Software-Praktikum

SS 2021, Blatt 1: UML Diagramme

Ausgabe: 30.08.2021

Prof. Dr. Sven Apel

Sie haben heute in der Vorlesung die Modellierungssprache UML (Unified Modelling Language) kennengelernt, in der Sie später in der Entwurfsphase unter anderem Klassen- und Sequenzdiagramme für Ihren Programmentwurf anfertigen werden.

Wir möchten in diesem Übungsblatt (und auch später im Tutorium) das Erstellen und die Notation dieser Diagramme üben.

Aufgabe 1 UML-basierte Modellierung

Laut einer Umfrage an der Universität des Saarlandes ist Pizza für 81.9% der Studenten eines der wichtigsten Grundnahrungsmittel. Zusammen mit ein paar BWL-Studenten entschließen Sie sich dazu ein Start-Up gründen, das die Studenten an der Uni jederzeit mit Pizza versorgen soll. Dazu soll es eine Bestell-App namens *PadsU* geben, sowie eine separate App für die Lieferanten, die Bestellungen ausliefern.

Der Anforderungskatalog lautet wie folgt:

- In einem Kundenkonto wird bei der Registrierung folgendes hinterlegt:
  Mailadresse, Passwort, Telefonnummer und Rechnungsadresse bestehend aus Vor- und Nachname, Straße, Hausnummer, PLZ und Ort. Die Rechnungsadresse kann bei einer Bestellung als Lieferadresse übernommen werden.
- Nach erfolgreichem Login kann man seine Daten bearbeiten, seine Bestellhistorie ansehen und eine neue Bestellung aufgeben.
- Um eine Bestellung aufzugeben, muss man ein Restaurant auswählen, Gerichte zum Warenkorb hinzufügen und die Bestellung vervollständigen, indem man optional eine von der Rechnungsadresse abweichende Lieferadresse, sowie eine Bezahlmethode (CASH oder ONLINE) angibt.
- Bei Online-Bezahlung sollte die Bestellung erst nach erfolgreicher Zahlung an das Restaurant weitergeleitet werden.
- Ein Restaurant muss den Erhalt einer Bestellungen bestätigen, woraufhin erst die Zubereitung und Auslieferung folgt.
- Die Konten der Restaurants sind auf einem Server gespeichert, was aber in der Modellierung dieser Aufgabe nicht berücksichtigt werden muss. Als Restaurant kann man sich einloggen, sein Menü bearbeiten, indem man Gerichte hinzufügt oder entfernt, sowie die Öffnungstage (MON-SUN) hinzufügen bzw. entfernen.
- Für die Lieferanten bzw. Fahrer von PadsU soll es eine extra App geben. Nach erfolgreicher Bewerbung wird dem Fahrer ein Lieferantenaccount zugeteilt. In seinem Account kann der Fahrer Aufträge annehmen, seinen Schichtplan erstellen, ändern (bzw. Änderungen bestätigen) und den Plan abgeben, sowie Bestellungen als ausgeliefert markieren. PadsU kann die Lieferantenaccounts verwalten, die Schichtpläne bestätigen oder ändern und den Lohn für getätigte Auslieferungen auszahlen.

a) Ergänzen Sie das UML-Klassendiagramm für den Bestell-App Teil, dh. überlegen Sie sich sinnvolle Klassen und zeichnen Sie ebenso die Assoziationen (inkl. Multiplizitäten) ein.

b) Erstellen Sie darauf aufbauend die Sequenzdiagramme für folgende Szenarien:

   (i) einen Nutzer-Account mit den folgenden Daten erstellen und einloggen:

   Mailadresse: mm@uds.de
   Passwort: weakpassword
   Telefonnummer: 01234

   Die Adresse kann vereinfacht als `billingAddress` mitgegeben werden. App-interne Authentifizierungsmethoden oder dergleichen können vernachlässigt werden.

(ii) eine Bestellung aufgeben:

Dazu fügt der Nutzer das Gericht *cheesecake* vom Restaurant *D'Apel* zum Warenkorb hinzu.
Das Auswählen der Bezahlungsmethode und der Lieferadresse können vereinfacht als GUI Methoden dargestellt werden (Bsp. `select payment method` vom Nutzer ausgehend). Denken Sie daran, dass das Restaurant die Bestellung bestätigen muss.

