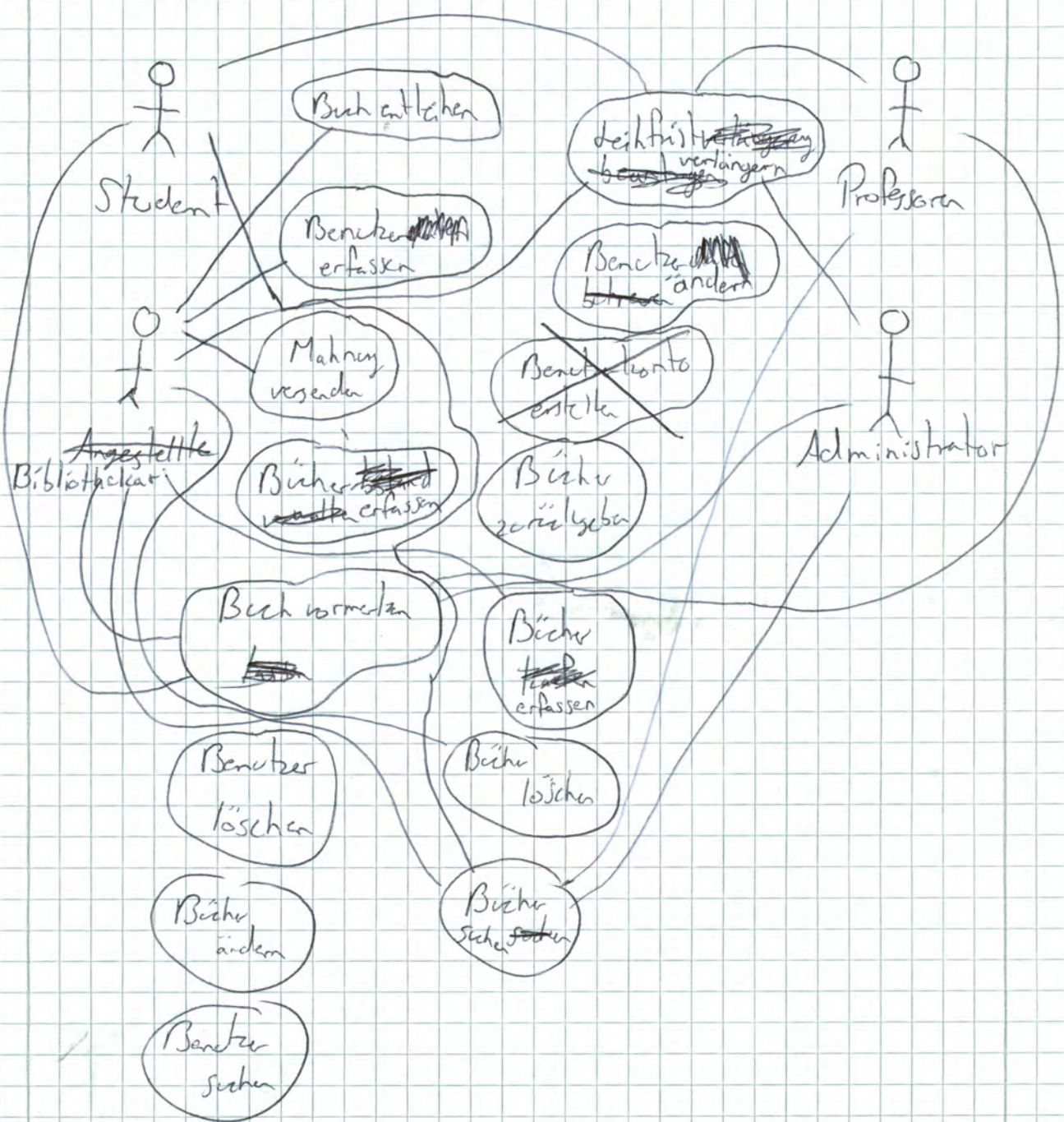
SuF
03.04.2014

Aufgabe

Erstellen Sie ein Anwendungsfalldiagramm für ein Software-System einer Hochschulbibliothek.

Beschreiben Sie jeden Anwendungsfall in 1-2 Sätzen.

♀ = Alleine



SWEV
17.04.14

Anwendungsfall Buch ausleihen (1)

- Anwendungsfall: Buch ausleihen
- Ziel : Buch ist ausgeliehen
- Kategorie : primär
- Vorbedingung: keine
- Nachbedingung Erfolg: Ausleihe eingetragen
- Nachbedingung Fehlschlag: Buch nicht ausleihbar oder Leser gesperrt: Absage an Leser
- Akteure: Bibliothekarin
- Auslösendes Ereignis: Leser steht mit Buch an der Ausleihe

- Beschreibung:
  1. Bibliothekarin gibt Lesernummer ein
  2. System identifiziert Leser
  3. Bibliothekarin gibt Buchnummer ein
  4. System identifiziert Buch
  5. System trägt Buchausleihe ein

- Erweiterungen:

  2a Leser gesperrt: Absage an Leser, Buch einbehalten, Anwendungsfall Ende

  4a Buch nicht ausleihbar: Absage an Leser, Buch einbehalten, Anwendungsfall Ende

- Alternativen:

  1a keine Lesernummer vorhanden: Bibliothekarin erfasst Leser neu, gehe zu 3

Anwendungsfall: Buch verlängern

- Anwendungsfall: Buch verlängern ✓
- Ziel: Buch ist verlängert ✓
- Kategorie: sekundär (da System würde auch funktionieren wenn Anwendungsfall nicht wäre → einfach nochmal ausleihen)
- Vorbedingung: ~~Buch ist ausgeliehen~~ Benutzer ist angemeldet (Vorbedingung nicht „zu viel" machen)
- Nachbedingung Erfolg: (Verlängerung eingetragen) neues Rückgabedatum eingetragen
- Nachbedingung Fehlschlag: ~~Buch nicht verlängert oder kein~~ für Verlängerung gesperrt: Absage an Leser, Altes Rückgabedatum
- Akteure: ~~Bibliothekarin~~ Benutzer (da online Fall)
- Auslösendes Ereignis: Leser ~~steht mit ausgeliehenem Buch~~ an der Verlängerung/Ausleihe ~~der~~ Benutzer will Buch verlängern
- Beschreibung:
  1. ~~Bibl. gibt Leserausweis ein~~ Benutzer will ausgeliehene Bücher sehen
  2. ~~System identifiziert Leser~~ System zeigt ausgeliehene Bücher an
  3. ~~Bibl. gibt Buchnr. ein~~ Benutzer wählt Bücher aus und verlängert diese
  4. ~~System identifiziert Buch~~ System zeigt ausgeliehene Bücher mit neuem Rückgabedatum an.
  5. System identifiziert Ausleihe
  6. ~~Bibl. trägt Verlängerung ein~~

