam (SDQ)
- Q&A
- Lernpartnerschaftsbörse

Best Practices für „Programmieren“

Info^2 | Info-Bites | Timur Sağlam

KASTEL – Institute of Information Security and Dependability
MCSE – Modelling for Continuous Software Engineering Group

Das Programmieren-Team

Yves Kirschner

Dominik Fuchß

Timur Sağlam

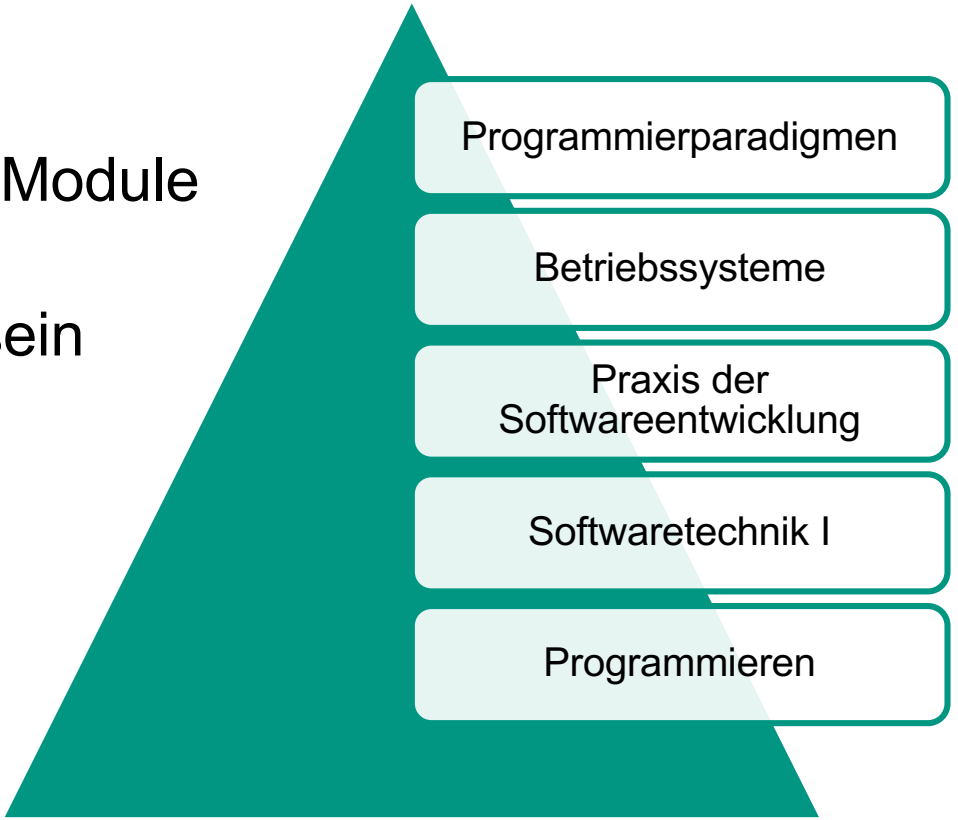
KASTEL – Institute of Information Security and Dependability
MCSE – Modelling for Continuous Software Engineering Group

Best Practices für „Programmieren“

- Heutige Ziele
  - Teilen von Erfahrungen, woran es scheitern könnte
  - Tipps, um Frust zu sparen und beste Chancen zu haben
- Nicht alle Praktiken werden bei den ersten Blättern anwendbar sein
  - Es muss nicht unbedingt alles sofort verstanden werden
  - Aber spätestens bei den Abschlussaufgaben solle darauf zurückgegriffen werden

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 Hilfsmittle
- s.kit.edu/Unterstützung im Studium

