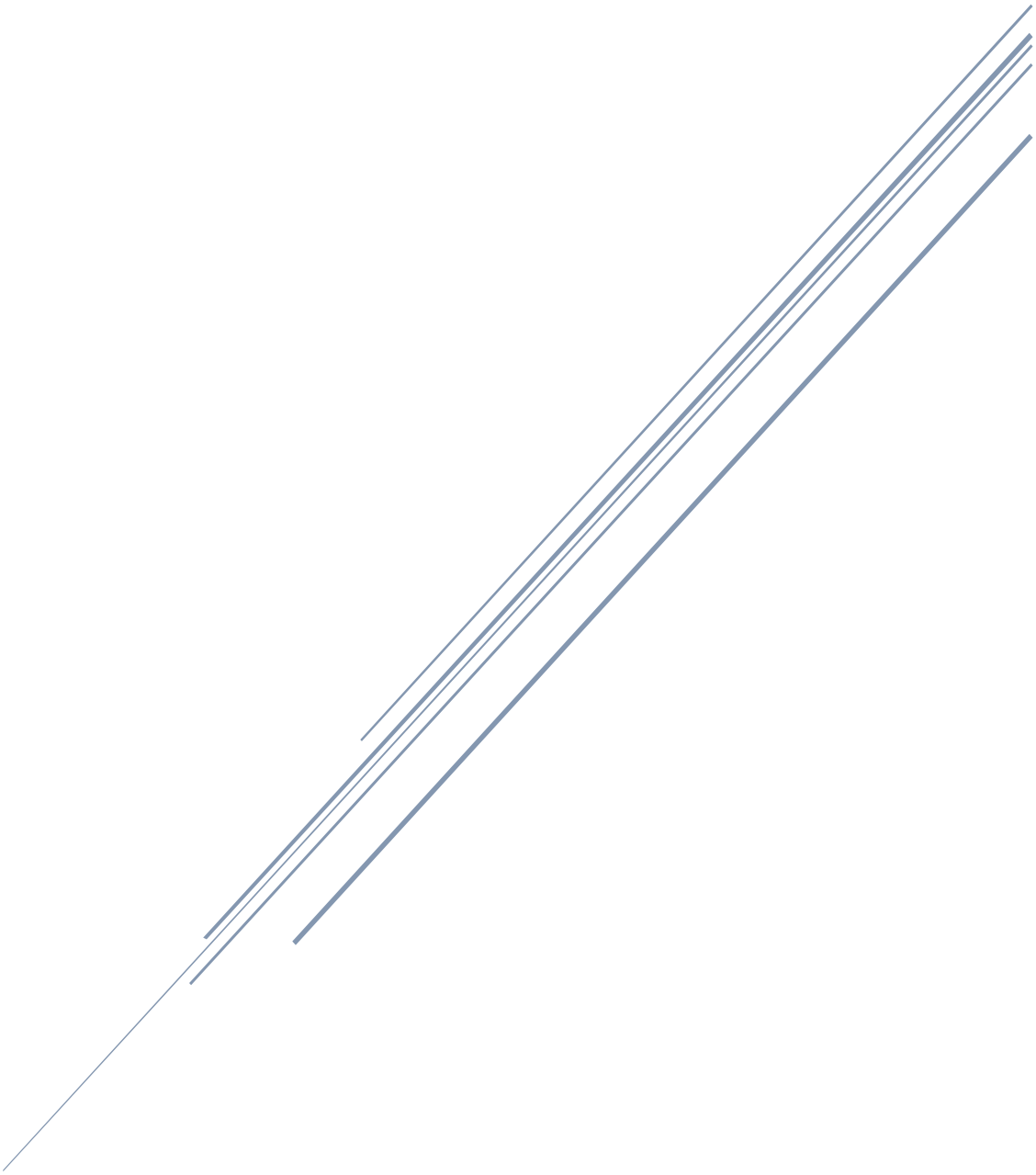
OOP-SEMESTERPROJEKT

Erik Nissen, Nico Johnsen, Pascal Groß



Fachhochschule Kiel

Objektorientierte Programmierung (in Java)

