VU Objektorientierte Modellierung	**Übung 3: Sequenzdiagramm**
188.391, SS2016	Übungsgruppen: 18.04.2016 - 21.04.2016

Aufgabe 1: Verhaltensmodellierung mittels Sequenzdiagramm

Wiederholen Sie das Kapitel aus der Vorlesung, das sich mit Sequenzdiagrammen beschäftigt.

- Welche vier Arten von Interaktionsdigrammen gibt es? Beschreiben Sie diese kurz. Wofür werden Interaktionsdiagramme eingesetzt?
- Wie ist ein Sequenzdiagramm prinzipiell aufgebaut? Welche Elemente kann es enthalten?
- Beschreiben Sie die Unterschiede zwischen synchronen und asynchronen Nachrichten.
- Was ist ein aktives Objekt, was ist ein passives Objekt? Wie unterscheiden sich diese?

Aufgabe 2: Verhaltensmodellierung mittels Sequenzdiagramm

Wiederholen Sie das Kapitel aus der Vorlesung, das sich mit Sequenzdiagrammen beschäftigt.

- Was ist eine Zustandsinvariante im Kontext des Sequenzdiagramms? Wie können Zeiteinschränkungen angegeben werden?
- Welche Arten von Verzweigungen und Schleifen können in Sequenzdiagrammen auftreten? Beschreiben Sie die entsprechenden Operatoren.
- Welche Operatoren stehen im Sequenzdiagramm zur Verfügung, um parallele Abläufe zu realisieren bzw. um Ordnungen im Ablauf festzulegen?
- Erklären Sie die kombinierten Fragmente aus der Gruppe “Filterungen und Zusicherungen”.

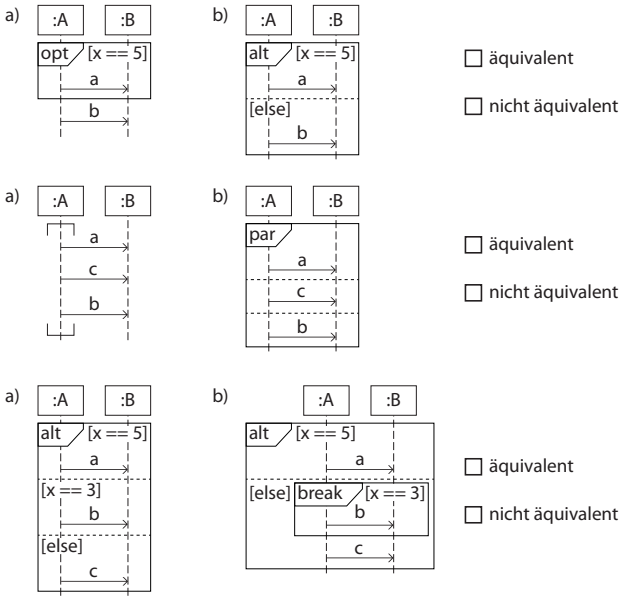
Aufgabe 3: Kommunikation im Sequenzdiagramm

Handelt es sich in den folgenden Kommunikationsszenarien um synchrone oder asynchrone Kommunikation? Identifizieren Sie die involvierten Interaktionspartner!

