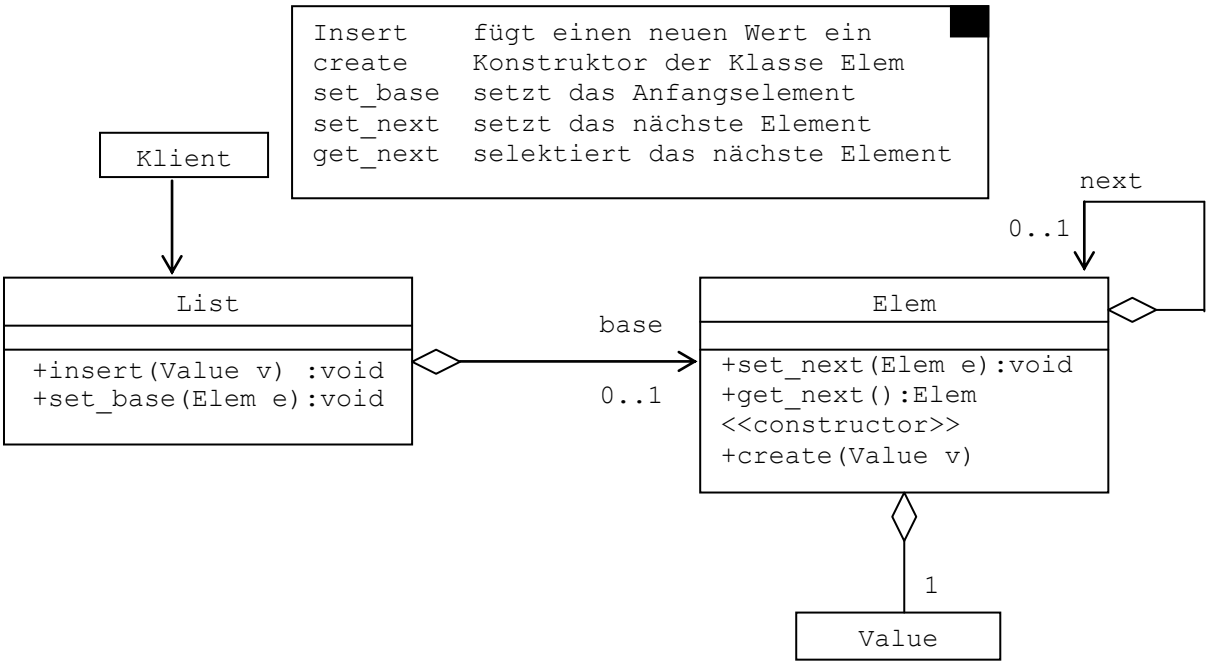
Aufgabenblatt
Softwaretechnik SS 2010

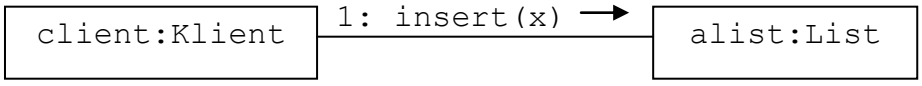
Prof. Dr. Th. Fuchß
Hochschule Karlsruhe – Technik und Wirtschaft
Fakultät für Informatik und Wirtschaftsinformatik
Fachgebiet Informatik

TEIL II
Design

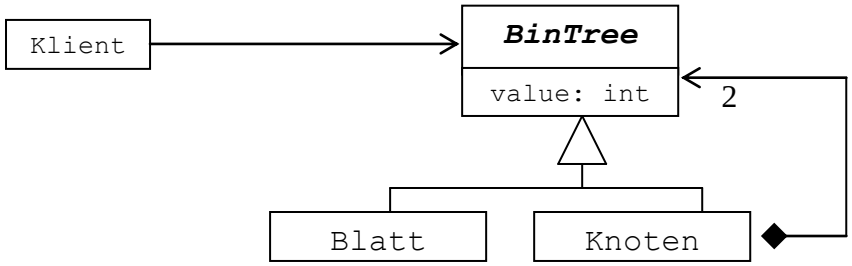
Aufgabe 7
Gegeben sei folgende Struktur einer verketteten Liste:



Skizzieren Sie das Einfügen eines neuen Wertes X (Instanz der Klasse Value) in eine verkettete Liste durch Anhängen am Anfang der Liste. Vervollständigen Sie hierzu das folgende Kollaborationsdiagramm.