Inhalt

Einleitung

Während der Corona-Pandemie im Jahr 2020 kam es zu einem Paradigmenwechsel: Von der sogenannten Work-Life-*Balance* wurde übergegangen in die Work-Life-*Integration*. Für viele Menschen bedeutet dies, digitale Arbeit für ihr Unternehmen in den eigenen vier Wänden zu verrichten. Damit einher geht unter anderem, zahlreiche neue Produkte in den eigenen Haushalt zu integrieren. Einen mobilen Computer zum Beispiel, eine stationäre Dockingstation, ein kabelgebundenes oder kabellose Headset, eine weitere Maus, eine weitere Tastatur, mehr Kabel, mehr Akkus, mehr Elektronik. Außerdem bedeutet das Arbeiten von zu Hause, auch die Nahrungsaufnahme in den eigenen vier Wänden durchzuführen. Kurzum: Durch den coronabedingten Paradigmenwechsel verwenden Menschen auf eine neue Art Elektronikprodukte und Lebensmittelprodukte.

Die im Folgenden beschriebene Anwendung soll es ihren Anwendern ermöglichen, Tabellen anzulegen für Elektronik- und Lebensmittelprodukte, bestehende Daten zu aktualisieren, auszugeben oder zu löschen.

Schnellstart-Guide

Vor dem ersten Start

Die Anwendung wurde geschrieben in der Programmiersprache Java in Version 15.0.2. Voraussetzung für die Ausführung der Anwendung ist, **mindestens Java-Version 11** zu verwenden. Darüber hinaus basiert die Projektstruktur auf dem **Maven-Standard**. Maven ist der Standard-Package-Manager im Java-Ökosystem. Vor dem ersten Start der Anwendung ist ergo sicherzustellen, dass Maven auf dem verwendeten Gerät verfügbar ist. Gemäß Mindestanforderungen ist bei den weiteren Schritten außerdem die IDE IntelliJ Community Edition 2020.3.x zu verwenden.

Die Anwendung das erste Mal starten

1. Das Projekt `OOP_Semesterprojekt.zip` herunterladen
2. Die Datei `OOP_Semesterprojekt.zip` an einem beliebigen Ort entpacken
3. Rechtsklicken auf den entpackten Ordner `OOP_Semesterprojekt` und die Option `Open Folder as IntelliJ IDEA Community Edition Project` wählen
    a. Alternativ den Ordner OOP_Semesterprojekt via IntelliJ importieren
4. In IntelliJ: Im Terminalfenster den Befehl "mvn clean install" eingeben und ausführen
5. In IntelliJ: Rechtsklicken auf "`Anwendung.java`". Den Befehl "`Run`" ausführen

Anschließend wird eine Bildschirmausgabe erzeugt. Außerdem kann in einem Webbrowser die URL http://localhost:8080 aufgerufen werden. Weitere Details über die Bedienung der Anwendung werden im Abschnitt Anwendungsdetails beschrieben.

Architektur

Kernkomponente der vorliegenden Anwendung ist das Java-Framework Spring Boot. Die Anwendung bietet REST-Schnittstellen an. Dabei werden Daten persistent in einer H2-Datenbank gespeichert, das Mapping von Java-Klassen auf Datenbanktabellen erfolgt über Hibernate.

Klassendiagramm nach UML-Standard

Die Anwendung arbeitet mit einer Basisklasse `Produkt` und den daraus abgeleiteten Unterklassen `Lebensmittelprodukt` und `Elektronikprodukt`.

Vorgriff auf den folgenden Abschnitt MVC-Design-Pattern: Die Klassen fungieren als Modell und enthalten ausschließlich Attribute mit der Sichtbarkeit `private` sowie Getter und Setter mit der Sichtbarkeit `public`. Der Übersicht wegen wurden im Klassendiagramm nicht alle Getter und Setter aufgeführt.