Tipp:

Versuchen Sie beide Bezahlungsmethoden im Diagramm unter Verwendung eines "optional"-Fragments darzustellen.

c) Ergänzen Sie die beiden Use-Case Diagramme für die PadsU App.

Anwendungsfälle *(Use-Cases)* beschreiben - grob gesagt - was die verschiedenen Akteure tun können/wie sie mit einem Programm interagieren können. Use-Case Diagramme sind also eine sehr simple Methode, um den Umfang der Funktionen eines Programms zu visualisieren.

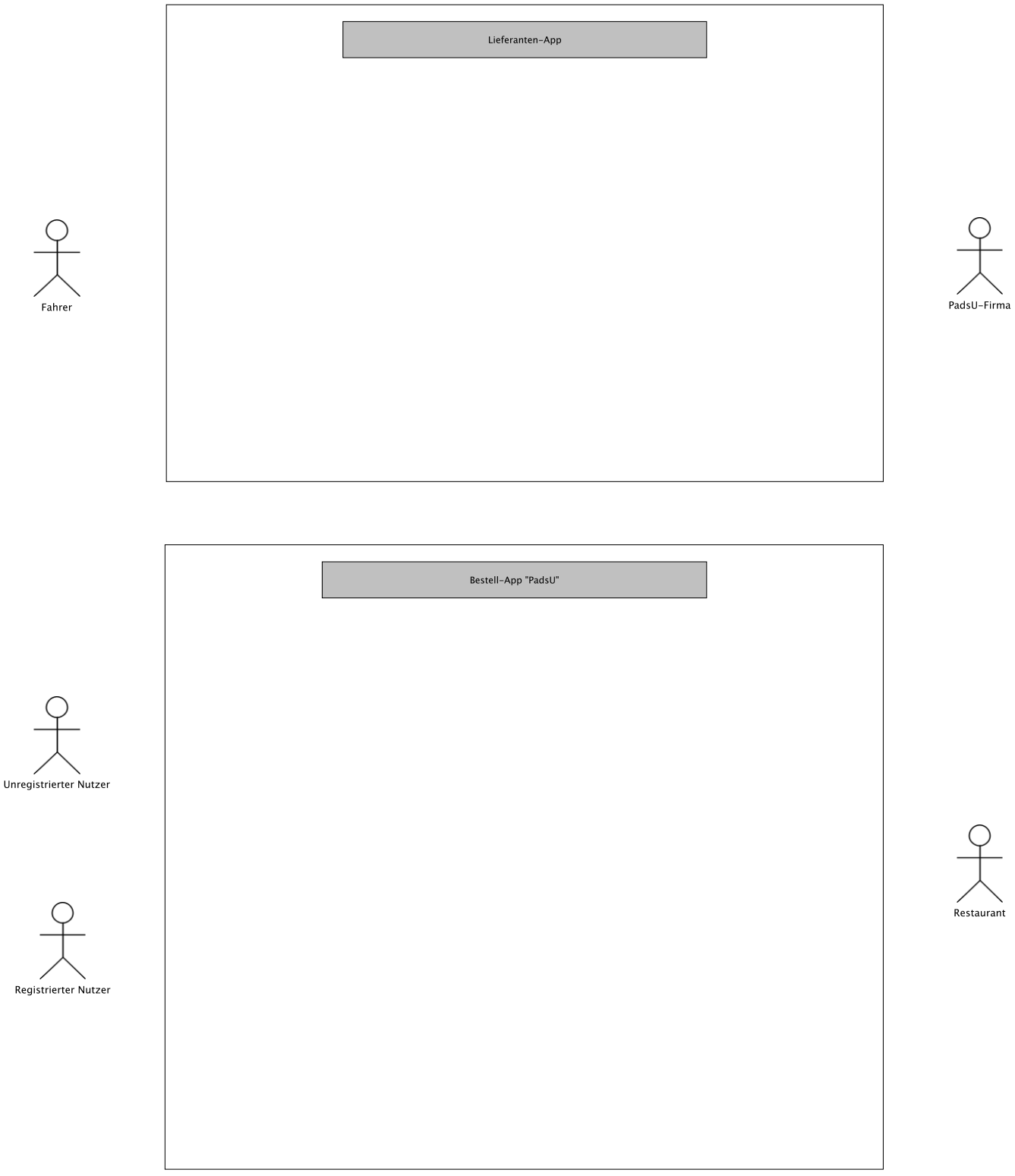
Weiterführende Infos zu Use-Case Diagrammen, include/extend Beziehungen etc. :