- Erweiterungen (= Erweiterung der einzelnen Aktion):
  - 2a: Leser gesperrt: Absage an Leser, Buch einbehalten, Anwendungsfall Ende
  - 4a: Buch nicht verlängerbar: Absage an Leser, Buch einbehalten, Anwendungsfall Ende
  - 5.: Buch nicht ausgeliehen: Absage an Leser, Anwendungsfall Ende (Benutzer)

3a 2a kein Buch verlängerbar: Anwendungsfall Ende
4a weil Verlängerung Buch ausgewählt: System regt Fehlermeldung an

und altes Rückgabedatum wird beibehalten. 4b: Benutzer will weitere Bücher ausleihen: gehe zu 3

Alternativen: (Alternative Ausführung der jeweiligen Aktion)

3a Benutzer will nicht verlängern: Anwendungsfall Ende

Anwendungsfall: Ware kaufen (siehe AB Projekt Computerspiel)

Vorbedingung: Tätigkeit annehmen und Ware verkaufen an diesem Tag nicht ausgeführt

Beschreibung:

1. Spieler will Ware sehen
2. System zeigt ~~Ware~~ verfügbare Waren an
3. Spieler wählt Ware und Menge aus und kauft diese
4. System übernimmt Ware in Besitz des Spielers, reduziert Goldstücke

Alternativen:

4a Goldstücke nicht ausreichend: Spieler hat keine Ware gekauft

4b Gewicht der Ware zu groß:

Erweiterung:

2a Benutzer will keine Ware kaufen: Anwendungsfall Ende

SHF
24.04.13

Aufgabe

- Finden Sie nicht funktionale Anforderungen für das Software-System einer Hochschulbibliothek

Technologische Anforderungen:

Betriebssystem: Windows Server oder Linux

Arbeitsspeicher 4 GB

Datenbanksystem: SQL-Server, MySQL

Programmiersprache: .Net, Java, PHP

Qualitätsanforderungen:

neue Versionen Betriebssystem und Datenbanksystem mit max 10 Aufwandstagen einführbar

Funktionen für Leser ohne Schulung bedienbar

" " Bibliothekar(in) nach 3 Tagen Schulung bedienbar

Antwortzeiten: max 3 Sekunden

Funktionen, die Datenbestand ändern mit Passwort schützen

Funktionen für Bibliothekar(in) Mo-Fr: 9-20 Uhr

" " Leser Mo-So: 0-24 Uhr

max 6h Ausfall pro Woche zw. 20-9 Uhr

~~Funktionen für Leser ohne Webanwendung~~

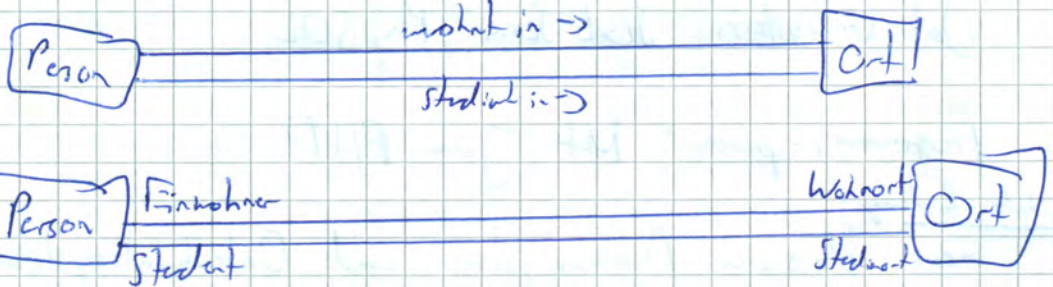
Anforderungen an Benutzeroberfläche

Funktionen für jeder als Webanwendung

Unterstützte Sprachen: de,

Aufgabe

- Wie kann man die Beziehungen "wohnt in" und „studiert in" zwischen Person und Ort durch Kardinalitäten angeben?

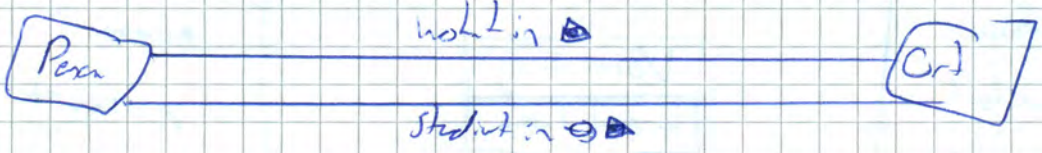


SWEV
25.04.14

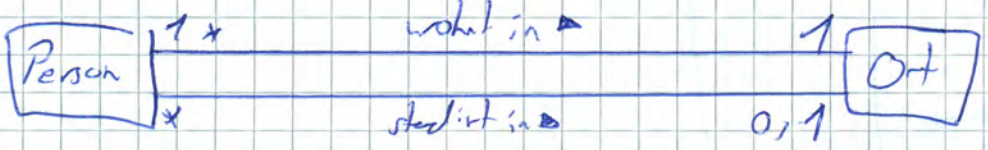
Aufgabe:

~~Aufgabe~~

- Geben Sie die Multiplizität bei der Beziehungen "wohnt in" und "studiert in" an.