MINT

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

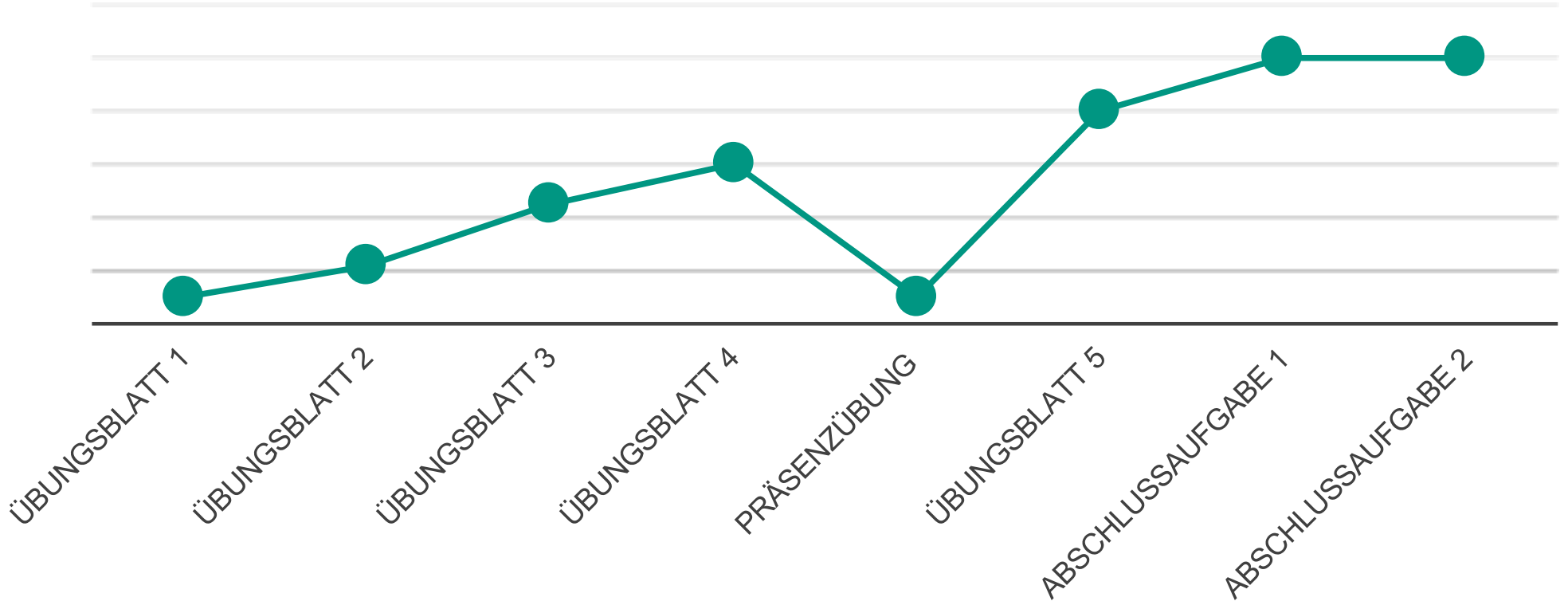
Forum-Wi / (Aktive) Fachschaft

- 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
  - Von Anfang an „saubere“ Programmierung denken und sich daran gewöhnen
- 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