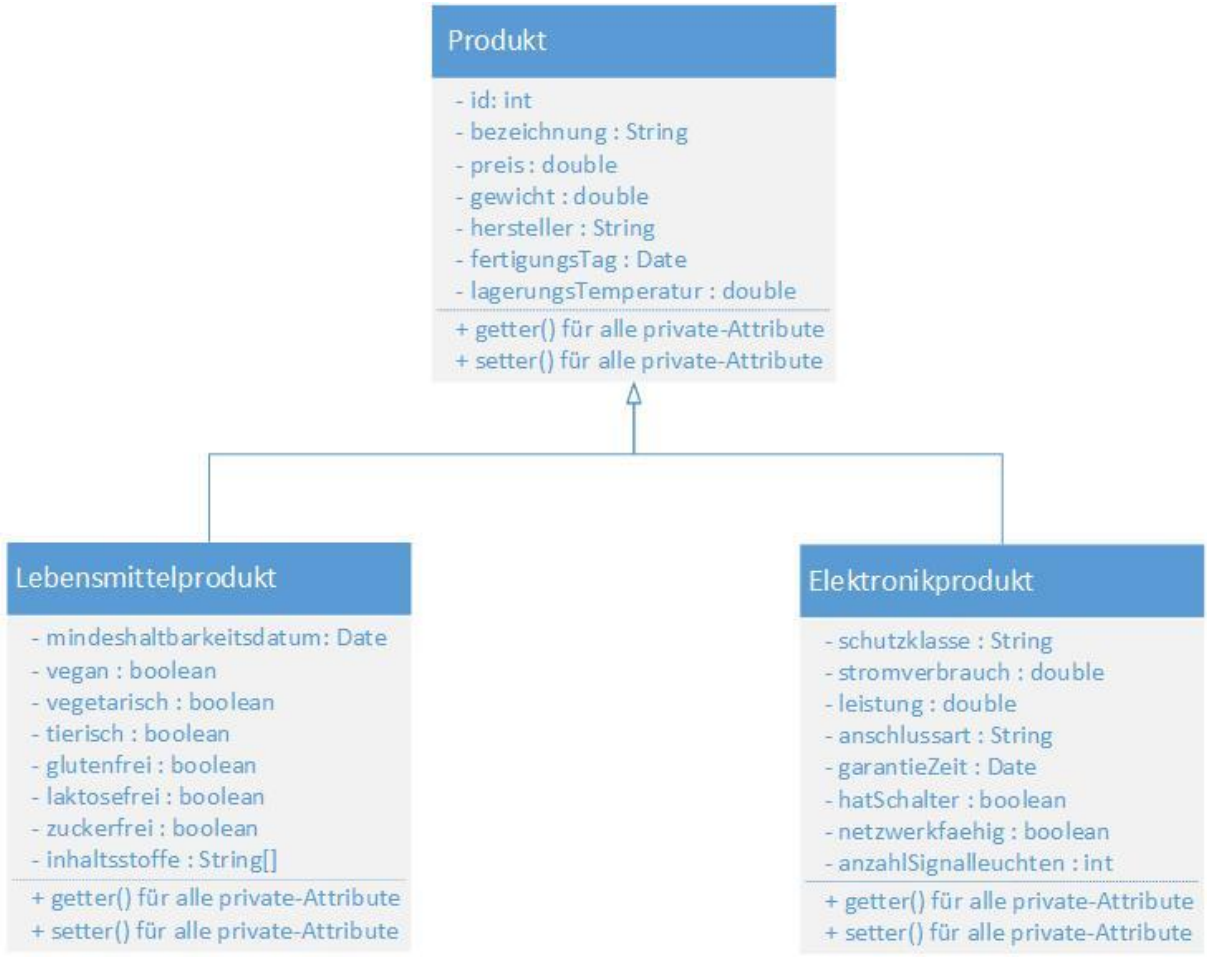


Abbildung 1: Klassendiagramm nach dem UML-Standard

Software-Design-Pattern: MVC-Design-Pattern

Die Anwendung wurde strukturiert nach dem Model-View-Controller-Pattern, kurz: MVC-Pattern. Es gibt die im Abschnitt Klassendiagramm nach UML-Standard vorgestellten Klassen `Produkt`, `Lebensmittelprodukt` und `Elektronikprodukt`, wobei die beiden zuletzt genannten aus der erstgenannten abgeleitet sind. Dabei enthalten diese drei Klassen ausschließlich Attribute, Getter und Setter. Die drei Klassen befinden sich im Package `model`.

View in MVC steht für die Visualisierung der Daten, welche im Model enthalten sind. Die Visualisierung der Daten erfolgt in der Anwendung über die Klasse `Objektdatenbank.java`, welche Methoden enthält, die wiederum die verschiedenen REST-Schnittstellen aufruft, um die CRUD-Operationen einmal auszuführen. CRUD steht für: Create, Read, Update, Delete. Die Daten werden dabei über eine Demo-Anwendung, welche in der Klasse `Anwendung.java` enthalten ist, in einem Terminal-Fenster auf dem Bildschirm ausgegeben.

Der Fluss der Daten wird verwaltet von Controllern. Die Anwendung bietet sowohl Spring-MCV-Controller, zu erkennen an der Annotation `@Controller`, als auch RESTful-Web-Service-Controller, welche zu erkennen sind an der Annotation `@RestController`. Die beiden Controller unterscheiden sich unter anderem in der Art und Weise wie der HTTP-Response-Body erstellt wird. Ein Spring-MVC-Controller kann Views zurückgeben, ein RESTful-Web-Service-Controller gibt ein Objekt zurück und Daten werden im JSON-Format direkt in die HTTP-Antwort geschrieben. Durch das Anbieten diverser Schnittstellen kann die hier beschriebene Anwendung in ebenso diverse Systeme eingebunden und von diversen Systemen aufgerufen werden. Controller werden im Package `controller` abgelegt.