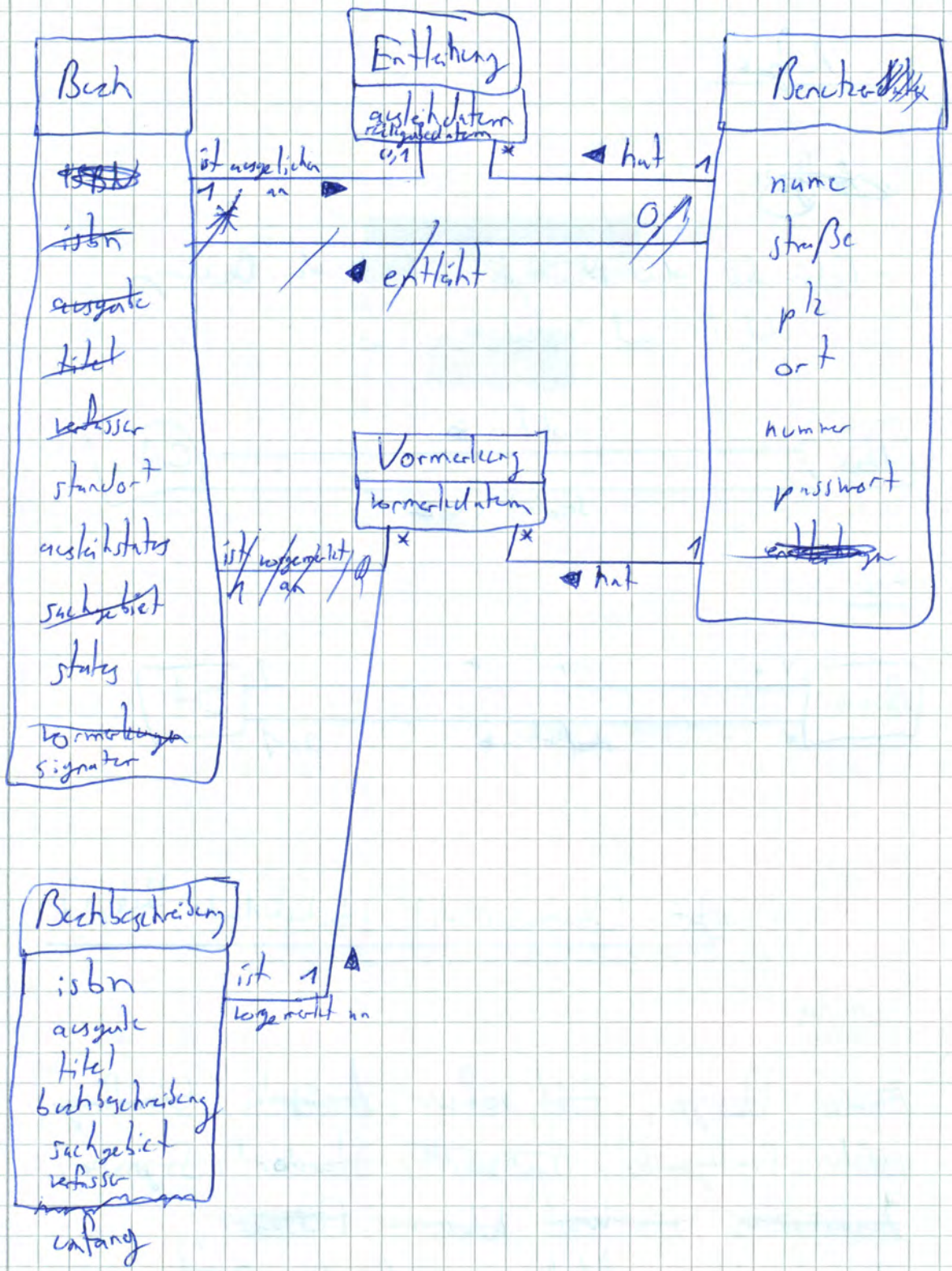
Lsg:



~~Das~~ Domainmodell Hochschulbibliothek (4)

Substantive:

~~Analyse~~, ~~Design~~, ~~Titel~~, ~~Verfasser~~, ~~Ausgabe~~, ~~Umfang~~, ~~ISBN~~, ~~Schlagwörter~~, ~~Sachgebiete~~, ~~Standort~~, ~~Signatur~~, ~~Ausleihstatus~~, ~~Passwort~~ Nummer, ~~Passwort~~, ~~Benutzerdaten~~, ~~Name~~, ~~Straße~~, ~~Plz~~, ~~Ort~~, ~~Status~~, ~~Entleihungen~~, ~~Vormerkungen~~


Buch
isbn
ausgabe
titel
verfasser
standort
ausleihstatus
sachgebiet
status
vormerkungen
signatur
Entleihung
ausleihdatum
rückgabedatum
Vormerkung
vormerkdatum
Benutzer
name
straße
plz
ort
nummer
passwort
buchbeschreibung
isbn
ausgabe
titel
buchbeschreibung
sachgebiet
verfasser
umfang
ist ausgeliehen an
1
*
0,1
hat
1
0,1
entleiht
ist vorgemerkt an
1
0,n
hat
1
ist 1
vorgemerkt an

SWEV
8.05.2014

~~Ei~~ (Geschäftsregeln Folie)

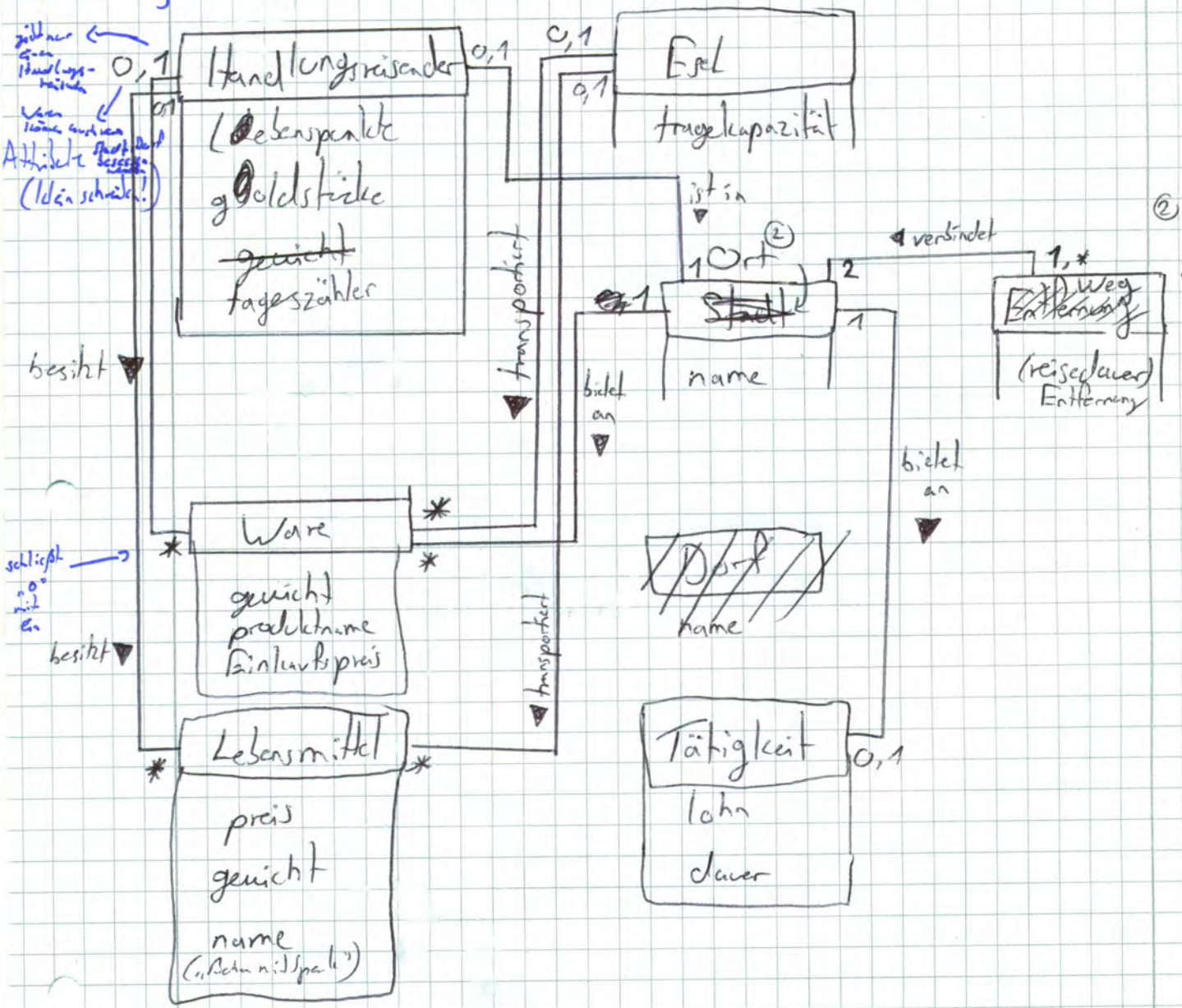
① kann man weglassen, da für das Funktionieren des Spiels nicht relevant