`https://www.microtool.de/wissen-online/was-ist-ein-use-case-diagramm/`

Zu a):

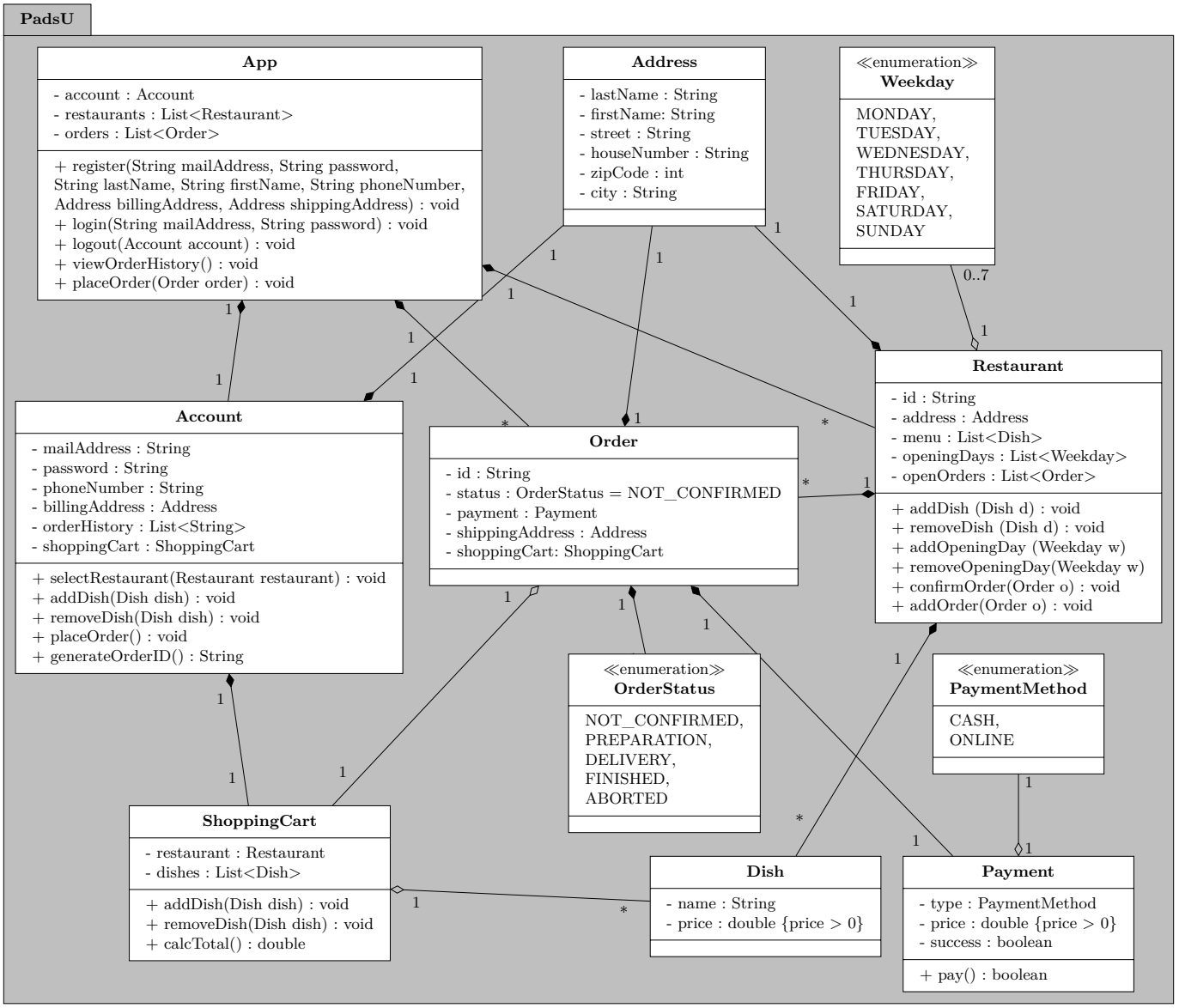

PadsU
App
- account : Account
- restaurants : List<Restaurant>
- orders : List<Order>
+ register(String mailAddress, String password,
String lastName, String firstName, String phoneNumber,
Address billingAddress, Address shippingAddress) : void
+ login(String mailAddress, String password) : void
+ logout(Account account) : void
+ viewOrderHistory() : void
+ placeOrder(Order order) : void