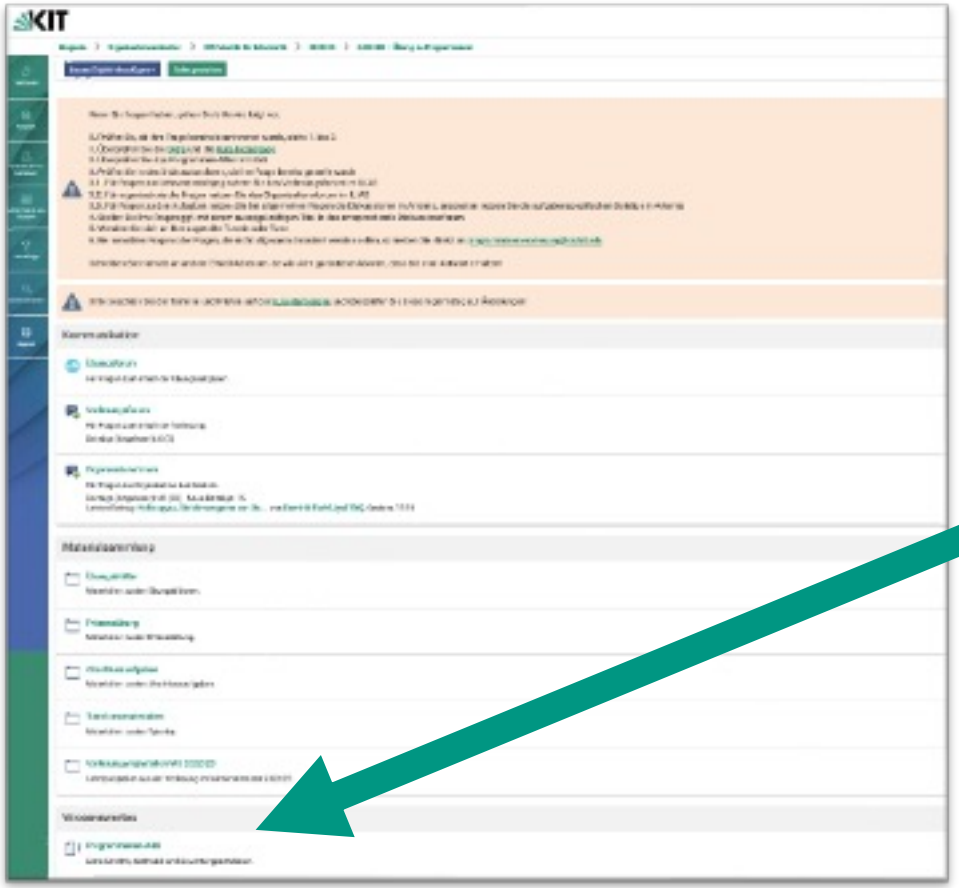
Praktische Tipps für die Aufgaben

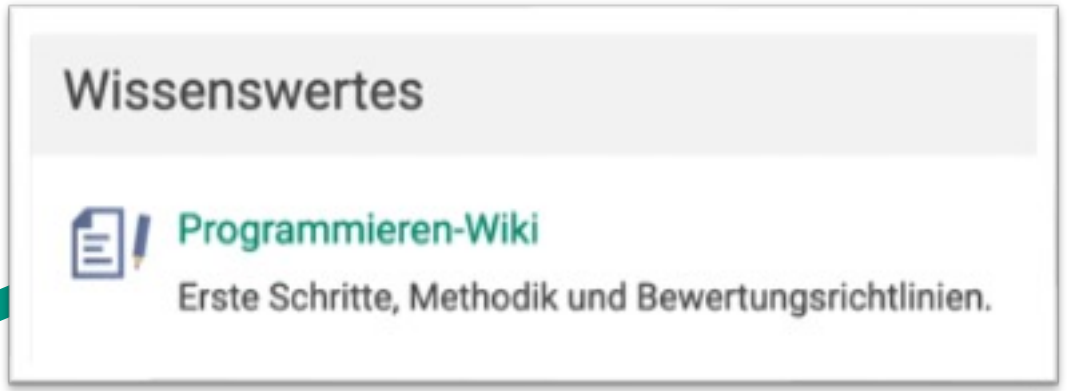
- Quelltextklone vermeiden
  - Sich nicht Wiederholen und alles nur einmal im Quelltext implementieren
  - Aus doppelten Quelltextstellen versuchen, Methoden zu extrahieren
- Lange Methoden vermeiden
  - Methoden zerteilen und auf mehrere kleine (private) Hilfsmethoden verteilen
  - Dokumentation der 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
  - Final, wenn nur einmal zugewiesen
  - Nur Klassenvariablen sollten statisch sein
- Herangehensweise
  - Aufgabestellung genau lesen
  - Exakt die erwartete Ausgabe liefern
  - Bei Unklarheiten sofort nachfragen