Substantive:

~~Handlungsreisender~~, ~~Stadt~~, ~~Ware~~, ~~Dorf~~, ~~Esel~~, ~~Goldstücke~~, ~~Gewicht~~, ~~Tragkapazität~~, ~~Preis~~, ~~Lebenspunkte~~, ~~Lohn~~, ~~Tätigkeit~~, ~~Kaufen~~, ~~Verkaufen~~, ~~Dauer~~, ~~Lebensmittel~~, ~~Interesse~~, ~~Kapital~~ ①, ~~Gras~~ ①, ~~Entfernung~~, ~~Warscheinlichkeit~~, ~~Produkt~~, ~~Töpfe~~, ~~Geschirr~~, ~~Hacke~~, ~~Hammer~~, ~~Weizen~~, ~~Mais~~, ~~Kartoffeln~~, ~~Rüben~~

„Vielfachheiten"

Illusie



② Stadt + Dorf zusammengefasst

Note:
„Dinge, die Raum einnehmen, werden eine Illusie"

Modellieren Sie ein Zustandsdiagramm für eine Zapfsäule an einer Tankstelle.

SWE V
15.05.14

Ereignisse: (alles etwas, dass von außen passiert)

~~Strom einschalten~~, ~~Zapfpistole entnehmen~~, ~~Zapfpistole einhängen~~, ~~Zapfsäule freigeben~~, ~~Strom ausschalten~~

Zustände: (wenn etwas länger dauert)

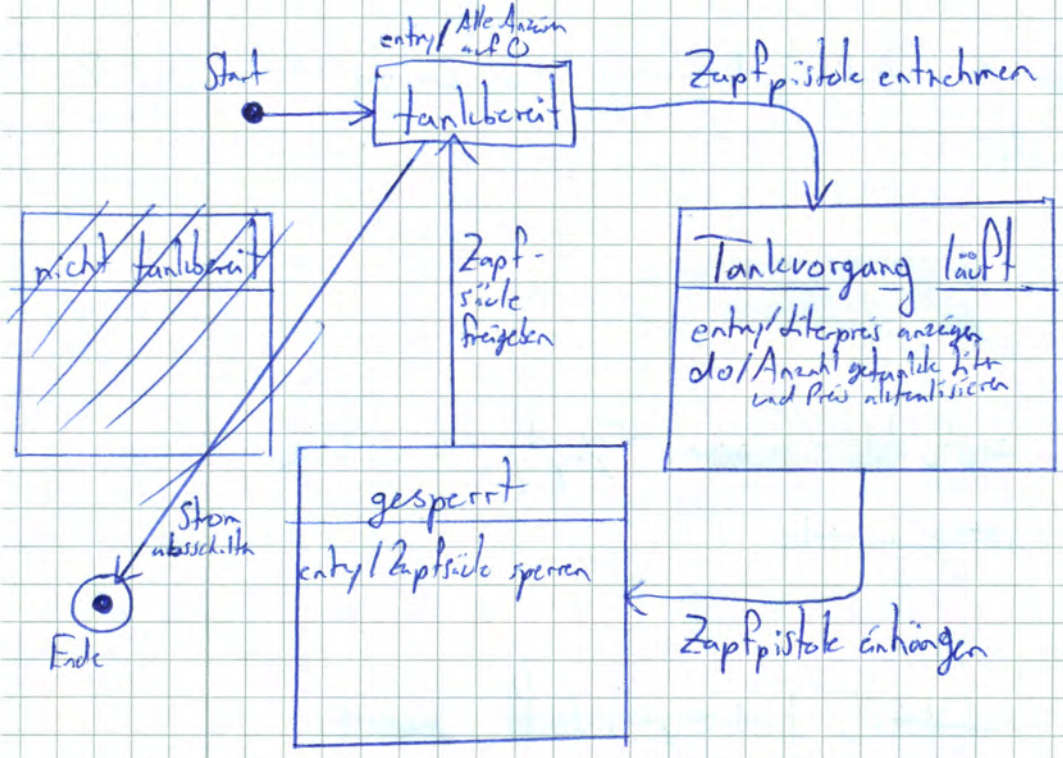
~~tankbereit~~, ~~nicht tankbereit~~, Tankvorgang läuft, ~~gesperrt~~

Bedingung:

~~Stecker/alle/Anzeigen auf/0~~ → nicht relevant
(Events in Aufgabe, nicht ~~aus~~ Ausgangsbasis)

Aktion:

~~Anzeigen auf 0 stellen~~, ~~Literpreis anzeigen~~, ~~Anzahl getankte Liter und Preis aktualisieren~~, Zapfsäule sperren

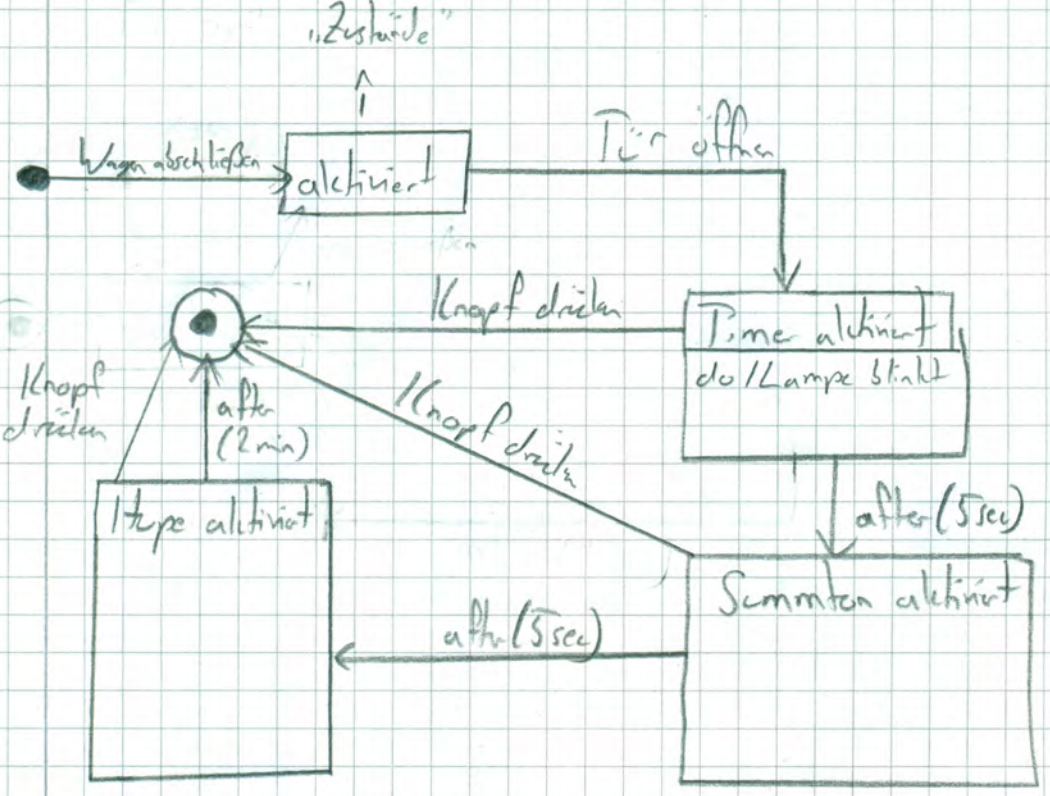


(Aufgabe: Kaffeeautomat zubereiten und ausschenken)

SWEV
16.05.14

Aufgabe: Modellieren Sie ein Zustandsdiagramm für eine Alarmanlage in einem Auto.

