Hochschule
Kaiserslautern
University of
Applied Sciences

Objektorientierte Programmierung (OoPr)

WS 2021/22

Bachelor Wirtschaftsinformatik (PO2020)

XX

Hochschule Kaiserslautern

30. Dezember 2021

Lernziele

Zentrales Lernziel

Am Ende der Vorlesung können Sie
ein Backend[1] einer Webapplikation
in Java programmieren.

[1] ohne Datenbank

Detaillierte Lernziele

- LO1: Sie können Java-Programme entwickeln, die vorgegebene Anforderungen erfüllen.
- LO2: Sie können grundlegende objektorientierte Programmierkonzepte anwenden, um **effizient programmieren** zu können.
- LO3: Sie können Hilfstechniken zur Vermeidung/Findung/Analyse von Fehlern in Software anwenden, um **hochqualitative** Software zu erstellen.
- LO4: Sie können durch eigene Recherchen ihre konzeptionellen und technischen Fähigkeiten erweitern, wenn Sie beim Programmieren an ihre Grenzen stoßen, um später ihre **Kompetenzen selbstständig erweitern** zu können.
- LO5: Sie erarbeiten Problemlösungen **konzeptionell**, indem Sie zunächst einen Lösungsentwurf erstellen, um dadurch schneller und robuster zu programmieren.
- LO6: Sie können Software gemeinsam **im Team entwickeln**, da dies in der IT-Welt gängige Praxis ist.

Organisation

Corona-Regelungen

- Es gilt 3G: Genesen, Geimpft oder Getestet
- Kommen Sie nicht zu Präsenz-Veranstaltungen, wenn Sie mögliche Corona-Symptome aufweisen
- In Präsenz: Immer per QR-Code ein- und auschecken


GitLab

Flipped-Classroom-Ansatz

- Pro Lerneinheit (je eine Woche) wird zur Verfügung gestellt:
  - Ein Video (20-30 Minuten)
  - Eine Übung
  - Zum Nachlesen bei Bedarf: Ein Abschnitt aus „Java ist auch eine Insel“ (OpenBook: `https://openbook.rheinwerk-verlag.de/javainsel/`)
- Vorgehensweise:
  1. Sie bearbeiten die Übung mithilfe des Videos (gerne auch das Buch zu Rate ziehen)
  2. Sie geben die Übung **vor** der Vorlesung bei `benjamin.berzel@hs-kl.de` ab[2].
  3. DANACH treffen wir uns und:
     - besprechen die Schwierigkeiten, die Sie bei der Bearbeitung der Übung hatten
     - vertiefen gemeinsam die Inhalte
     - Ich erläutere Hintergründe zu verwendeten Technologien (Java, Git, JUnit, Spring, Maven, REST, etc.)

[2] später direkt über GitLab

Was Flipped-Classroom **nicht** ist

- Sie haben das Video im Besten Fall überflogen (morgens an der Bushaltestelle).
- Für die Übung hatten Sie keine Zeit mehr und können daher auch keine Frage stellen.
- Sie kommen unvorbereitet in die Hybrid-Veranstaltung und versuchen zu folgen.
- Das ist **Zeitverschwendung**. :)

Aufwand der Veranstaltung

- Umfang: 5 CP
- 1 CP entspricht ca. 25-30 h Arbeitsaufwand pro Semester
- Arbeitsaufwand: 125-150 h
- 15 Vorlesungswochen
- $\Rightarrow$ Ca. 8-10h pro Woche
- $\Rightarrow$ **4-6h außerhalb der Vorlesung**

Hochschule Kaiserslautern
University of Applied Sciences

Wie geht's los

1. Mithilfe der SETUP-Videos (heute in der Veranstaltung finalisieren):
   1.1 Java JDK installieren
   1.2 Eclipse installieren
   1.3 GitLab-Integration einrichten
2. Erste Übung aus GitLab pullen (branch master):
   clone https://gitlab.rlp.net/eugen.staab/basic-java-application.git
3. Bearbeitete Übung (nur die Datei „AssignmentOne.java“) vor der nächsten Übung per E-Mail an XX schicken.

Vorlesungskultur

- Die Vorlesung wird anspruchsvoll
- Wenn Sie am Ball bleiben, werden Sie erfolgreich sein!
- Offene Fehlerkultur
- Wir helfen Ihnen:
  - Ich stehe jederzeit für Fragen zur Verfügung!
  - Helfen Sie sich gegenseitig!

