

Aufgabe 1: Verhaltensmodellierung mittels Sequenzdiagramm

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

- Welche 4 Arten von Interaktionsdiagrammen 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".

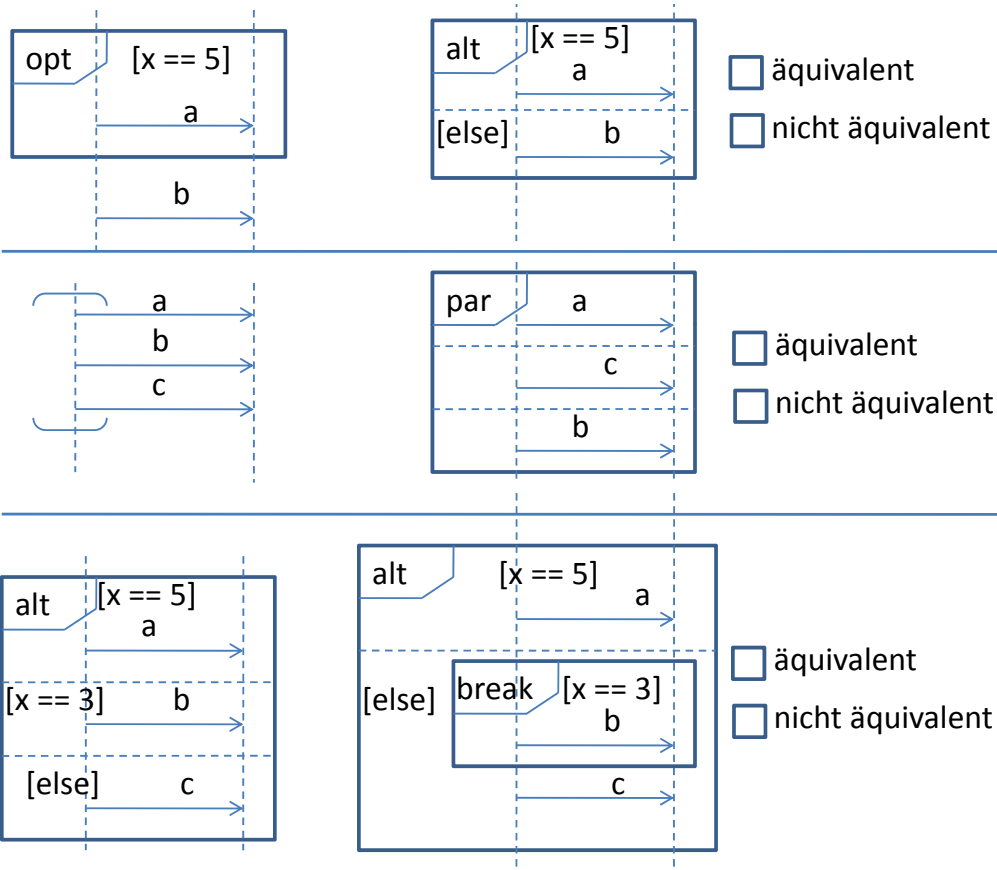
Aufgabe 3: Kommunikation im Sequenzdiagramm

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

In einem Unternehmen werden die Mitarbeiter über ein Megaphone gebeten das Gebäude zu verlassen.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
In dem Unternehmen ist es üblich, Sitzungen mit Partnern via Videokonferenz abzuhalten.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Wird einem Mitarbeiter gestattet von zu Hause zu arbeiten, kommuniziert dieser über ein Firmenhandy mit seinen Kollegen.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Ein Mitarbeiter verschickt seinen Urlaubsantrag per E-Mail an seinen Vorgesetzten.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Die Empfangsdame des Unternehmens beschreibt Frau Müller den Weg zur Personalabteilung.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Frau Müller informiert sich bei der Personalabteilung über ein offenes Stellenangebot.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Frau Müller schreibt ihre Eindrücke bezüglich des Stellenangebots in ein Forum.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Frau Müller verschickt ihre Bewerbungsunterlagen per E-Mail.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Die Personalabteilung gibt Frau Müller am Telefon einen Termin für das Vorstellungsgespräch bekannt.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Die Dame aus der Personalabteilung und der IT-Abteilungsleiter besprechen den Verlauf des Bewerbungsgesprächs.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron

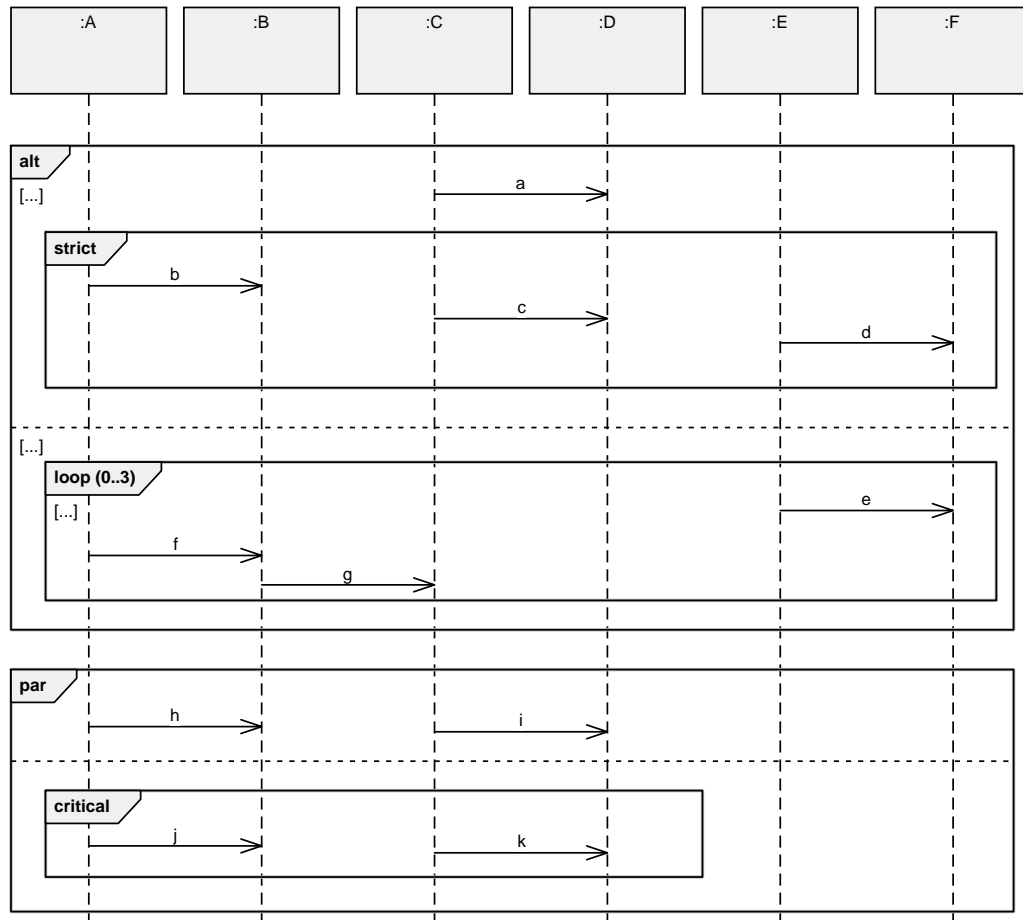
Aufgabe 4: Kombinierte Fragmente

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.



