

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".

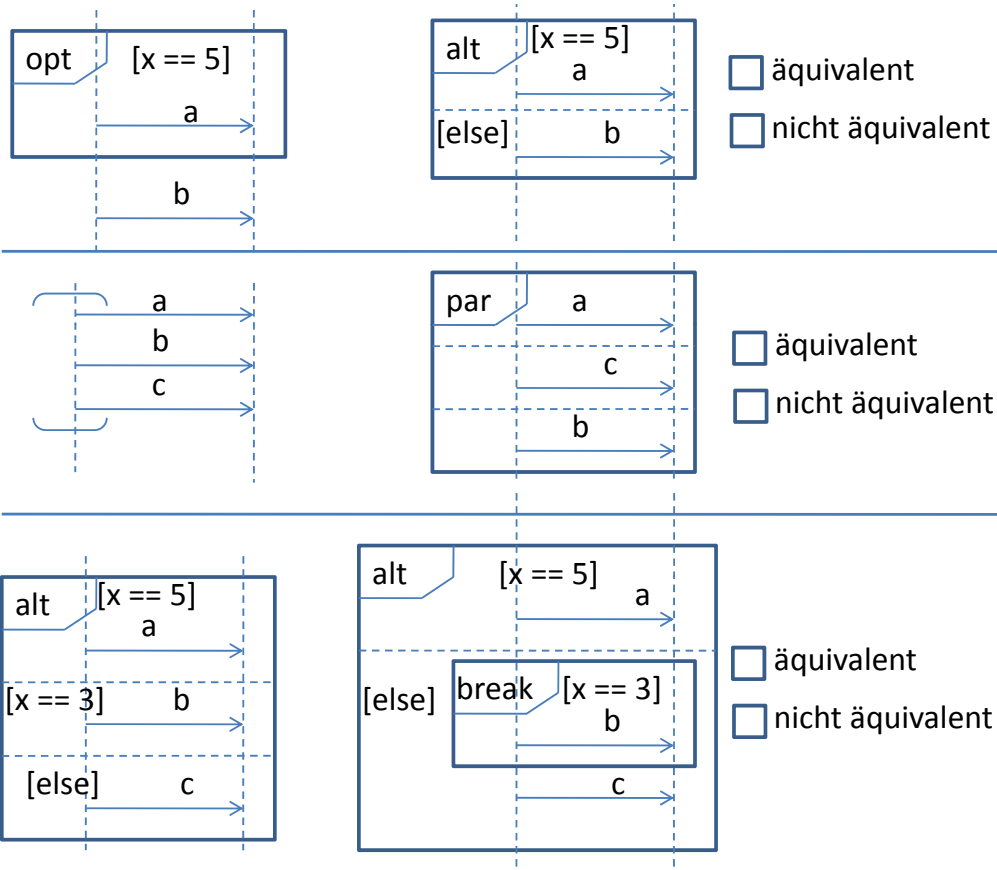
Aufgabe 3: Kommunikation im Sequenzdiagramm

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

In einer Lautsprecherdurchsage wird Passagier Maier gebeten zur Information zu kommen.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Die Frau bei der Information informiert Herrn Maier, dass sein Flug gestrichen wurde.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Sie ruft eine Webseite mit verfügbaren Zimmern auf.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Sie reserviert beim Hotel ein Zimmer per Telefon.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Das Hotel schickt ihr die Daten per E-Mail.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Herr Maier plaudert mit einer anderen Person, die denselben Flug nehmen wollte.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Herr Maier twittert seine Probleme mit dem Flug.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Im Chat trifft er einen Freund und diskutiert sein Problem.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Dann schaut sich Herr Maier die Nachrichten an.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron
Er bekommt ein SMS, dass sein Flug in einer Stunde geht.	<input type="checkbox"/> synchron	<input type="checkbox"/> asynchron