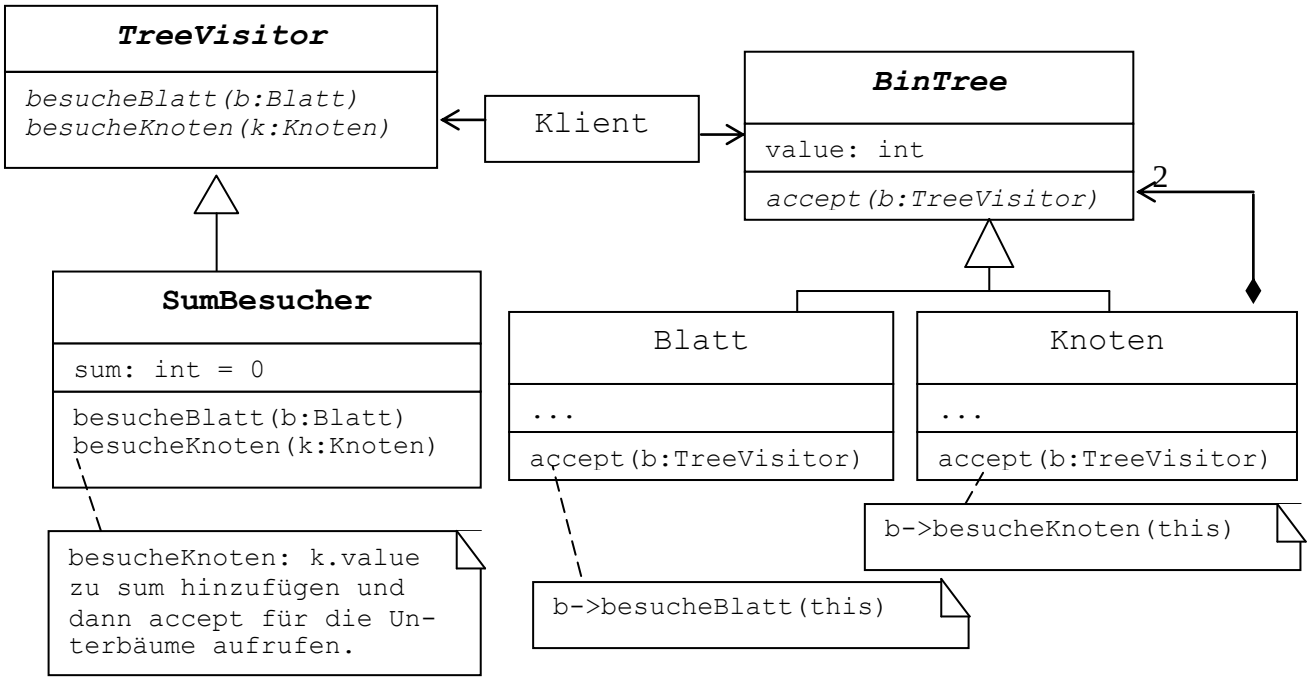


Aufgabe 8
Gegeben sei folgende Struktur:



Skizzieren Sie anhand eines Interaktionsdiagramms einen Tiefensuch-Algorithmus. Wählen Sie hierzu ein geeignetes Szenario und ergänzen Sie dieses um fehlende Operationen und Attribute. Erstellen Sie auf dieser Basis ein Java-Programm.

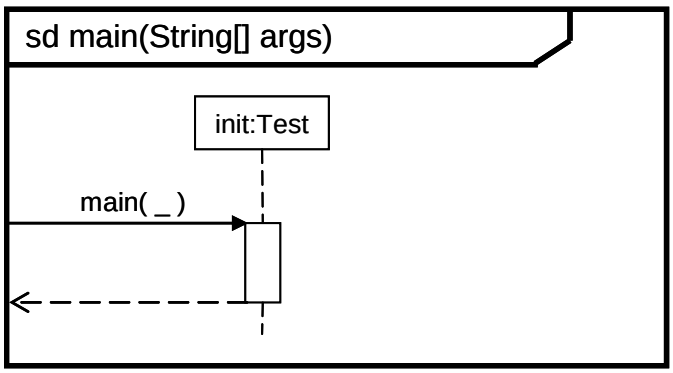
Aufgabe 9
In einem weiteren Schritt soll eine Summenbildung über alle Elemente des Baumes erfolgen. Sie entscheiden sich, dies mittels Besuchermuster zu realisieren. Hierzu entwerfen Sie folgendes Diagramm. **Implementieren Sie dies in Java.**