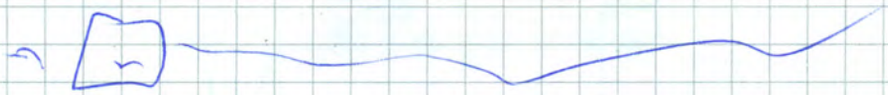
Zustände: ~~aktiviert~~, ~~deaktiviert~~, ~~Summton aktiviert~~, ~~Timer aktiviert~~
~~Lampe blinkt~~, ~~Hupe aktiviert~~, Hupe deaktiviert,



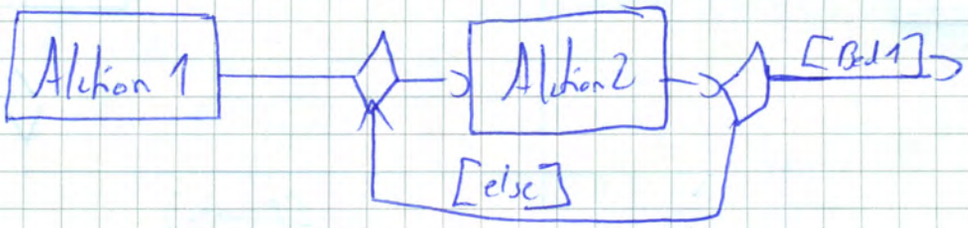
SWE
22.05.2014

Aufgabe

Was ist falsch?



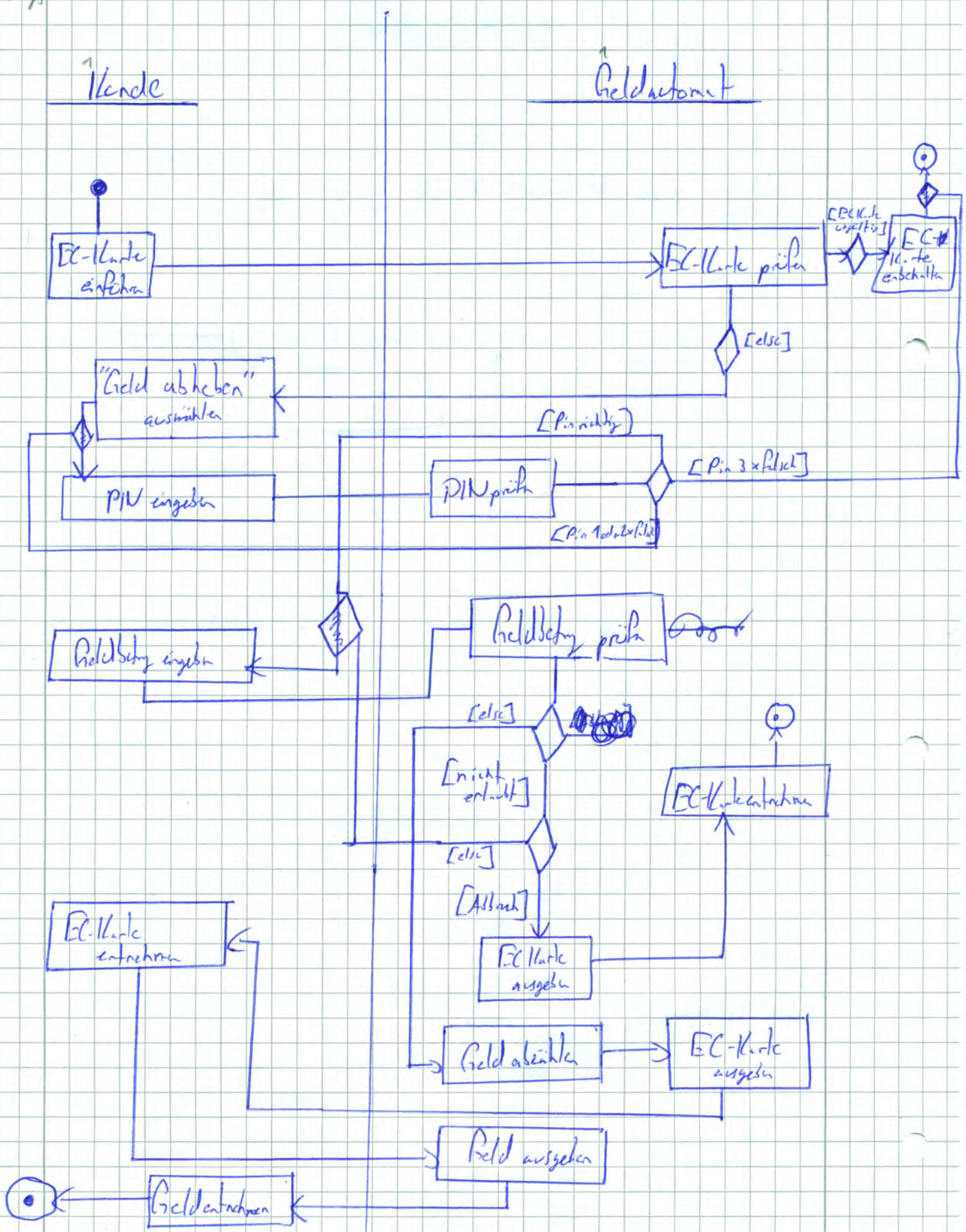
Lösung: [so sieht korrekte Modellierung einer Schleife aus]



Aufgabe

Modellieren Sie für den Anwendungsfall "Geld am Bankautomaten abheben" ...