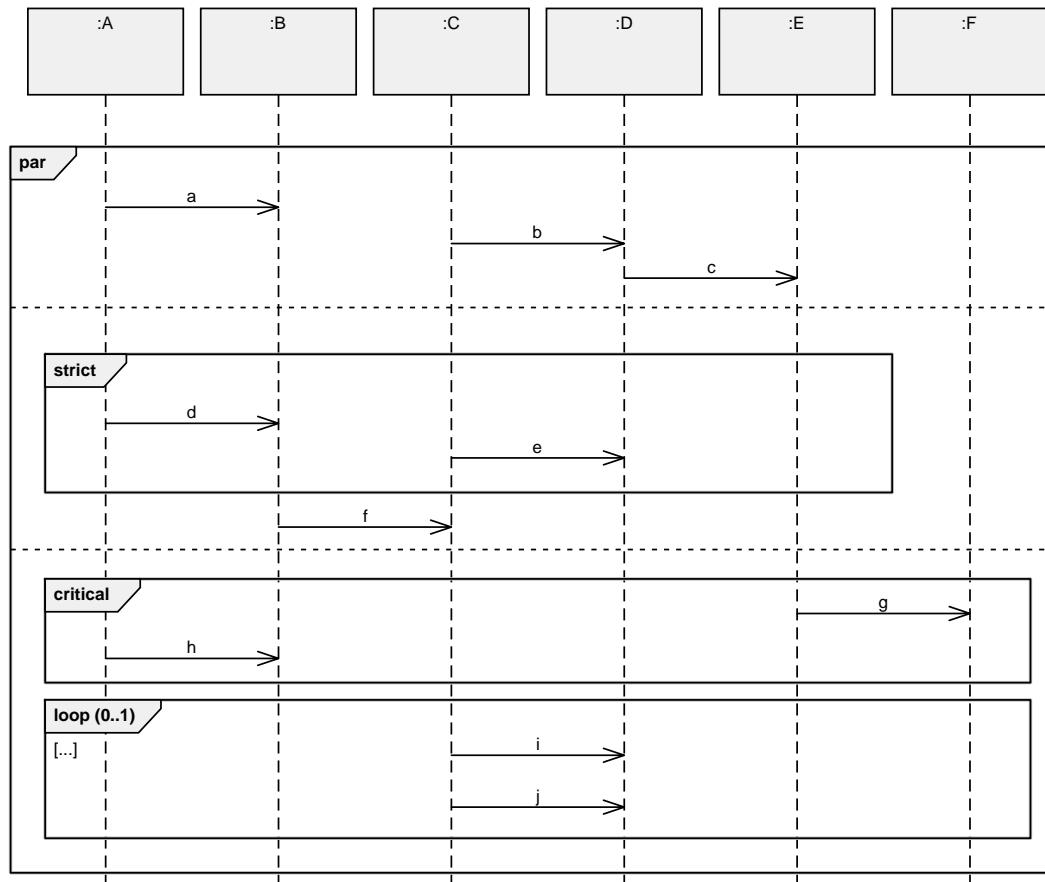
Aufgabe 4: Kombinierte Fragmente

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.



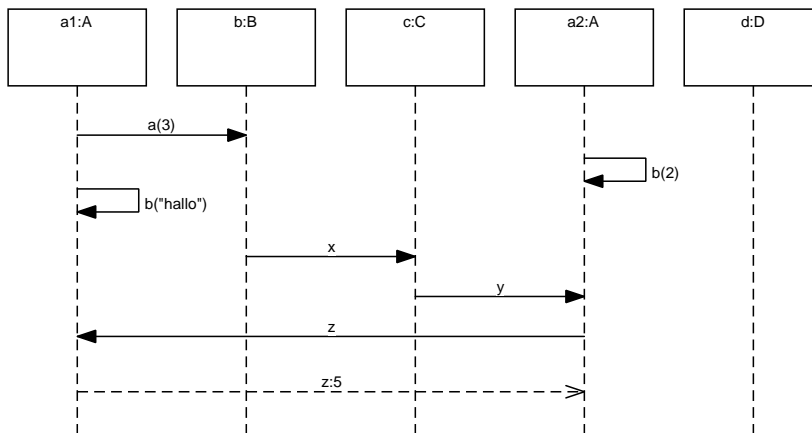
b) **Berechnung von Traces**

Beschreiben Sie alle möglichen Ereignisfolgen des folgenden Diagramms.



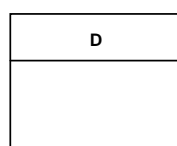
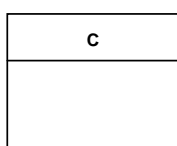
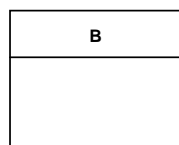
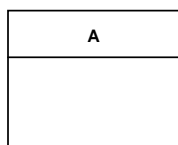
Aufgabe 5: Klassendiagramm aus Sequenzdiagramm

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 des gegebenen Programms mit einem Sequenzdiagramm dar. Modellieren Sie auch allfällige Antwortnachrichten und Parameter.

Sie können davon ausgehen, dass alle nicht explizit deklarierten Variablen bereits deklariert und initialisiert sind.

```
public class Client {
    ...
    String msg=read();

    while(msg!="exit") {
        resp=server.post(msg, forum);

        if(resp!="ok") {
            print("Sending_error");
            exit;
        }

        msg=read();
    }

    String read() {...}
    void print(String msg) {...}
}
```

```
public class Server {
    ...
    String post(String msg, int forumid) {
        Worker w = new Worker();
        w.setMsg(msg, forumid);
        w.start();

        return "ok";
    }
}

public class Worker extends Thread {
    ...
    void setMsg(String msg, int forumid) {
        if(exists(forumid) {
            push(msg, forumid);
        }
    }

    boolean exists(int forumid) {...}
    void push(String msg, int forumid) {...}
}
```