Methodik Abzüge

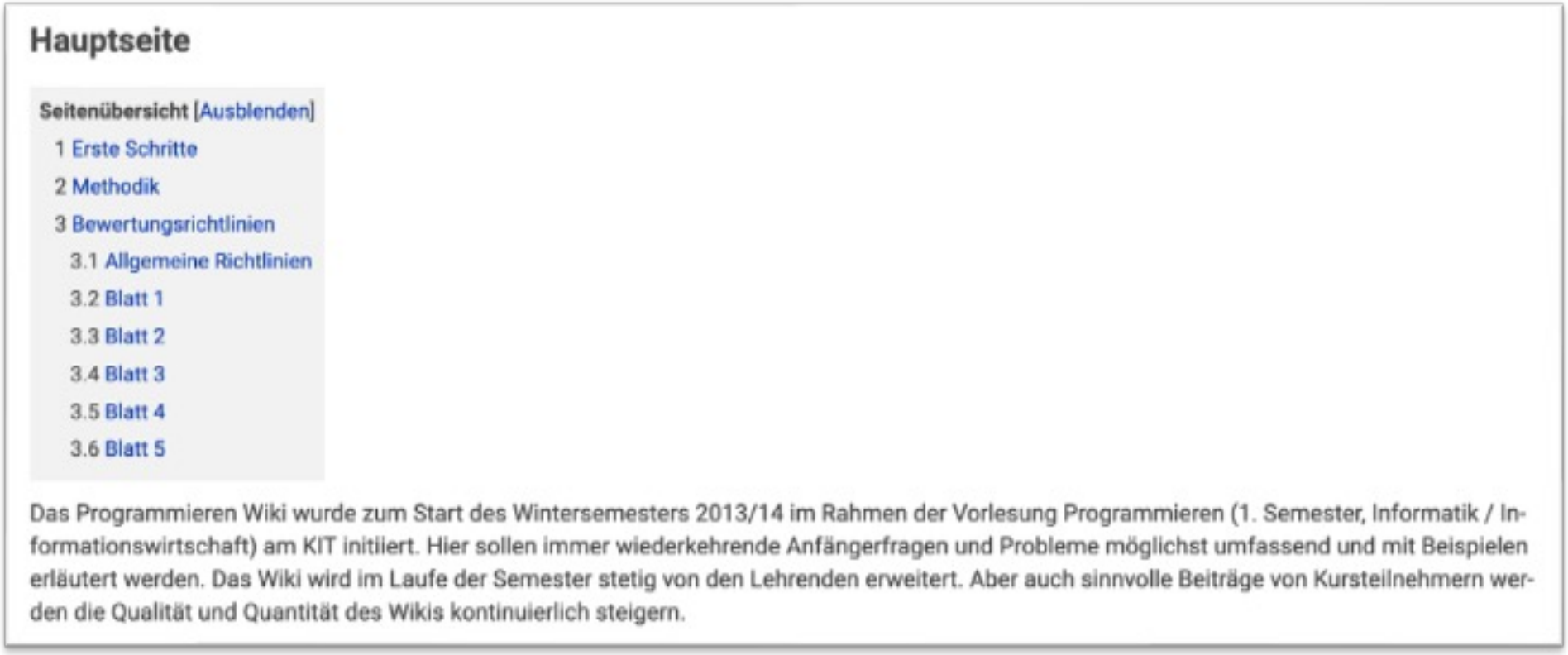
- Neben Funktionalität wird auch Methodik geprüft
- Großer Teil der Bewertung von Übungsblättern und Abschlussaufgaben
- Alle Bewertungskriterien sind im Ilias-Wiki: *s.kit.edu/wiki*

Ilias-Wiki




Wissenswertes
Programmieren-Wiki
Erste Schritte, Methodik und Bewertungsrichtlinien.