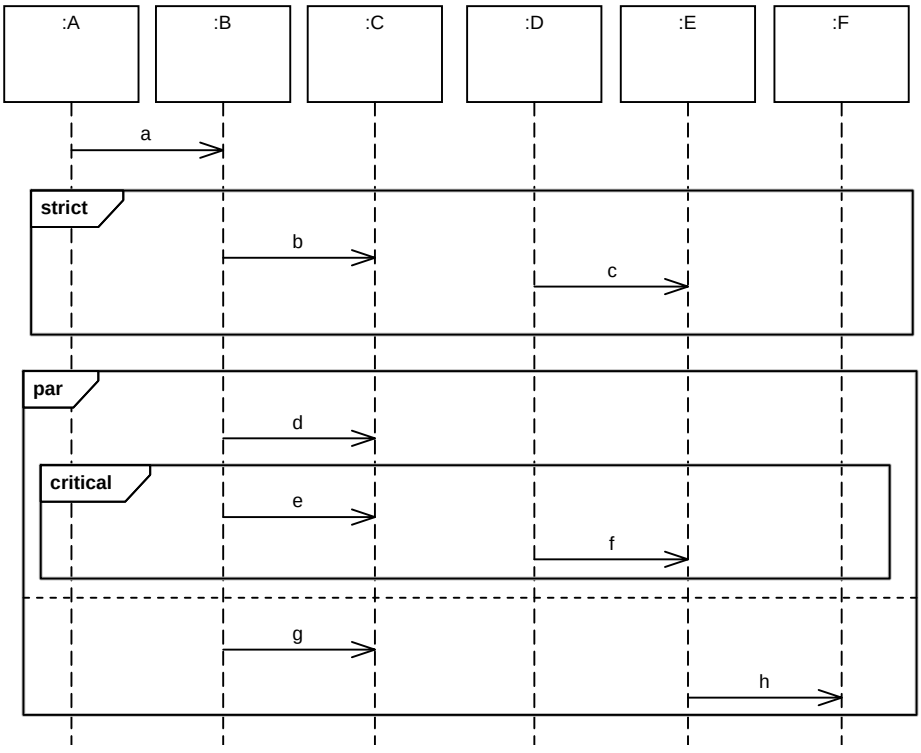
Eine Mitarbeiterin informiert einen Gast über die Zimmerpreise.	☐ synchron	☐ asynchron
Auf einem Infoscreen im Wellnessbereich werden die neuesten Massagen und Behandlungen eingeblendet.	☐ synchron	☐ asynchron
Eine Mitarbeiterin begleitet die Hotelgäste auf ihr Zimmer.	☐ synchron	☐ asynchron
Über das Zimmerradio ertönt beruhigende Entspannungsmusik.	☐ synchron	☐ asynchron
Der Gast schaltet sämtliche Lichtquellen des Hotelzimmers per Touch-Screen ein.	☐ synchron	☐ asynchron
Ein Gast nutzt den Computer in der Hotellobby, um einem Freund ein E-Mail zu senden.	☐ synchron	☐ asynchron
Ein Gast beschwert sich per Telefon bei der Rezeption, dass ein Bademantel fehlt.	☐ synchron	☐ asynchron
Ein Gast füllt einen Fragebogen zur Qualität des Hotels aus.	☐ synchron	☐ asynchron
Ein Ehepaar diskutiert, ob es zuerst die Nachmittagsjause oder den Fitnessraum in Anspruch nehmen soll.	☐ synchron	☐ asynchron
Vor der Abreise bezahlt ein Gast die Hotelrechnung mit Kreditkarte.	☐ synchron	☐ asynchron

Aufgabe 4: Kombinierte Fragmente

a) **Äquivalenzen**
Gegeben sind jeweils zwei Ausschnitte eines Sequenzdiagramms. Kreuzen Sie an, ob die beiden Ausschnitte jeweils „äquivalent“ oder „nicht äquivalent“ sind. Begründen Sie warum.