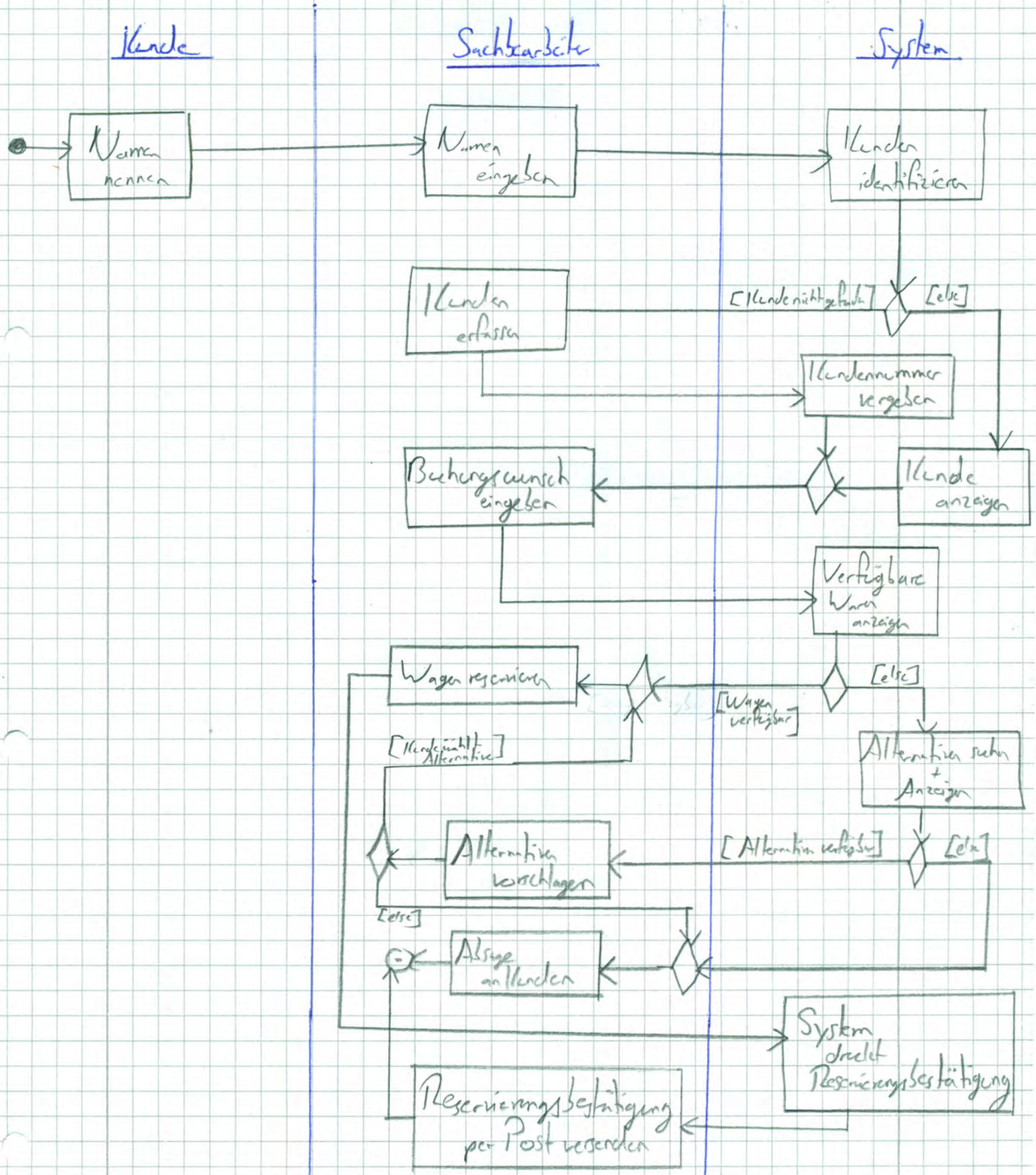
Verantwortungsbereiche



SWE
23.05.2014

Aufgabe

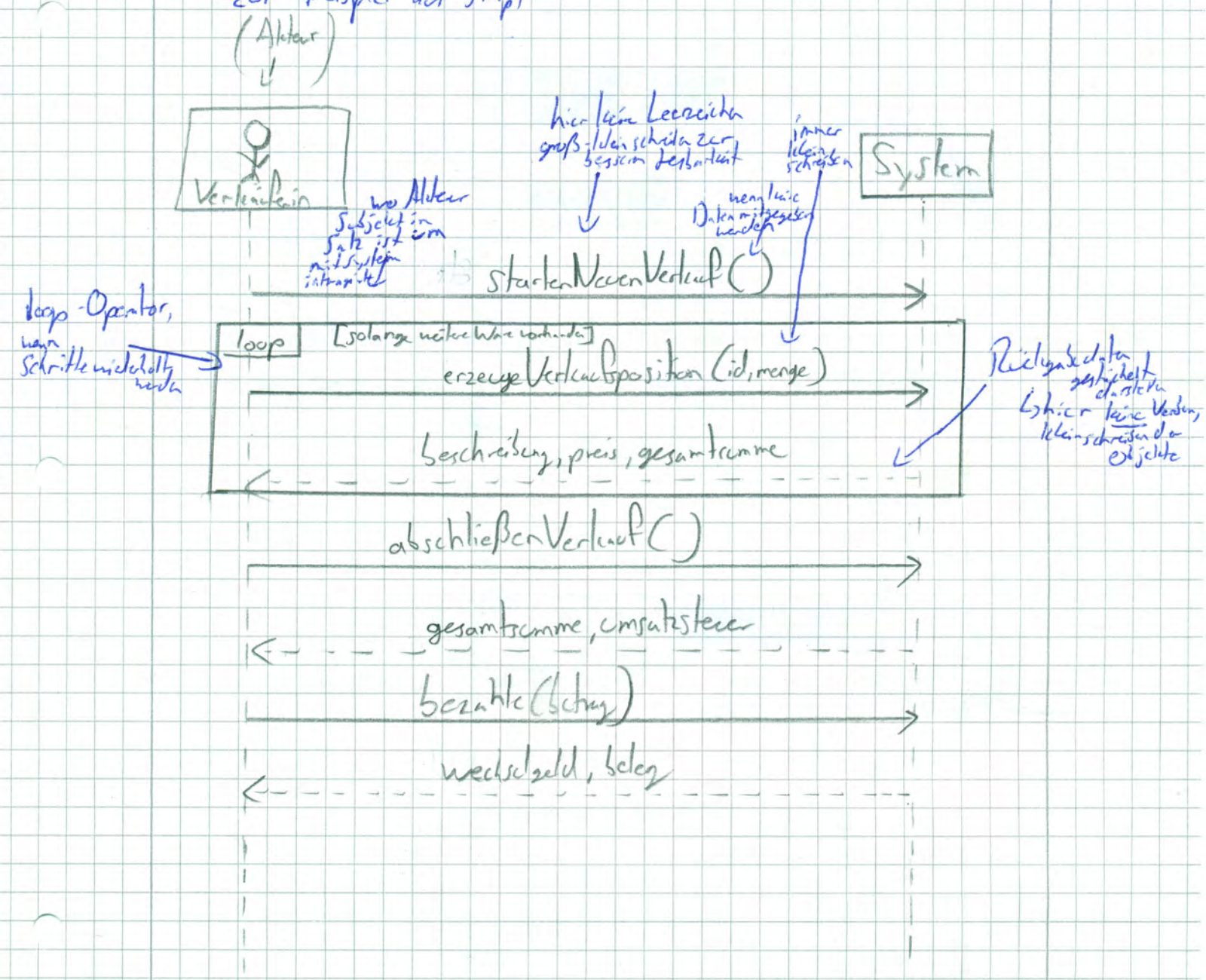
Modellieren Sie Anwendungsfall „Miete Telefonzelle"