Projektarbeit

Organisatorisches

- Arbeit in 3er-Teams:
  - Jedes Team bekommt ein Projekt in GitLab
  - Das Projekt enthält bereits die Grundstruktur Ihres Projektes
- Programmierung einer Software
- Ausgabe des Themas: 01.12.2021
- Deadline: 22.02.2022
- Bitte sorgen Sie dafür, dass bis zum 8.12.2021 Ihre Infrastruktur funktioniert. Bis dahin helfen XX und ich gerne bei technischen Problemen.

Bewertung der Projektarbeit

- Bewertet wird zu 50 % die Individualleistung und zu 50 % die Teamleistung.
- Bewertung nach folgenden Kriterien:
  - Wie gut deckt das Programm die Anforderungen ab?
  - Wie viele Bugs enthält die Software (stichprobenartige Tests)?
  - Wie selbsterklärend sind die Variablen und Methoden benannt?
  - Wie passend ist der Code dokumentiert (dort, wo notwendig)?
  - Wie sinnvoll kamen objektorientierte Konzepte zum Einsatz?
  - Wie ausführlich wurde der Code reviewed?
  - Wie regelmäßig wurde gepusht?
  - Bonus: Wie viele automatisierte Tests wurden geschrieben?
  - Bonus: Wie gut sieht die REST-Schnittstelle aus?
  - Bonus: Wie sinnvoll kamen funktionale Konzepte zum Einsatz?

Projektarbeit: Thema

Terminplaner für eine Autowerkstatt

Mit der Software arbeiten *Disponenten* und *Automechaniker*:

- Disponenten planen Termine für Automechaniker und Kundenberater.
- Automechaniker müssen sehen können, was ihre nächsten Arbeiten an welchem Kundenfahrzeug sind.

Source: `https://www.faz.net/`

Projektphasen

- Die Anforderungen sind eingeteilt in drei Phasen:

Source: https://fixate.it

- Eine Phase darf erst dann angefangen werden, wenn die vorangegangene Phase *vollständig umgesetzt* ist.

Projektarbeit: Formulierung der Anforderungen

- Die funktionalen Anforderungen sind in Form von User Stories beschrieben
  - User Story: „Als *Benutzer X* möchte ich *Y* tun können, um *Z* zu erreichen.“
  - Wirklich programmieren müssen Sie in diesem Falle nur *Y* – der Rest hilft Ihnen die Anforderung besser zu verstehen.
- Jede Anforderung hat eine eindeutige Anforderungs-ID (z. B. /MVP020/), damit wir uns leichter darüber austauschen können.

Projektarbeit: Anforderungen an das MVP (1/2)

/MVP010/ Als Disponent möchte ich **Arbeiten**, jeweils mit ID, Namen und Dauer, verwalten können, damit ich diese nicht für jeden Termin neu anlegen muss.

/MVP020/ Als Disponent möchte ich **Arbeitsbühnen**, jeweils mit ID und Name verwalten können, damit ich diese nicht für jeden Termin neu anlegen muss.

/MVP030/ Als Disponent möchte ich **Fahrzeuge** mit eindeutigem Kfz-Kennzeichen, Marke, Modell, Baujahr und Zulassungsdatum verwalten können, um später deren Reparaturhistorie nachvollziehen zu können.

/MVP040/ Als Disponent möchte ich **Kunden** mit eindeutiger Kundennummer, Name, Rechnungs- und Kontaktdaten und einer Menge von Fahrzeugen verwalten können, um für die Kunden Termine anlegen zu können.

Projektarbeit: Anforderungen an das MVP (2/2)

/MVP050/ Als Disponent möchte ich **neue Termine** anlegen können, um so eine Planung für die Zukunft erstellen zu können; es gibt drei Arten von Terminen:

- Arbeitstermin: Termin für eine Menge von *Arbeiten* an einem *Kundenfahrzeug* auf einer *Arbeitsbühne*.
- Beratungstermin: Termin mit frei gewählter Dauer für eine Beratung eines *Kunden*.
- Reinigungstermin: Termin zur Reinigung einer *Arbeitsbühne* (Schnellreinigung: 30 Minuten, Intensivreinigung: 60 Minuten)

/MVP060/ Als Disponent möchte ich eine **Wochenübersicht** aller Termine einsehen können, um zu sehen, wo noch freie verfügbare Termine für die Bühnen sind.

Projektarbeit: Anforderungen für das MMP (1/2)

/MMP010/ Als Disponent möchte ich **Benutzer** der Typen *Disponenten*, *Kundenberater* und *Automechaniker* jeweils mit Benutzername, Vor- und Nachname verwalten können.

/MMP020/ Als Disponent möchte ich beim Anlegen eines Arbeitstermins einen **Automechaniker**, beim Anlegen eines Beratungstermins einen **Kundenberater** und beim Anlegen eines Reinigungstermins einen **Disponenten** mitangeben müssen, damit klar ist, wer die Verantwortung für die Durchführung der entsprechenden Tätigkeit hat.