Aufgabe 10
Gegeben sei folgendes Java-Programm:

```
import java.util.*;

abstract class SortArray {
    protected int[] myArray;
    public void sort() { //Bubble
        int i,j;
        for (i = myArray.length - 1; i > 0; i--) {
            for (j = 0; j < i; j++) {
                if (cond(myArray[j], myArray[j + 1])) {
                    swap(j, j + 1);
                }}}}
    private void swap(int index1, int index2) {
        int tmp = myArray[index1];
        myArray[index1] = myArray[index2];
        myArray[index2] = tmp;
    }
    abstract boolean cond(int val1, int val2);
    public int[] getArray   (){return myArray;}
}

class UpArray extends SortArray {

  public UpArray(int [] anArray) {
       myArray = (int []) anArray.clone();
       this.sort();
   };
  boolean cond(int val1, int val2) { return (val1 > val2);}
}

public class Test{

  public static void main(String[] args) {
    int[] theArray      = {1, 7, 2};
    UpArray   a1  = new UpArray(theArray);
}
```

Beschreiben Sie den Programmablauf nach dem Aufruf der Main-Methode. Vervollständigen Sie hierzu das folgende Sequenzen-Diagramm.

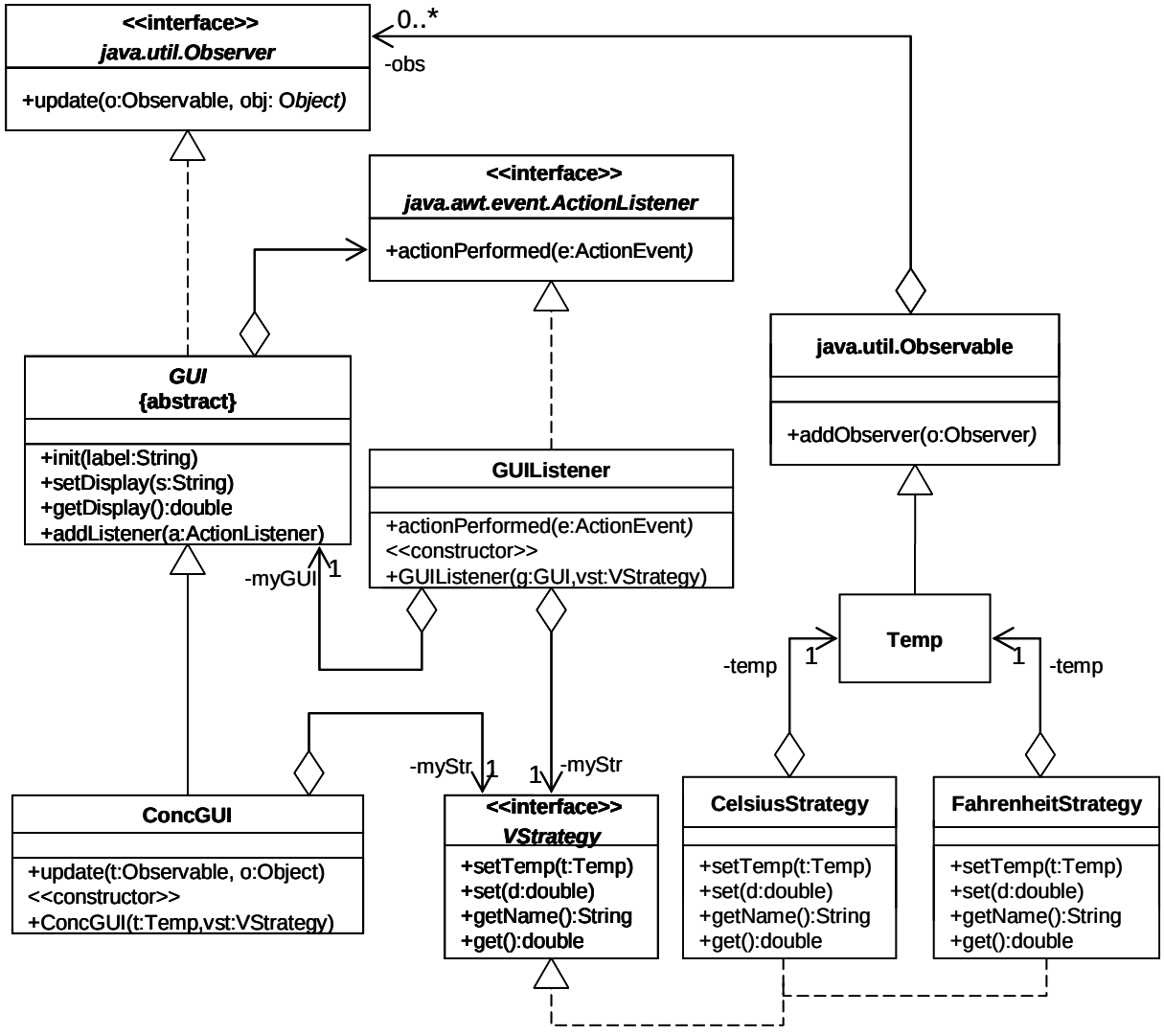


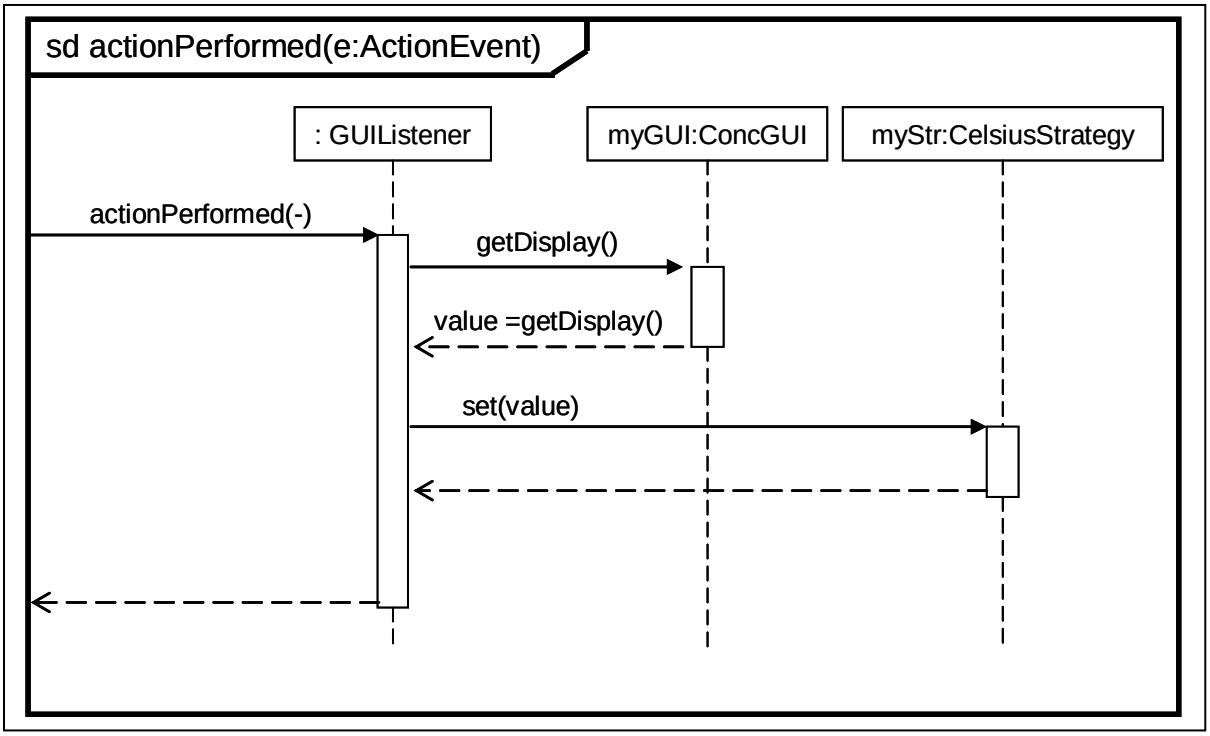
Aufgabe 11

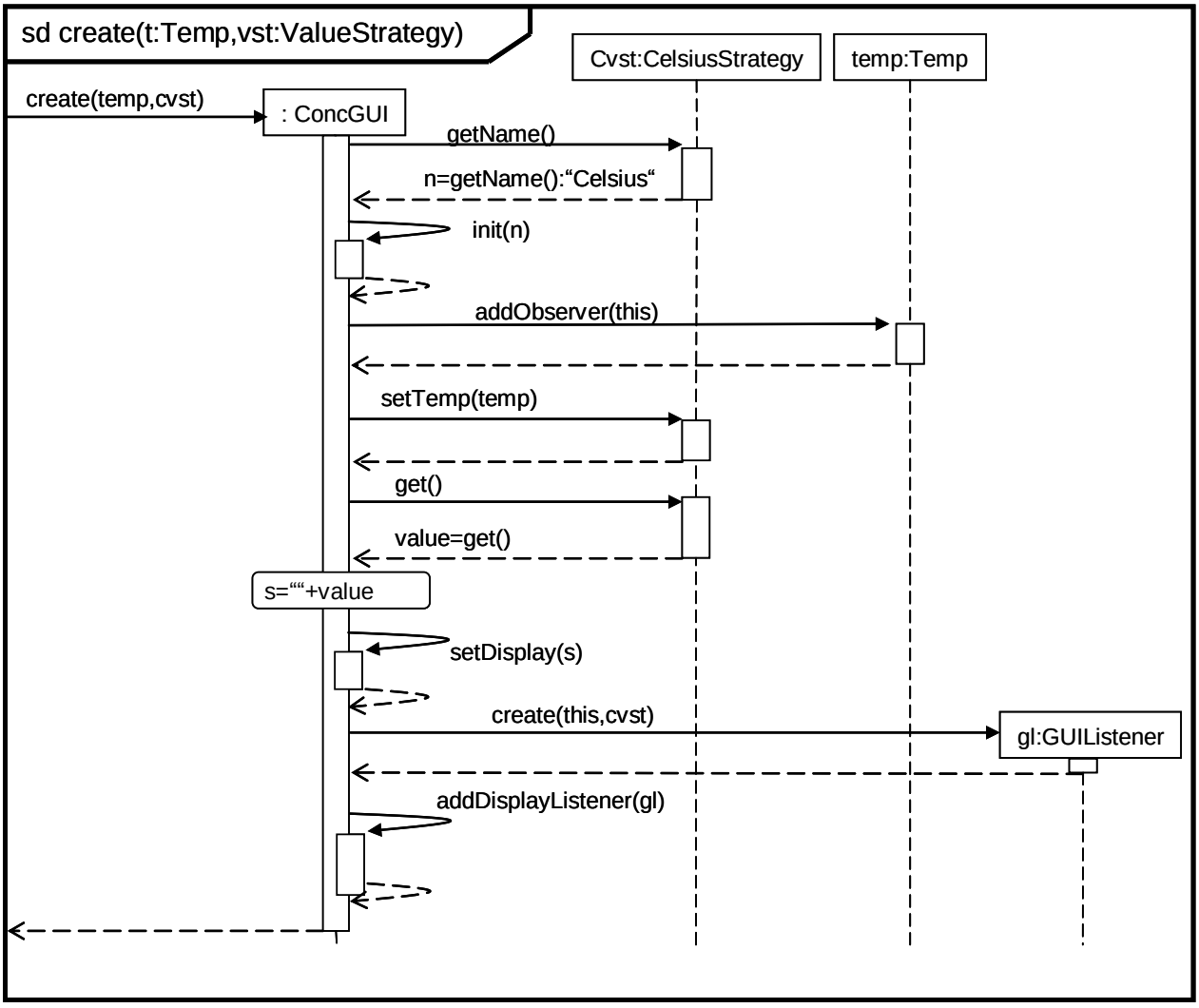
Als Sie nach dem verlängerten Wochenende an Ihren Arbeitsplatz zurückkommen, finden Sie folgende Notiz Ihres Kollegen auf Ihrem Schreibtisch vor:

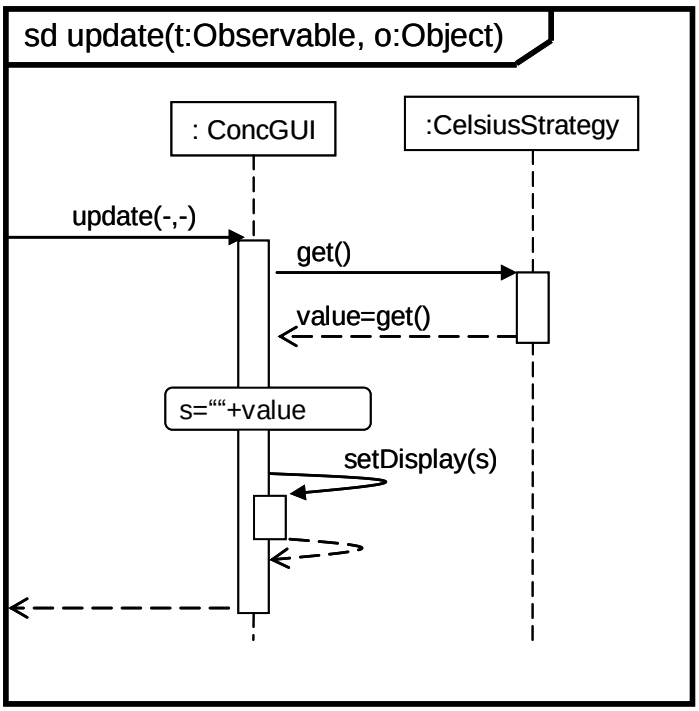
> *„Könntest Du bitte für mich die beiden Klassen „**ConcGUI**“ und „**GUIListener**“ implementieren, ich habe es am Freitag leider nicht mehr geschafft – der Rest ist implementiert. Das Design (Objekt- und Dynamikmodell) findest Du unter ~donald/pub/projectTemperature/ . Bei den Sequenz-Diagrammen habe ich als Szenario stets die Celsiusstrategie verwendet und Angaben zu Returnwerten nur bei Bedarf gemacht.*
> *Gruß Donald* “

Als pflichtbewusster Kollege beginnen Sie sofort mit der Realisierung.
Implementieren Sie die Klassen „ConcGUI“ und „GUIListener“ gemäß Designvorlage in Java.




sd actionPerformed(e:ActionEvent)
: GUIListener
myGUI:ConcGUI
myStr:CelsiusStrategy
actionPerformed(-)
getDisplay()
value =getDisplay()
set(value)


sd create(t:Temp,vst:ValueStrategy)
Cvst:CelsiusStrategy
temp:Temp
create(temp,cvst)
: ConcGUI
getName()
n=getName():“Celsius“
init(n)
addObserver(this)
setTemp(temp)
get()
value=get()
s=““+value
setDisplay(s)
create(this,cvst)
gl:GUIListener
addDisplayListener(gl)



Aufgabe 12

Wieder nutzen Sie den Brückentag um allein und in aller Ruhe Ihren Aufgaben im Büro nachzugehen und liegen gebliebene Aufgaben zu erledigen. Doch wie so oft kommen Sie auch heute nicht dazu, finden Sie doch folgende Notiz Ihres Kollegen Benjamin auf Ihrem Schreibtisch vor:

> *„Könntest Du bitte für mich die Implementierung der Queue abschließen. Ich habe es leider nicht mehr vor dem freien Tag geschafft .Lediglich die Klasse „**Elem**“ und die Methode „**Empty**“ der Klasse „**FIFO**“ sind implementiert. Die Methode „**read**“ habe ich spezifiziert und das Objektmodell ist fertig. Das bisherige Design und den ersten Teil der Implementierung findest Du unter ~beni/pub/projec/ZOO/Stack/*
>
> *Gruß Benjamin* “

Als pflichtbewusster Kollege beginnen Sie sofort mit der Umsetzung, um den Projektverlauf nicht zu gefährden.

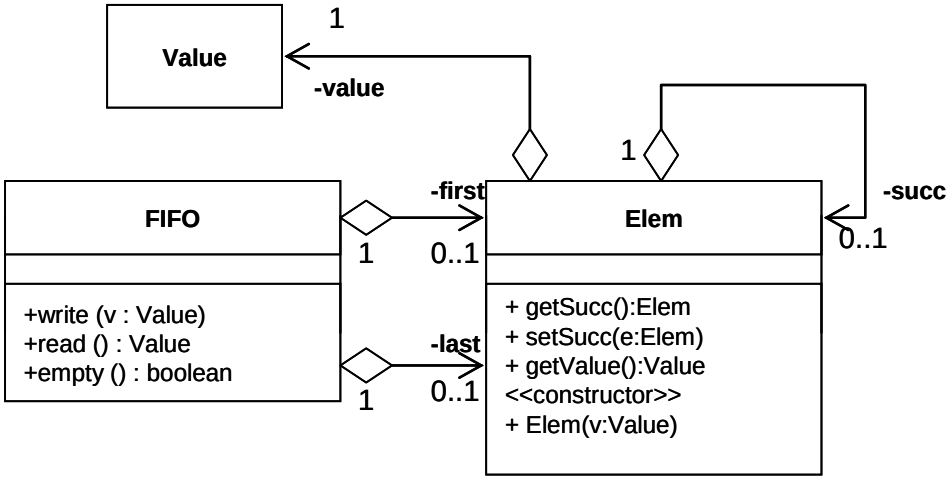
a) Vervollständigen Sie das nachfolgende Design und spezifizieren Sie die Operation **write** der Klasse **FIFO** mithilfe eines Sequenzdiagramms.