System-Sequenzdiagramm

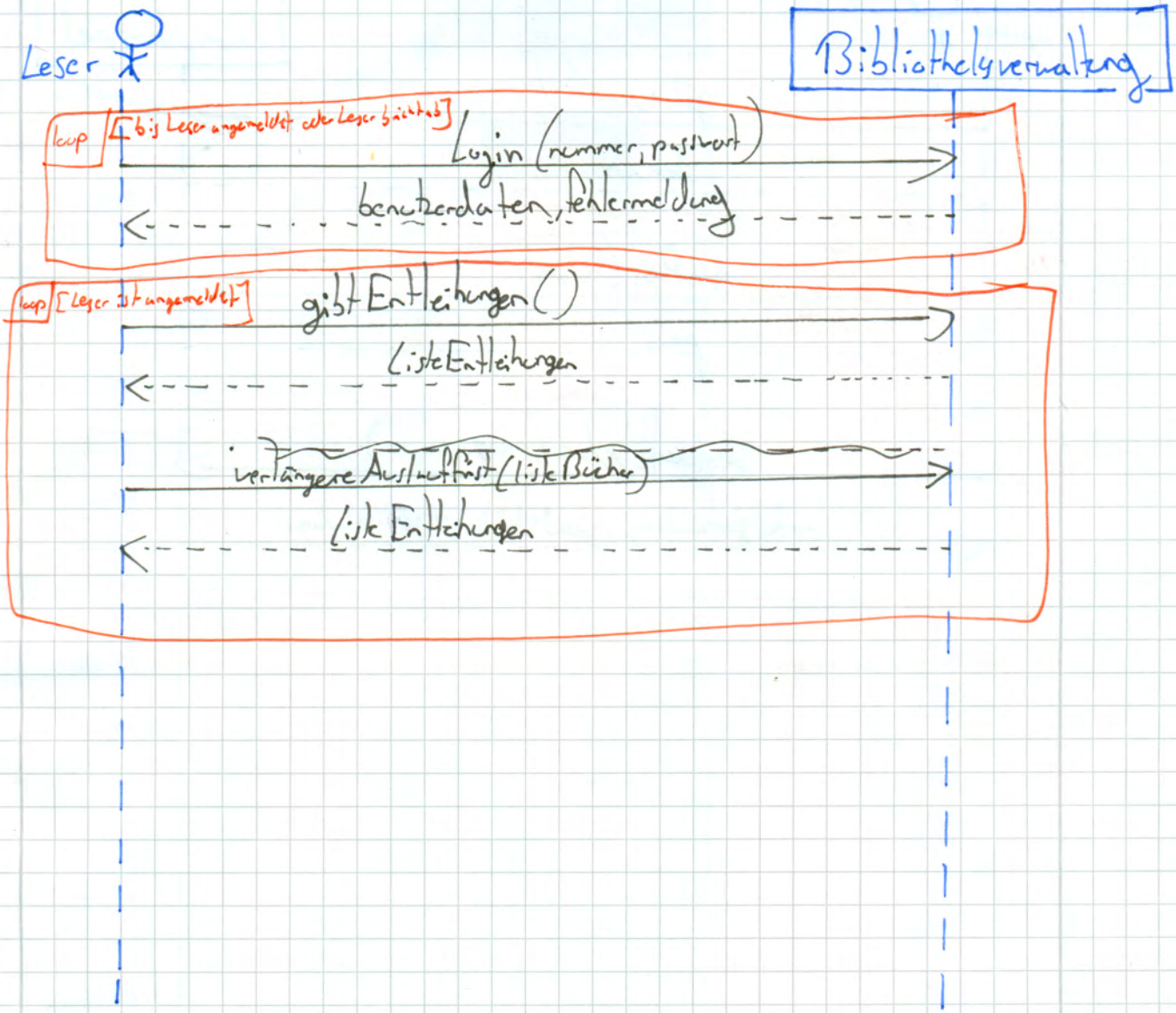
→ !: Nachrichten immer von links nach rechts! (Akteur zu System)