Ilias-Wiki: Übersicht

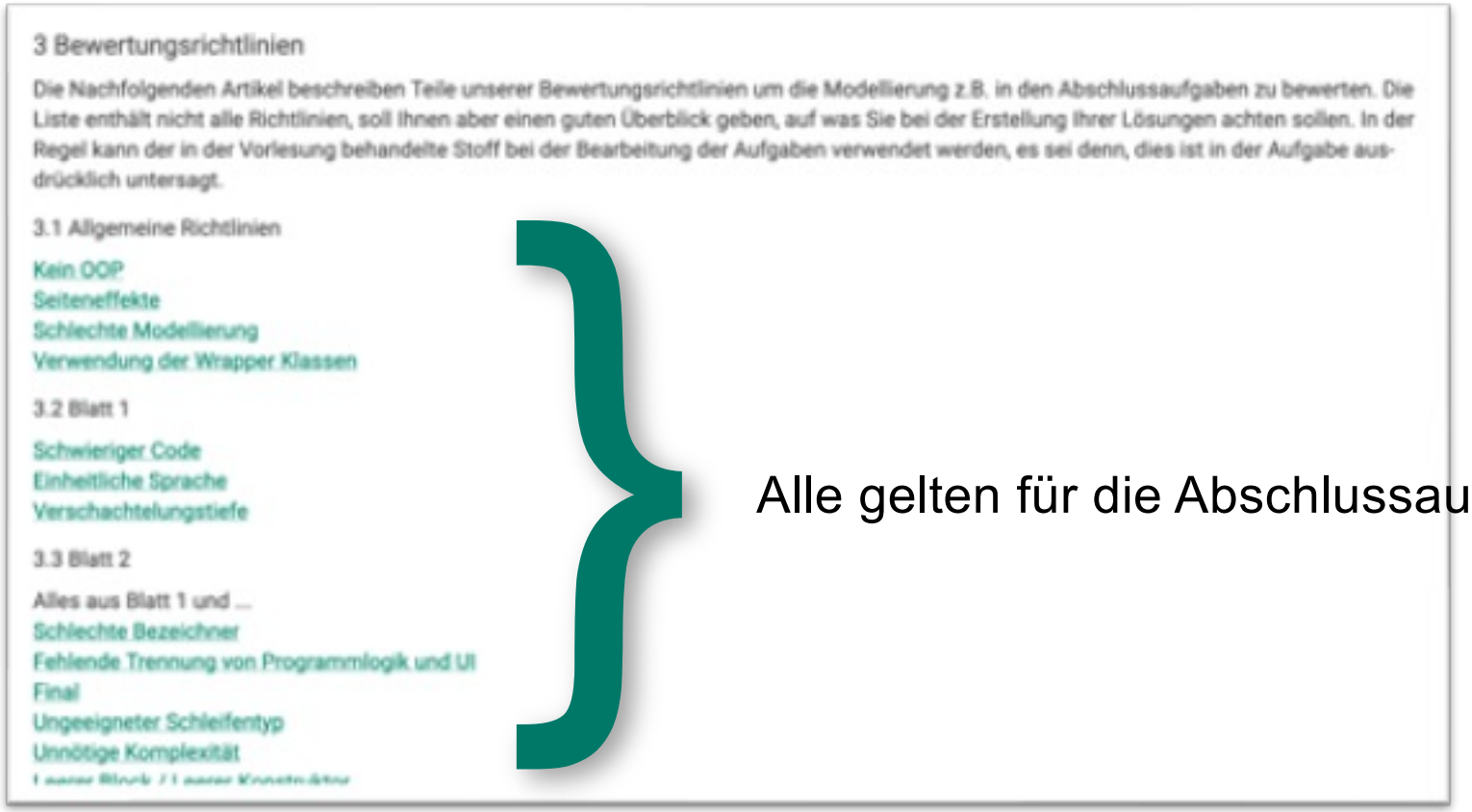


Hauptseite

Seitenübersicht [Ausblenden]