Abbildung der CRUD-Operationen

Um zum Beispiel ein neues Produkt-Objekt in der im Abschnitt Architektur erwähnten Datenbank zu speichern, wird ein POST-Request an die Schnittstelle `/produkt` gesendet. Die Objekteigenschaften werden dem HTTP-Request-Body entnommen. Der Controller ruft anschließend einen Service auf – Services werden abgelegt im Package `service` – und dieser Service wiederum führt eine Methode aus einem Repository aus. Repositories werden abgelegt im Package `repository`, wobei jedes Repository ein Interface ist, welches das `JpaRepository` implementieren. Service und Repository werden verknüpft über die Annotation `@Autowired`.

Persistenz der Datenbank

Die Anwendung speichert Daten in einer H2-Datenbank. In der Standardkonfiguration ist diese nicht persistent, sondern schreibt Daten in den flüchtigen Arbeitsspeicher, sodass diese nach einem erneuten Starten der Anwendung nicht wieder verfügbar sind. Einstellungen an der Datenbank werden in der Datei application.properties vorgenommen, welche im Ordner resources abgelegt ist. In der vorliegenden Konfiguration wird eine Persistenz hergestellt über die Einstellung:

```
spring.datasource.url=jdbc:h2:./src/main/resources/datenbank;
```

Damit erfolgt die Anweisung, im aktuellen Projektverzeichnis im Ordner `[...]/resources` eine Datei `datenbank.mv.db` anzulegen. Diese Datei erscheint, sobald die Ausführung der Anwendung das erste Mal gestoppt wurde. Durch die weiteren Einstellungen in der Konfigurationsdatei wird sich bei jedem weiteren Start der Anwendung mit dieser Datei verbunden und diese aktualisiert.