Zu c):


Lieferanten-App
Fahrer
PadsU-Firma
Bestell-App "PadsU"
Unregistrierter Nutzer
Registrierter Nutzer
Restaurant

Lösung

a)



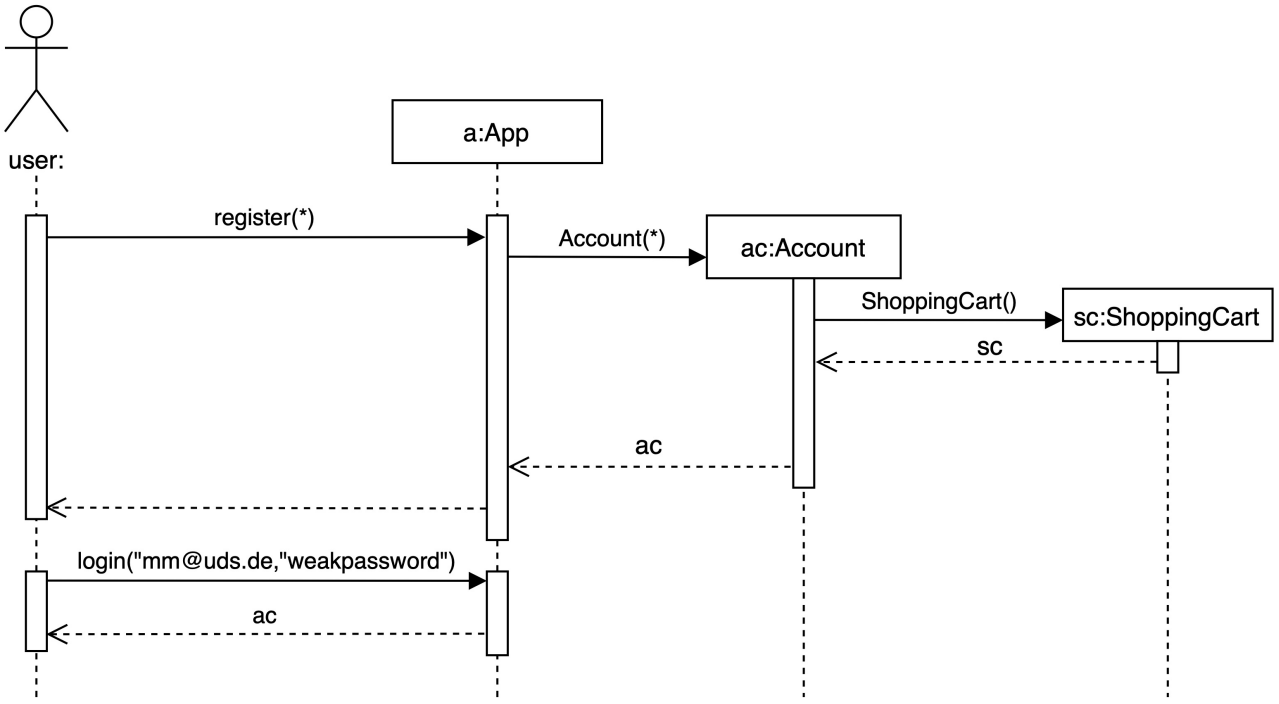
Wichtige Hinweise zum Erstellen eines Klassendiagramms:

- *Funktionalität*: Eine einzig große Klasse, die alles kann, will man vermeiden. Daher sollte verschiedene Funktionalität in verschiedene Klassen sinnvoll ausgegliedert werden.
- *Beziehungen zwischen Klassen*: Eine Klasse, die ein oder mehrere Objekte einer anderen Klasse besitzt, hat eine Assoziation zu dieser. Dabei sollte auf die Multiplizitäten geachtet werden.

  Wenn die Existenz eines Objektes von der anderen Klasse abhängig ist, dann ist es eine *Komposition*. Wenn ein Objekt auch alleine existieren kann, ist es eine *Aggregation*.