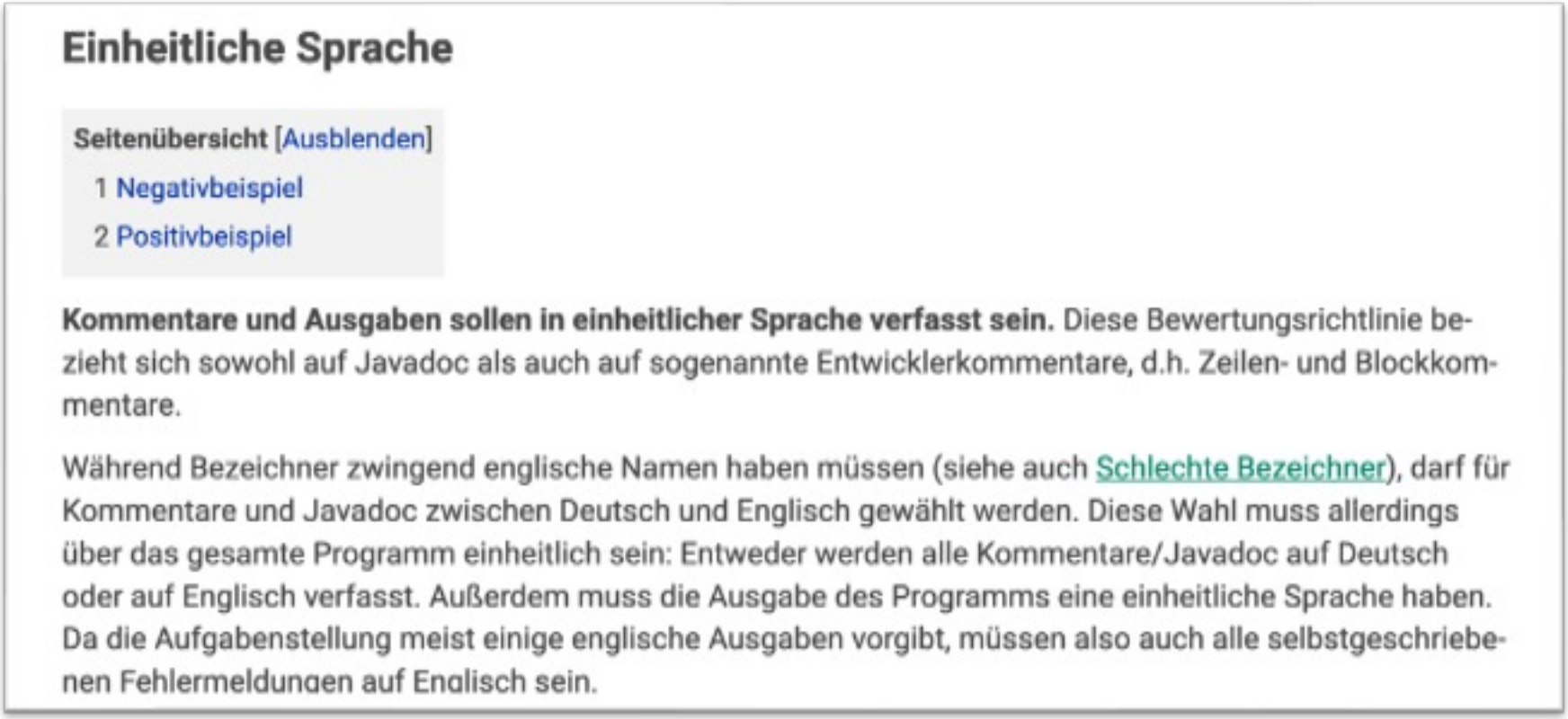
Das Programmieren Wiki wurde zum Start des Wintersemesters 2013/14 im Rahmen der Vorlesung Programmieren (1. Semester, Informatik / Informationswirtschaft) am KIT initiiert. Hier sollen immer wiederkehrende Anfängerfragen und Probleme möglichst umfassend und mit Beispielen erläutert werden. Das Wiki wird im Laufe der Semester stetig von den Lehrenden erweitert. Aber auch sinnvolle Beiträge von Kursteilnehmern werden die Qualität und Quantität des Wikis kontinuierlich steigern.

