DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Software-Engineering II

Programmentwurf
TINF20B1
5.+6. Semester (2022/2023)

Thema:
Kostenrechner für Fahrgemeinschaften

Dozent:
Daniel Lindner

Bearbeitender:
Yannik Schiebelhut

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abbildungsverzeichnis

1 Beschreibung des Programms

Im Rahmen eines dualen Studiums an der DHBW bietet es sich an, für die Praxisphasen mit anderen Studenten Fahrgemeinschaften zum Betrieb zu bilden. Diese Fahrgemeinschaften haben zwar einen festen Rahmen an Mitgliedern, jedoch haben diese alle verschiedene und wechselnde Termine, wodurch sich der Anwendungsfall für ein Programm ergeben hat, welches den Fahrer einer Fahrgemeinschaft dabei unterstützt, mit wenig Aufwand eine für alle Beteiligten faire Abrechnung zu erstellen.

1.1 Funktionalität

Das Programm richtet sich an den Besitzer eines Autos, weiterhin als Fahrer bezeichnet. Der Fahrer kann mehrere „Fahrgemeinschaften“, also Gruppen von Personen, definieren. Dabei kann eine Person in mehreren Fahrgemeinschaften sein und eine Fahrgemeinschaft hat in der Regel mehrere Mitfahrer.

Innerhalb einer Fahrgemeinschaft werden Fahrperioden angelegt. Eine Fahrperiode bezeichnet dabei die Menge aller Fahrten zwischen zwei Tankstopps. Innerhalb einer Fahrperiode haben alle Fahrten dieselbe Strecke, denselben Spritverbrauch und denselben Spritpreis. Des Weiteren kann ein Fixbetrag definiert werden, um etwa Verschleißkosten des Fahrzeugs auf die Mitfahrer umzulegen. In einer Fahrperiode wiederum kann eine beliebige Menge an Fahrten angelegt werden. Für jede dieser Fahrten wird ausgewählt, welche Mitglieder der Fahrgemeinschaft im Fahrzeug saßen. Wenn wieder getankt wird, wird die Fahrperiode im System abgeschlossen. Dabei wird der Anteil an den entstandenen Fahrkosten für jeden Mitfahrer innerhalb der Fahrperiode ermittelt. Für diesen Anteil wird ein PayPal-Link generiert und der jeweiligen Person über Telegram zugesandt, um auch den Bezahlvorgang angenehm zu gestalten.

Eine Person wiederum hat einen Namen, eine Adresse und eine Telegram-Chat-ID, über die diese Person zu kontaktieren ist.

1.2 Technologien

Umgesetzt ist das Programm in Java, wobei Maven als Build-System verwendet wird. Die Datenhaltung erfolgt lokal im JSON-Format mittels der externen GSON-Bibliothek. Das Programm verfügt über eine grafische Nutzeroberfläche, welche mit Java Swing erstellt ist. Weiterhin wird auf die Telegram API zugegriffen. Hierfür ist allerdings im Rahmen einer einfachen Proof of Concept Implementierung keine externe Bibliothek vonnöten.

Der Quellcode wird in einem Git-Repository verwaltet, welches auf GitHub unter https://github.com/yschiebelhut/carpool-java zu finden ist.

2 Domain Driven Design

2.1 Analyse der Ubiquitous Language

Das Programm wird im Rahmen einer Fahrgemeinschaft einer deutschsprachigen Gruppe von Studenten erstellt, weshalb Deutsch als Sprache für die Ubiquitous Language gewählt wird.

Die Beschreibung in Abschnitt 1.1 entspricht dem Sprachgebrauch eines Domänenexperten, weshalb sich diese Wortwahl auch im Quellcode widerspiegeln sollte. Die wichtigsten Begriffe sind hierbei:

- Fahrgemeinschaft
- Fahrperiode
- Distanz (bestehend aus Betrag und Streckeneinheit)
- Fahrt
- Person (je nach Kontext auch als Mitfahrer bezeichnet, diese Begriffe sind austauschbar)
- Adresse (bestehend aus Straße und Ort)
- Telegram-Chat-ID
- PayPal-Link

Weiterhin wird festgelegt, dass für Informatikstudenten einzelne englische Begriffe keine Sprachbarriere darstellen. Deshalb werden Getter- und Setter-Methoden weiterhin mit englischen Vorsilben konstruiert, um den Quellcode nah an Programmiersprachweiten Standards zu halten.

Aus der Funktionsbeschreibung des Programms und der Analyse der Ubiquitous Language ergibt sich das in Abbildung 2.1 dargestellte Domänenmodell.