  Beispiel: Das `Address`-Objekt *shippingAddress* in der `Order`-Klasse kann nur existieren, wenn es die zugehörige Order gibt. Anders formuliert: Wenn die Order nicht existiert, existiert auch nicht die zugehörige *shippingAddress*. Eine Order hat genau eine *shippingAddress*, und somit genau ein `Address`-Objekt (= Multiplizitäten).
- *Sichtbarkeiten*: Allgemein gilt zu Sichtbarkeiten: So viel Sichtbarkeit wie nötig und so wenig Sichtbarkeit wie möglich. Typischerweise sind Attribute `private` (Minus-Zeichen vor den Attributen). Deren Werte werden über sogenannte Getter- und Setter-Methoden ausgelesen bzw. gesetzt, die allgemein aufgrund ihrer Trivialität nicht im Klassendiagramm aufgenommen werden müssen. Methoden sind, sofern sie öffentlich verfügbar sein sollten `public`. Jegliche Hilfsmethoden sind typischerweise ebenfalls `private`.
- *Initialwerte und Zusicherung*: Gelten für primitive Datentypen gewisse Bedingungen oder liegt ein Standardwert vor, so muss dies im Klassendiagramm gekennzeichnet sein.

b) (i)

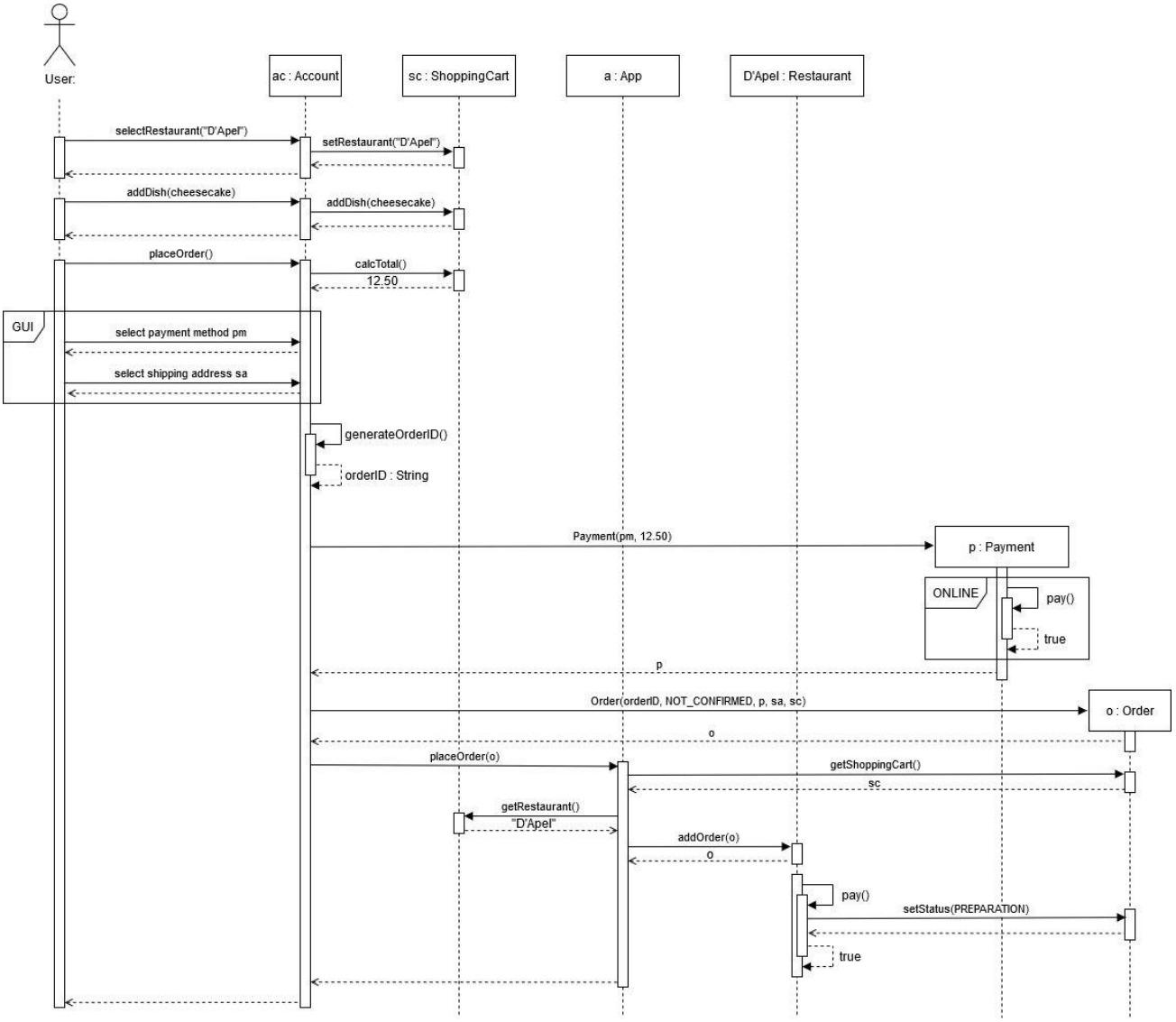


* = ”mm@uds.de”, ”weakpassword”,”01234”, `billingAddress`

Wichtige Hinweise zur Erstellung von Sequenzdiagrammen:

- *Akteur*: Der Akteur ist typischerweise der Nutzer, der durch eine Eingabe von außerhalb ein bestimmtes Szenario auslöst.
- *Klassen*: Bei den Klassen ist es wichtig, jederzeit die Benennung des Objektes (z.B. `ac`) und den Klassennamen (z.B. `Account`) zu annotieren.
- *Erstellung*: Objekte, die erst im Verlauf des Szenarios erstellt werden, müssen zuerst über einen Konstruktor-Aufruf im Sequenzdiagramm erstellt werden (z.B. das Objekt `ac` in Diagramm (i)). Für bereits existente Objekte gilt dies nicht.
- *Lebenslinie*: Die Lebenslinie (gestrichelte Linie) gibt an, wie lange dieses Objekt existiert.
- *Methodenrumpf*: Der Methodenrumpf (dicker Balken) gibt an, dass auf dem jeweiligen Objekt die entsprechende Methode aktiv ist. Wenn innerhalb eines Methodenaufrufs eine zweite Methode auf der gleichen Instanz des Objektes aufgerufen wird, wird ein weiterer Balken für den zweiten Aufruf eingezeichnet (Bsp: Sequenzdiagramm (ii) Aufruf von `generateOrderID()`).
- *Entfernen von Objekten*: Sollte ein Objekt in einem Szenario nicht mehr gebraucht und dadurch entfernt werden, sollte dies entsprechend durch eine endende Lebenslinie und einem `X` gekennzeichnet werden.
- *Methodenaufrufe*: Bei Methodenaufrufen sollen die übergebenen Parameter konkret angegeben werden (z.B. der Aufruf der Methode `login` in obigem Lösungsvorschlag). Unsere Aufgabenstellungen beschreiben konkrete Szenarien, d.h. sie müssen keine Alternativszenarien bei der Erstellung Ihrer Sequenzdiagramme berücksichtigen.

(ii)


User:
ac : Account
sc : ShoppingCart
a : App
D'Apel : Restaurant
selectRestaurant("D'Apel")
setRestaurant("D'Apel")
addDish(cheesecake)
addDish(cheesecake)
placeOrder()
calcTotal()
12.50
GUI
select payment method pm
select shipping address sa
generateOrderID()
orderID : String
Payment(pm, 12.50)
p : Payment
ONLINE
pay()
true
p
Order(orderID, NOT_CONFIRMED, p, sa, sc)
o : Order
o
placeOrder(o)
getShoppingCart()
sc
getRestaurant()
"D'Apel"
addOrder(o)
o
pay()
setStatus(PREPARATION)
true

c)


Lieferanten-App
Schichtplan erstellen/ändern/abgeben
Schichtplan bestätigen/ändern
<<include>>
Login
Lieferantenaccounts verwalten
Auftrag annehmen
<<include>>
<<include>>
Lohn auszahlen
Fahrer
PadsU-Firma
Bestellung ausliefern
EP1: Barbezahlung
EP2: Onlinebezahlung
<<extend>>
<<extend>>
EP1: Bezahlung entgegennehmen
EP2: Bestellung abstellen
(kontaktlose Lieferung)
Bestell-App "PadsU"
Account erstellen
Unregistrierter Nutzer
<<include>>
Persönliche Daten erfassen/bearbeiten
Öffnungstage bearbeiten
<<include>>
<<include>>
Bestellhistorie ansehen
Menü bearbeiten
<<include>>
<<include>>
Login
Registrierter Nutzer
Restaurant
Logout
<<include>>
<<include>>
<<include>>
Bestellung bestätigen
Bestellung aufgeben
EP: Lieferadresse anpassen
<<include>>
Bezahlungsmethode angeben
EP: Online Bezahlung
<<include>>
<<include>>
<<extend>>
<<extend>>
Restaurant auswählen
Zu Warenkorb hinzufügen
EP: Lieferadresse ändern
bezahlen