Anwendungsdetails

Die Klasse `Anwendung.java` enthält eine Demo-Anwendung, welche Objekte vom Typ Produkt, Lebensmittelprodukt und Elektronik mittels REST-Schnittstelle in einer H2-Datenbank erstellt (create), ausliest (read), aktualisiert (update) und löscht (delete). In den folgenden Abschnitten wird beschrieben, wie die Schnittstellen zu bedienen sind, um selbst CRUD-Operationen zu veranlassen.

Voraussetzung für das Bedienen der Schnittstellen ist, dass die Anwendung wie im Abschnitt Die Anwendung das erste Mal starten beschrieben gestartet wird. Beim Starten der Anwendung wird ein lokaler Tomcat-Server initialisiert, welcher auf Port 8080 lauscht. Um zu verifizieren, dass die Anwendung ordnungsgemäß gestartet wurde, wird zunächst http://localhost:8080/ in einem Webbrowser aufgerufen. Daraufhin erscheint eine Weboberfläche, welche unter anderem diese Dokumentation enthält sowie Links zum Ausgeben von Tabellen auf dem Bildschirm.

Anschließend können die Schnittstellen nun wie in den folgenden Abschnitten bedient werden.

Hinweis: Der Übersicht wegen werden die praktischen Beispiele demonstriert mit der Klasse Produkt und einer Kommandozeileneingabe in Windows PowerShell.

Ein Produkt anlegen (CREATE)

Ein generisches Produkt wird angelegt mittels `POST`-Request an die Schnittstelle `/produkt`. Elektronik- und Lebensmittelprodukte werden mittels POST-Request an die Schnittstelle `/elektronik` bzw. `/lebensmittel` gespeichert.

Der Übersicht und Bedienfreundlichkeit wegen wird empfohlen, für das Versenden von HTTP-Anfragen das Programm "Postman" (externer Link: https://www.postman.com/downloads/) zu verwenden. Alternativ kann eine bereits vorinstallierte Kommandozeileneingabe wie Linux Shell oder Windows PowerShell verwendet werden.

Es wird angenommen, dass die Anwendung in den meisten Fällen auf Windows-Systemen ausgeführt wird, deswegen werden im Folgenden Windows-Befehle demonstriert. Linux-Benutzer verwenden `cURL`.

Beim Aufbau der HTTP-Anfragen muss das JSON-Format eingehalten werden, der Request-Body besteht demnach unter anderem aus `"attributname":"attributwert"`. Die Attributnamen sind den Klassendiagrammen aus dem Abschnitt Klassendiagramm nach UML-Standard zu entnehmen.

Die folgende Windows-PowerShell-Eingabe kann markiert, kopiert und mit STRG + SHIFT + V in die Windows PowerShell eingefügt werden.

```
Invoke-WebRequest -UseBasicParsing http://localhost:8080/produkt
-ContentType "application/json" -Method POST -Body
'{"bezeichnung":"erstes Produkt", "preis":"123.456",
"gewicht":"789.101", "hersteller":"Dokumentation",
"fertigungsTag":"2021-05-29", "lagerungsTemperatur":"20.21"}'
```

Werte eines Produkts auslesen (READ)

Alle Produkte werden ausgegeben mittels GET-Request an die Schnittstelle `/produkt`. Alle Lebensmittelprodukte werden ausgegeben mittels GET-Request an die Schnittstelle `/lebensmittel`. Alle Elektronikprodukte werden ausgegeben mittels GET-Request an die Schnittstelle `/elektronik`.

Der Übersicht wegen wird im Folgenden die Funktionsweise am Beispiel der Schnittstelle `/produkt` demonstriert, die übrigen Schnittstellen funktionieren analog dazu.