Ilias-Wiki: Bewertungsrichtlinien



Alle gelten für die Abschlussaufgaben!

Ilias-Wiki: Einheitliche Sprache



Einheitliche Sprache

Seitenübersicht [Ausblenden]

1 Negativbeispiel

2 Positivbeispiel

Kommentare und Ausgaben sollen in einheitlicher Sprache verfasst sein. Diese Bewertungsrichtlinie bezieht sich sowohl auf Javadoc als auch auf sogenannte Entwicklerkommentare, d.h. Zeilen- und Blockkommentare.

Während Bezeichner zwingend englische Namen haben müssen (siehe auch Schlechte Bezeichner), darf für Kommentare und Javadoc zwischen Deutsch und Englisch gewählt werden. Diese Wahl muss allerdings über das gesamte Programm einheitlich sein: Entweder werden alle Kommentare/Javadoc auf Deutsch oder auf Englisch verfasst. Außerdem muss die Ausgabe des Programms eine einheitliche Sprache haben. Da die Aufgabenstellung meist einige englische Ausgaben vorgibt, müssen also auch alle selbstgeschriebenen Fehlermeldungen auf Englisch sein.

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");
    }
}
```


KASTEL – Institute of Information Security and Dependability
MCSE – Modelling for Continuous Software Engineering Group

Ilias-Wiki: Einheitliche Sprache

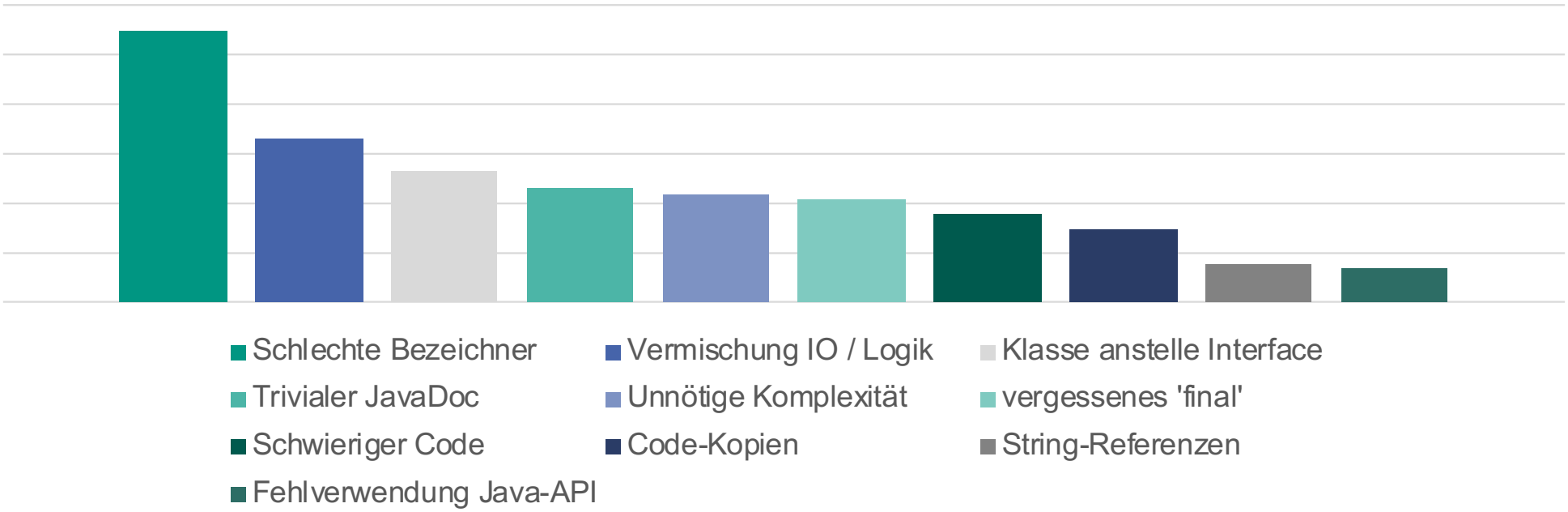
2 Positivbeispiel