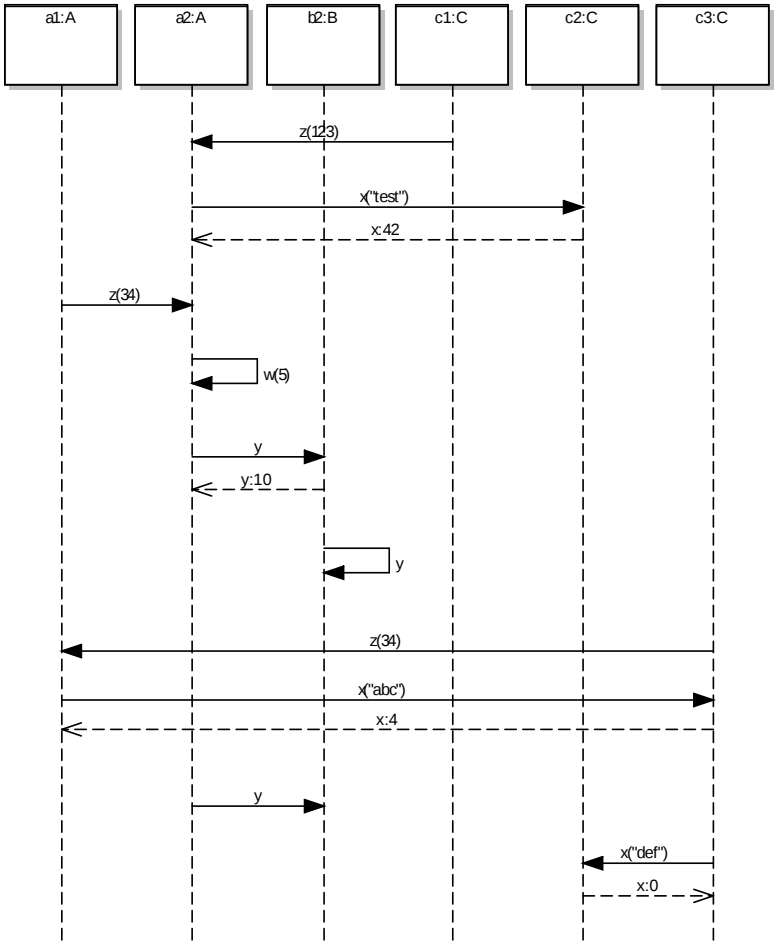


b) **Berechnung von Traces**
Beschreiben Sie alle möglichen Ereignisfolgen des folgenden Diagramms.



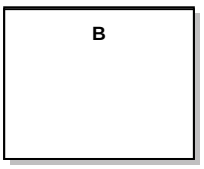
Aufgabe 5: Klassendiagramm aus Sequenzdiagramm

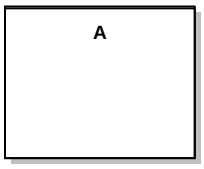
Gegeben ist folgendes Sequenzdiagramm:

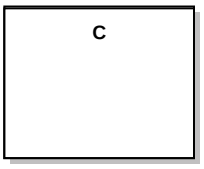


Vervollständigen Sie nachfolgendes Klassendiagramm.

- Operationsdefinitionen mit Typangaben, soweit ersichtlich
- Beziehungen zwischen Klassen in Form von navigierbaren Assoziationen: Zeichnen Sie nur Navigationsrichtungen ein, die aus dem gegebenen Sequenzdiagramm ersichtlich sind.







Aufgabe 6: Darstellung von Programmabläufen mittels Sequenzdiagramm

Stellen Sie die Abläufe von folgendem Programm mittels Sequenzdiagramm dar. Modellieren Sie auch allfällige Antwortnachrichten.
Sie können davon ausgehen, dass alle nicht explizit deklarierten Variablen bereits deklariert und initialisiert sind. Der Ausdruck „...“ markiert vernachlässigte Codeteile, die nicht modelliert werden müssen.

```
class Main {
        ...
        Worker w1 = new Worker();
        w1.run();

        print("We will buy the cheaper one!");

        q = w1.getQuantity();

        if(q < 1) {
                print("Invalid quantity");
                exit; // Programm wird beendet
        }

        Article a1 = new Article();
        pa1 = a1.getPrice(w1);

        Article a2 = new Article();
        pa2 = a2.getPrice(w1);

        if (pa1 < pa2) {
                for ( int i = 1; i <= q; i++ ) {
                        resNr = w1.reserve(a1);
                        res[i] = resNr;
                }
                else
                        ...
        }

        private void print(String m) {...}
}

class Worker extends Thread {
        public void run() { }

        public int getQuantity {
                return quantity;
        }

        public double convertToEuro (double amount) {
                double resultInEuro = amount*0.814929509;
                return resultInEuro;
        }

        public int reserve(Article art) {
                ...
                return nr;
        }
}
```

```
class Article {
        public double getPrice(Worker w1) {
                if (currency.equals("EURO")) {
                        return price;
                } else {
                        double priceConv = w1.convertToEuro(price);
                        return priceConv;
                }
        }
}
```