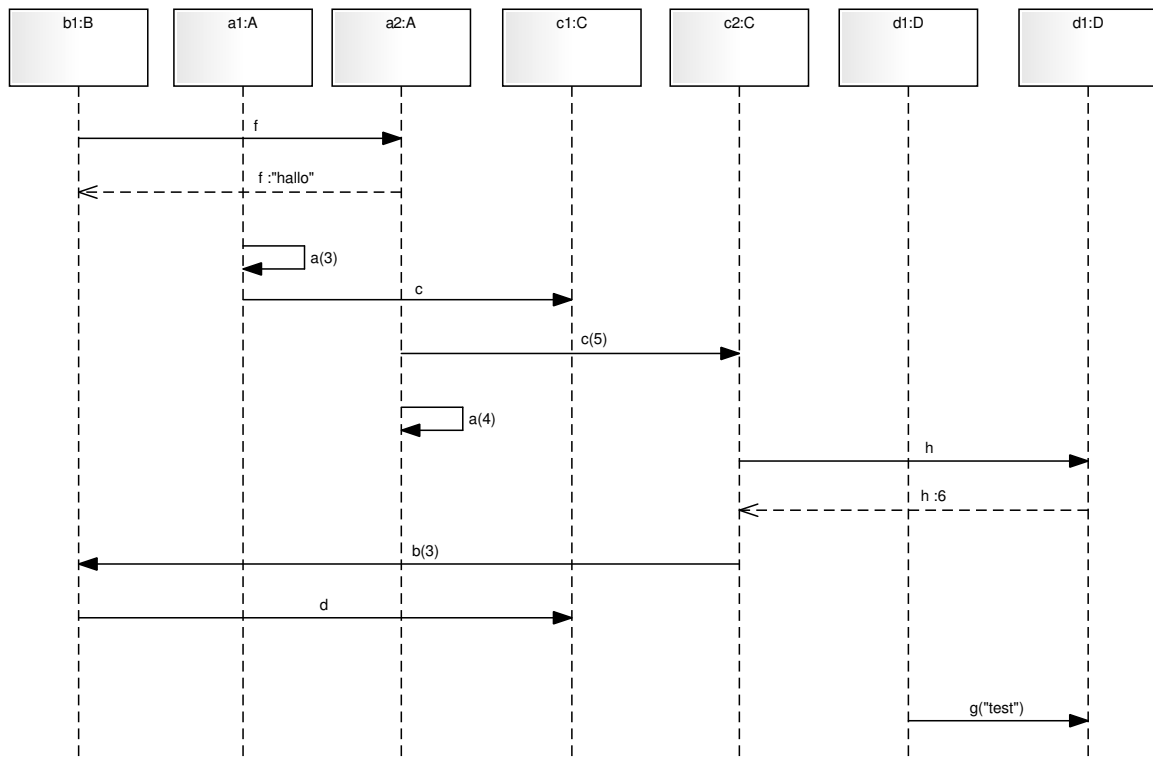
b) **Berechnung von Traces**

Beschreiben Sie alle möglichen Ereignisfolgen des folgenden Diagramms.



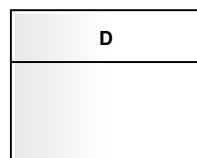
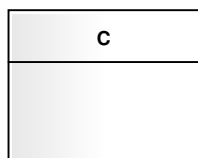
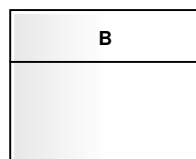
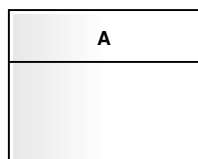
Aufgabe 5: Klassendiagramm aus Sequenzdiagramm

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.

```
class Main {
    ....
    Worker w = s1.getConnection(user, pw);

    if(w==null) {
        print("Error");
        exit;          // Programm wird beendet
    }

    do {
        m = w.getMail();
        print(m);
    } while (m != null);

    status = w.sendMail("abc", "test");
    ...
    private void print(String m) {
        ...
    }
}
```

```
class Server {

    public Worker getConnection(
        String user, String pw) {
        Worker w = new Worker();
        w.start();
        return w;
    }

}

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

    public boolean sendMail
        (String msg, String receiver) {...}

    public String getMail() {...}

}
```