/MMP030/ Als Automechaniker möchte ich einen Arbeitstermin als **erledigt**, oder **abgebrochen** markieren können, um so dem Disponenten die Information zu geben, ob er die Arbeit fakturieren kann (Faktura nicht Teil dieser Software!).

Projektarbeit: Anforderungen für das MMP (2/2)

/MMP040/ Als Disponent möchte ich **alle Arbeitstermine des gestrigen Tages** einsehen können, die erledigt wurden, um zu wissen, was ich fakturieren kann.

/MMP050/ Als Disponent möchte ich alle Arbeiten sehen, die in der Vergangenheit an einem bestimmten Fahrzeug durchgeführt wurden (die „Fahrzeug-Historie“), um den aktuellen Stand des Fahrzeugs besser verstehen zu können.

Projektarbeit: Anforderungen für das MLP (1/2)

/MLP010/ Als Automechaniker möchte ich sehen können, welche **Arbeiten ich heute noch auf welcher Bühne erledigen** muss, um zu wissen, was mich als nächstes erwartet und ich mich entsprechend darauf vorbereiten kann.

/MLP020/ Als Disponent möchte ich eine **Fehlermeldung** bekommen, wenn ich versuche einen Termin zu buchen, der mit einem anderen Termin auf der gleichen Bühne **kollidiert**, um dann einen alternativen Termin buchen zu können.

/MLP030/ Als Disponent möchte ich die Möglichkeit haben, automatisch den nächstbesten Termin für einen Reinigungstermin (schnell oder intensiv) auf einer bestimmten Bühne buchen zu lassen, um mit möglichst wenig Aufwand eine Reinigung einplanen zu können.

Projektarbeit: Anforderungen für das MLP (2/2)

/MLP040/ Als Disponent möchte ich für einen Arbeitstermin die **nächsten drei verfügbaren Termine** vorgeschlagen bekommen, um diese dem Kunden als Vorschlag unterbreiten zu können.

Projektarbeit (1/2)

Vorgehensweise

1. Überlegen Sie sich ein UML-Klassendiagramm, das Ihre Klassen darstellt. Alternativ ein Entity-Relationship-Modell.
2. Lassen Sie Ihr Projekt auf der vorgegebenen Grundstruktur basieren (Vorschlag):
   - Ihr Package: `de.hs_kl.staab.planner`
   - Klassenstruktur:

     `PlanningService` Stellt alle Funktionen für die Planung des Kalenders bereit und ruft andere Services auf (beispielsweise Services für Kunden oder Bühnen). Hat eine Instanz des:

     `PlanningCalendar` Repräsentiert den Terminkalender und dessen Daten selbst.

     `PlanningController` (Optional!) Greift auf den `PlanningService` zu und stellt die Funktionalität über eine REST-Schnittstelle nach außen (über einen Applikationsserver) zur Verfügung.

Projektarbeit (2/2)

Vorgehensweise

3. Lieber weniger Funktionalität, aber höhere Qualität!
   - Fügen Sie peu-à-peu neue Funktionen Ihrem System hinzu, aber versuchen Sie möglichst zu jedem Zeitpunkt im Git eine lauffähige, „saubere“ Software zu haben.
   - Bevor Sie neue Funktionen hinzufügen, sollten die Software in einen lauffähigen, sauberen Zustand gebracht werden.
4. Pushen Sie häufig, reviewen Sie viel und testen Sie noch mehr!

Was bedeutet „verwalten“?

- Viele Anforderungen verwenden den Begriff „verwalten“.
- Beispiel: Der Benutzer soll die Möglichkeit haben Kunden zu verwalten.
- Im Rahmen dieser Anforderungen sind damit immer diese Funktionalitäten gemeint:
  - Eine Methode soll die angefragten Daten zurückgeben können
  - Eine Methode soll neue Daten hinzufügen können
  - Eine Methode soll einen Datensatz überschreiben können; Tipp: Anhand der ID können Sie den alten Datensatz löschen und dann die Methode zum hinzufügen aufrufen.
- Im Beispielfall könnten das die folgende Methoden sein (im `CustomerService`):

```
public Customer getCustomerById(int customerId) {...};
public List<Customer> getCustomers() {...}; // Ggf. mit Parametern
public void createNewCustomer(...) {...};
public void updateCustomer(Customer customer) {...};
```

Themen der nächsten Vorlesungen

08.12.21	Automatisiertes Testen, JavaDoc
15.12.21	REST-Apis und Swagger
22.12.21	Funktionale Programmierung
05.1.21	Clean Code
12.1.21	Offene Fragerunde
19.1.21	**Gastvortrag:** XX überSpring