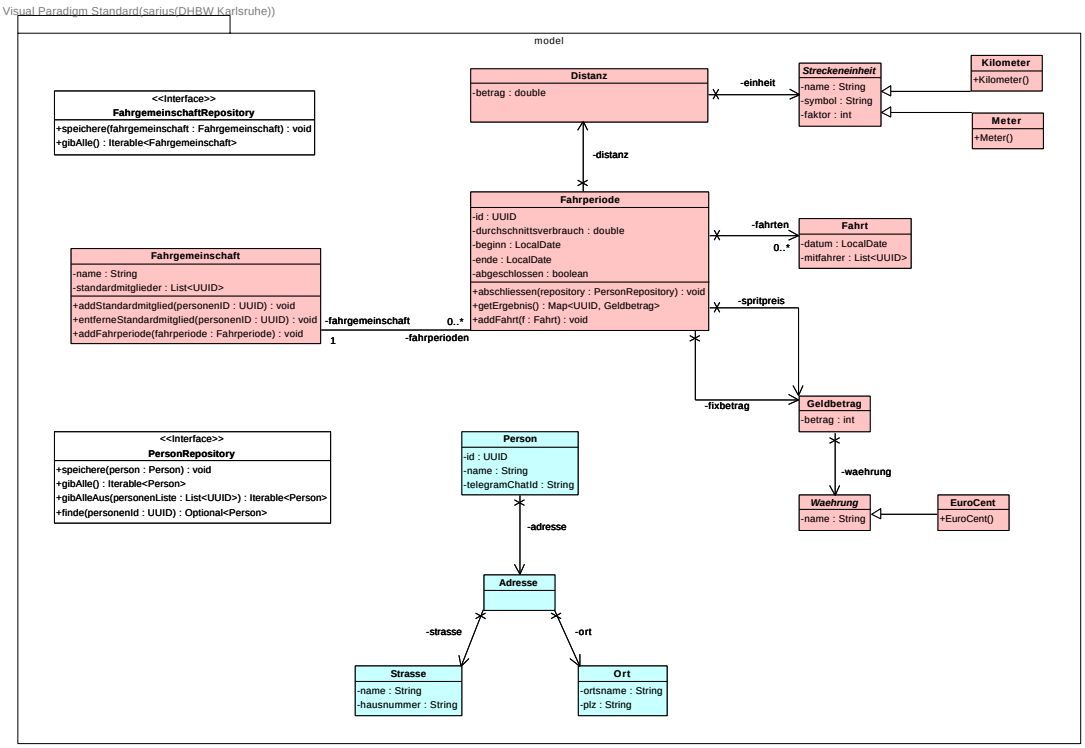

Abbildung 2.1: Domänenmodell

2.2 Entities

Im dargestellten Domänenmodell (Abbildung 2.1) finden sich folgende Entitäten:

- Person
- Fahrgemeinschaft
- Fahrt

Eine Person besitzt die Eigenschaften Name, Adresse und Telegram-Chat-ID, welche sich alle ändern können, ohne dass sich die Identität der Person ändert. Einer Fahrgemeinschaft werden im Laufe der Zeit Fahrperioden und möglicherweise Mitglieder hinzugefügt, es ist jedoch weiterhin dieselbe Fahrgemeinschaft. Ähnlich sieht es bei einer Fahrperiode aus, der nach und nach Fahrten hinzugefügt werden. Hieran lässt sich ablesen, dass alle diese Klassen einen Lebenszyklus besitzen und somit als Entitäten einzuordnen sind.

Jede Entität verfügt über eine Identität. Im Falle der Fahrgemeinschaft wird hierfür der Name der Fahrgemeinschaft als natürlicher Identifier gewählt, da ein Fahrer realistischerweise kaum mehr als eine Handvoll verschiedene Gruppen mitnimmt. Für Person und Fahrperiode reichen die Attribute nicht aus, um eine Entität eindeutig zu identifizieren. Deshalb wird hier eine UUID als künstlicher Schlüssel vergeben.

2.3 Value Objects

Die restlichen, in Abbildung 2.1 dargestellten, Klassen werden als Value Objects realisiert. Dies sind:

- Distanz
- Unterklassen von Streckeneinheit
- Fahrt
- Geldbetrag
- Unterklassen von Waehrung
- Adresse

- Strasse
- Ort

Alle diese Klassen besitzen keinen Lebenszyklus und definieren sich rein über ihre Werte, weshalb sich auch keine Identität besitzen und sich somit nicht als Entität einordnen lassen.

2.4 Aggregates

Eine Adresse stellt eine Eigenschaft einer Person dar. Weiterhin setzt sich eine Adresse aus einer Strasse und einem Ort zusammen. Adresse, Strasse und Ort existieren nur im Kontext einer Person. Wird die Person gelöscht, so existiert auch die zugehörige Adresse nicht mehr. Deshalb werden Person, Adresse, Strasse und Ort zu einem Aggregat zusammengefasst. Person ist in diesem Aggregat die einzige Entität und ist deshalb automatisch die Root-Entität. Im Domänenmodell wird dieses Aggregat türkis hervorgehoben.

Im Falle von Fahrperioden stellen Distanz, Streckeneinheit, Fahrt, Geldbetrag und Waehrung Value Objects dar, die Eigenschaften von Fahrperiode zu klassifizieren sind. Deshalb werden diese Klassen zu einem Aggregat zusammengefasst. Weiterhin existieren Fahrperioden jedoch nur im Kontext ihrer Fahrgemeinschaft. Wird die Fahrgemeinschaft gelöscht, so entfernt dies auch die Fahrperioden. Deshalb wird diesem Aggregat auch noch die Fahrgemeinschaft hinzugefügt und diese außerdem als Root-Entität gewählt. Im Domänenmodell wird dieses Aggregat rot hervorgehoben.

2.5 Reposiotries

Gemäß dem Grundsatz „Ein Repository pro Aggregat“ werden zwei Repositories für den Zugriff auf den persistenten Speicher definiert. Dies sind PersonRepository und FahrgemeinschaftRepository. Diese erlauben jeweils den Zugriff auf die jeweilige Root-Entität des Repositories - also Person respektive Fahrgemeinschaft.