```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");
    }
}
```


KASTEL – Institute of Information Security and Dependability
MCSE – Modelling for Continuous Software Engineering Group

Die häufigsten methodischen Fehler bei Abschlussaufgaben



Weitere Bewertungsrichtlinien im Wiki: s.kit.edu/wiki

Schlechte Bezeichner

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

```
1  public static int s(final int a, final int b) {
2      return a - b;
3  }
```

```
1  public static int subtract(final int minuend, final int subtrahend) {
2      return minuend - subtrahend;
3  }
```

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;
    }
}
```


KASTEL – Institute of Information Security and Dependability
MCSE – Modelling for Continuous Software Engineering Group

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;
}
```

KASTEL – Institute of Information Security and Dependability
MCSE – Modelling for Continuous Software Engineering Group

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

```
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

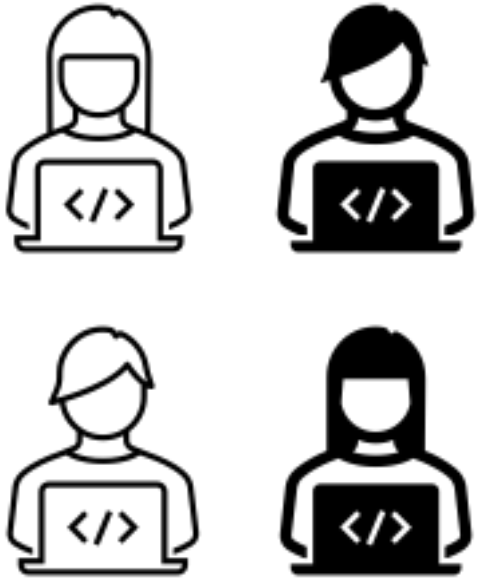
- Fragen und Unklarheiten können immer auftreten
  - Wenn ihr eine Frage habt, haben andere vielleicht die gleiche Frage
  - Wir bieten viele Möglichkeiten, Fragen zu stellen
- Bei Rückfragen werden Aufgaben gegebenenfalls aktualisiert
  - Lediglich minimale Aktualisierungen oder weitere Klarstellungen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

Reihenfolge bei Fragen

1. Websuche
2. FAQ / Programmieren-Wiki prüfen
   - *s.kit.edu/faq*
   - *s.kit.edu/wiki*
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



Links und Anlaufstellen:

- Programmieren Seite: https://sdq.kastel.kit.edu/wiki/Übung_zu_Programmieren_SS24
- Fachschaft Mathematik / Informatik: Beratung durch Studierende des Fachs Informatik / Mathematik: https://www.fsmi.uni-karlsruhe.de/
- Forum Wirtschaftsinformatik: Beratung durch Studierende der Fachrichtung Wirtschaftsinformatik: https://www.forum-wi.de/
- MINT-Kolleg Baden-Württemberg: Vorkurse, Semesterkurse (Einführung in die Programmierung mit Java), Aufbaukurse Informatik (in der vorlesungsfreien Zeit, usw.): https://www.mint-kolleg.kit.edu/index.php
- Unterstützung im Studium (Fakultät für Informatik)
  - Webseite: https://www.informatik.kit.edu/10509.php
  - Info²: Folien PDF und Link zu Video vom heutiges Veranstaltung
  - YouTube Playlist: https://www.youtube.com/playlist?list=PL22ZNLSohCREnOwW5FzHO1SGP-zK8oEGx

Danke für Ihre Aufmerksamkeit!