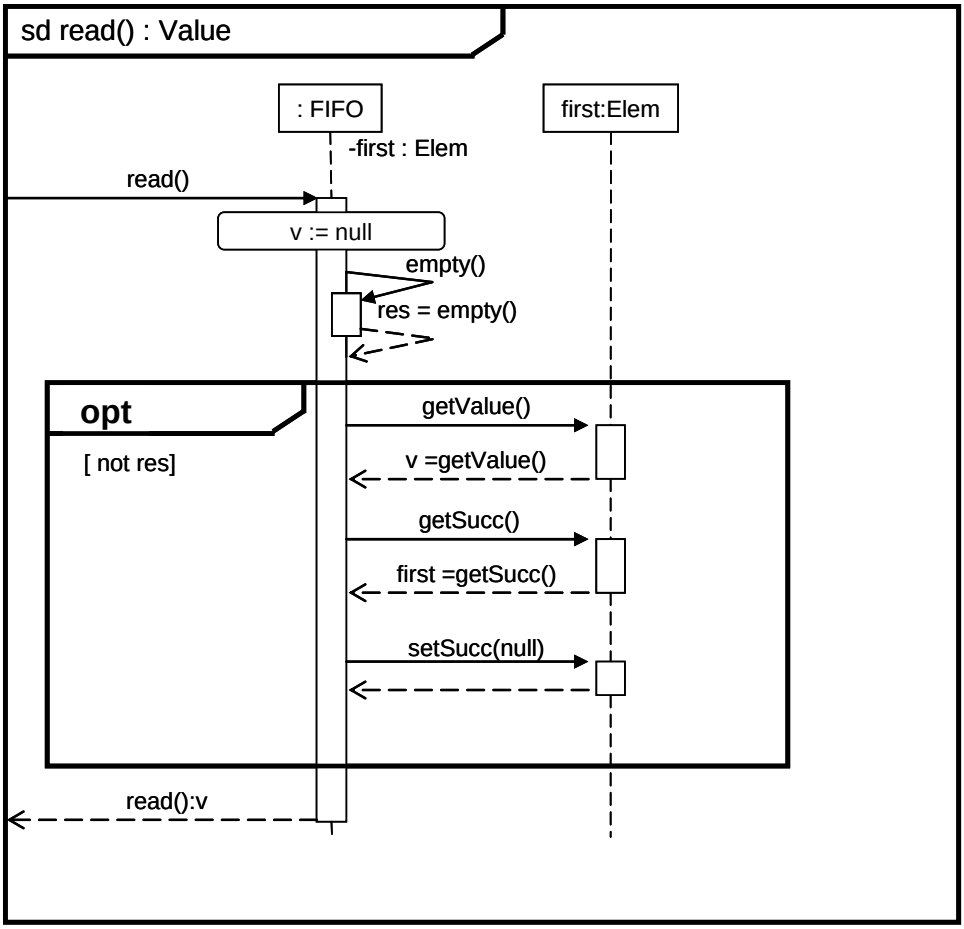
b) Vervollständigen Sie die Java-Implementierung der Klasse **FIFO**, indem Sie die Methoden **read** und **write** implementieren. Nutzen Sie hierzu die Vorarbeiten *Ihres Kollegen* und Ihr Design aus Teil **a**).

write fügt einen neuen Wert hinten in den FIFO ein
read liest den vordersten Wert des FIFO aus und entfernt ihn
empty prüft, ob der FIFO leer ist

setSucc setzt das nachfolgende Element
getSucc selektiert das nachfolgende Element
getValue liefert den Wert des Elements

Elem erzeugt ein neues Element mit entsprechendem Wert





```
class FIFO {
    private Elem first;
    private Elem last;

    public boolean empty(){
        return this.first == null;
    }
    private void setFirst(Elem e){
        this.first = e;
    }
    private void setLast(Elem e){
        this.last = e;
    }


    public void write(Value v){
    /*
      hier ergänzen
    */
    }

    public Value read(){
    /*
      hier ergänzen
    */
    }
}
```