Die folgende Windows-PowerShell-Eingabe kann markiert, kopiert und mit STRG + SHIFT + V in die Windows PowerShell eingefügt werden.

```
Invoke-WebRequest -UseBasicParsing http://localhost:8080/produkt
```

Alternativ kann die Weboberfläche für das Abrufen der Daten verwendet werden:
http://localhost:8080/produkt.html

Nach einer Produkt-ID suchen

Die folgende Windows-PowerShell-Eingabe kann markiert, kopiert und mit STRG + SHIFT + V in die Windows PowerShell eingefügt werden.

```
Invoke-WebRequest -UseBasicParsing
http://localhost:8080/produkt/4
```

Alternativ kann die Weboberfläche für das Abrufen der Daten, die eine bestimmte ID enthalten, verwendet werden: http://localhost:8080/produkt/4

Nach einer Produkt-Bezeichnung suchen

Die folgende Windows-PowerShell-Eingabe kann markiert, kopiert und mit STRG + SHIFT + V in die Windows PowerShell eingefügt werden.

```
Invoke-WebRequest -UseBasicParsing
http://localhost:8080/produkt?bezeichnung=erstes%20Produkt
```

Alternativ kann die Weboberfläche verwendet werden, um die Produkttabelle nach einem Produkt mit einer bestimmten Bezeichnung zu durchsuchen:
http://localhost:8080/produkt?bezeichnung=erstes%20Produkt

Dabei ist der URL der Parameter `?bezeichnung=` und ein zu suchender Wert anzufügen.

Einen bestehenden Produkteintrag aktualisieren (UPDATE)

Um einen bestehenden Produkt-Eintrag zu aktualisieren, wird ein POST-Request an die Schnittstelle `/produkt` geschickt. Wichtig: Damit ein bestehender Eintrag aktualisiert wird, muss der Request-Body die ID des zu aktualisierenden Produkts enthalten.

Gegeben sei ein Produkt, welches angelegt wurde nach Anleitung im Abschnitt Ein Produkt anlegen. Die ID sei 821. Dann wird der Eintrag mit folgendem Request aktualisiert:

Die folgende Windows-PowerShell-Eingabe kann markiert, kopiert und mit STRG + SHIFT + V in die Windows PowerShell eingefügt werden.

```
Invoke-WebRequest -UseBasicParsing http://localhost:8080/produkt
-ContentType "application/json" -Method POST -Body '{"id":"821",
"bezeichnung":"aktualisiertes Produkt", "preis":"111.222",
"gewicht":"333.444", "hersteller":"Dokumentation",
"fertigungsTag":"2021-06-02", "lagerungsTemperatur":"20.21" }'
```

Um das aktualisierte Produkt auszugeben, kann die folgende URL aufgerufen werden:

http://localhost:8080/produkt?bezeichnung=aktualisiertes%20Produkt

Einen Produkteintrag löschen (DELETE)

Das Löschen eines Produktes, Elektronik- oder Lebensmittelproduktes erfolgt über die ID. Diese muss demnach bekannt sein. Dann kann ein Eintrag gelöscht werden mittels DELETE-Request an die Schnittstelle `/produkt/{id}`, `/lebensmittel/{id}` bzw. `/elektronik{{id}`. Die Schnittstellen funktionieren nach jeweils demselben Prinzip, der Übersicht wegen wird exemplarisch die Schnittstelle `/produkt/{id}` demonstriert.

Gegeben sei ein Produkt, welches angelegt wurde nach Anleitung im Abschnitt Ein Produkt anlegen. Die ID sei 821. Dann wird der Eintrag mit folgendem Request gelöscht:

Die folgende Windows-PowerShell-Eingabe kann markiert, kopiert und mit STRG + SHIFT + V in die Windows PowerShell eingefügt werden.

```
Invoke-WebRequest -UseBasicParsing
http://localhost:8080/produkt/821 -Method DELETE
```

Ein erneuter Aufruf von http://localhost:8080/produkt?bezeichnung=aktualisiertes%20Produkt zeigt, dass kein Produkt mit dieser ID mehr vorhanden ist.