zur Beispiel mit Skript

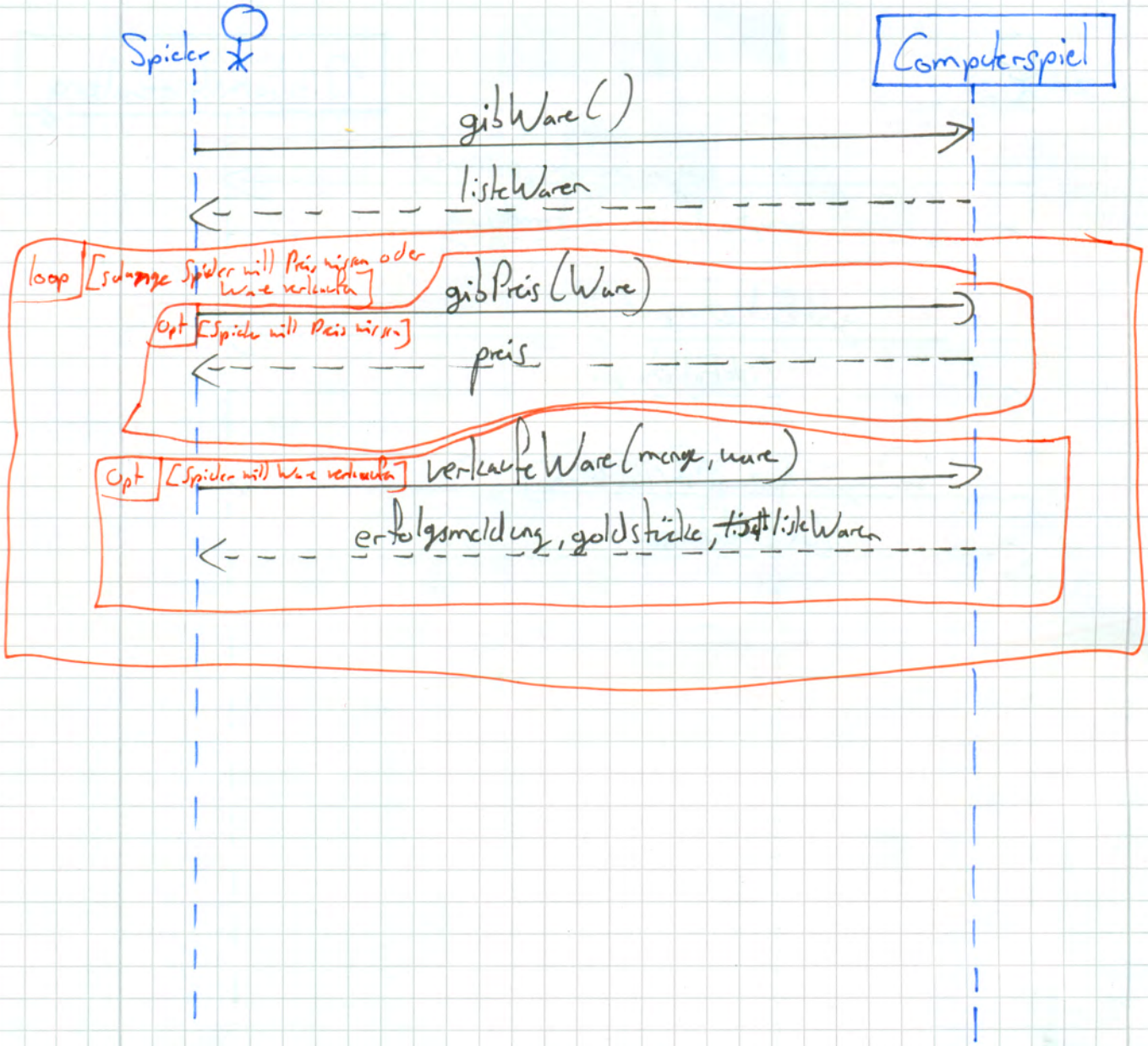


SWE
05.06.14

Aufgabe: Systemsequenzdiagramm "Buch verlängern" erstellen



Aufgabe: Systemsequenzdiagramm Ware verkaufen



1 Domänenmodell Verkaufsanwendung

bezahle(gegebener Betrag)

Vorbedingung: aktueller Verkauf existiert
aktueller Verkauf ist vollständig ist wahr

Nachbedingung: Bezahlungsobjekt b wurde erzeugt
b.gegebener Betrag wurde gegebener Betrag zugewiesen
b wurde mit aktuellem Verkauf verbunden
b wurde mit Hauptbuch verbunden

beende Verkauf()

Vorbedingung: aktuellerVerkauf existiert

Nachbedingung: aktueller Verkauf ~~ist Voll~~ ist Vollständig wurde wahr

Aufgabe

- Beschreiben Sie die Operationskontrakte für die Systemoperationen:

  verkaufeWaren(ware, menge) aus dem Anwendungsfall
  Ware verkaufen

verkaufeWare(ware, menge)

Vorbedingung: ~~Ware befindet~~ ware befindet sich in Anzahl menge
im Besitz des Spielers

Nachbedingung: spielfigur.geldstücke wurden um ware.verkaufspreis * menge
erhöht

ware.anzahl wurde um menge reduziert

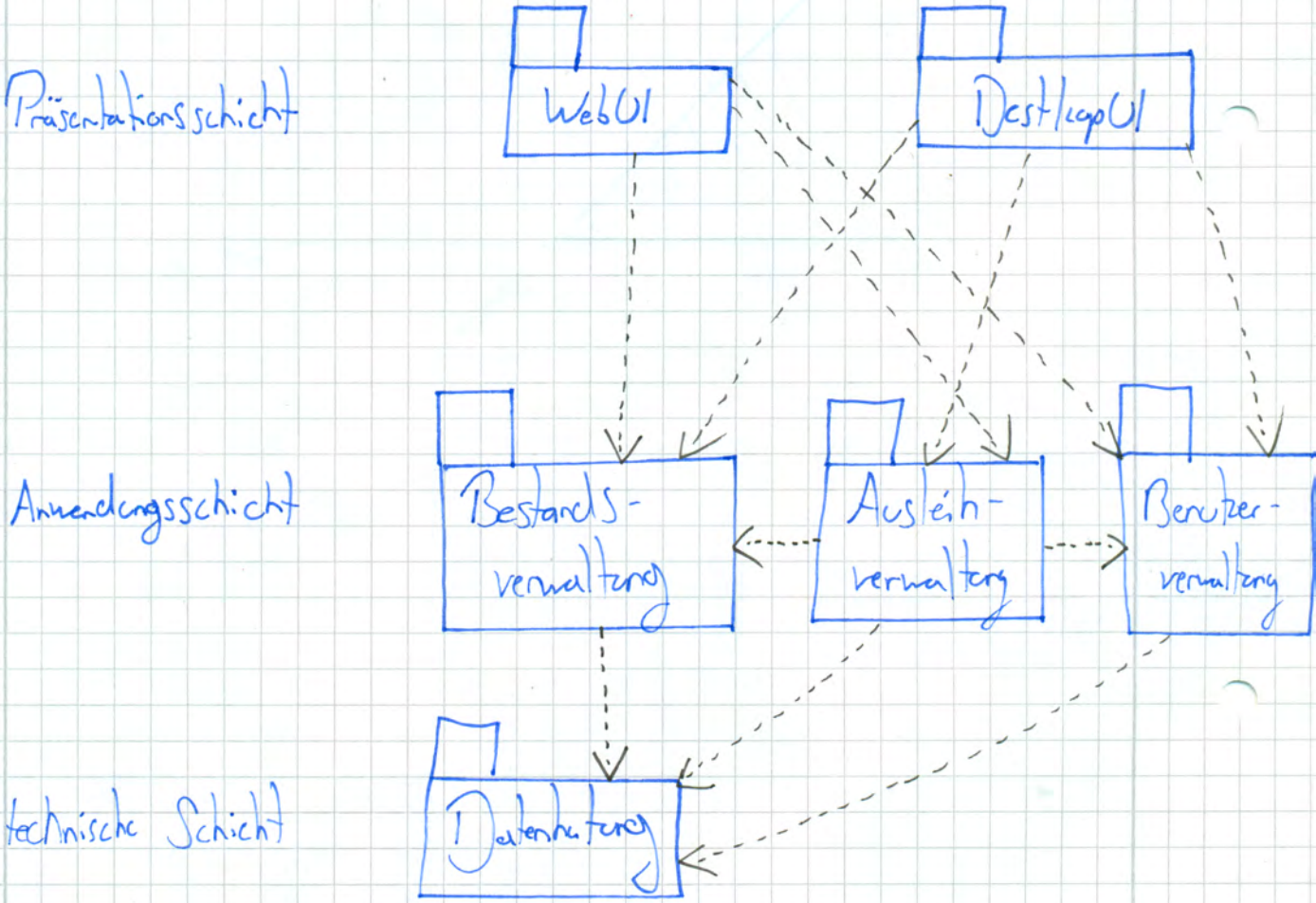
ware.anzahl wurde um menge reduziert wenn
ware.anzahl 0 wurde, wurde Verbindung von
Ware zu Spielfigur gelöscht

SWE V
12.06.14

Aufgabe

- Entwerfen Sie ein Paketdiagramm ..... Software System ... Hochschulbibliothek
- Gehen Sie von einer Drei-Schichten ....
  (→ siehe Skript)

Lösung:



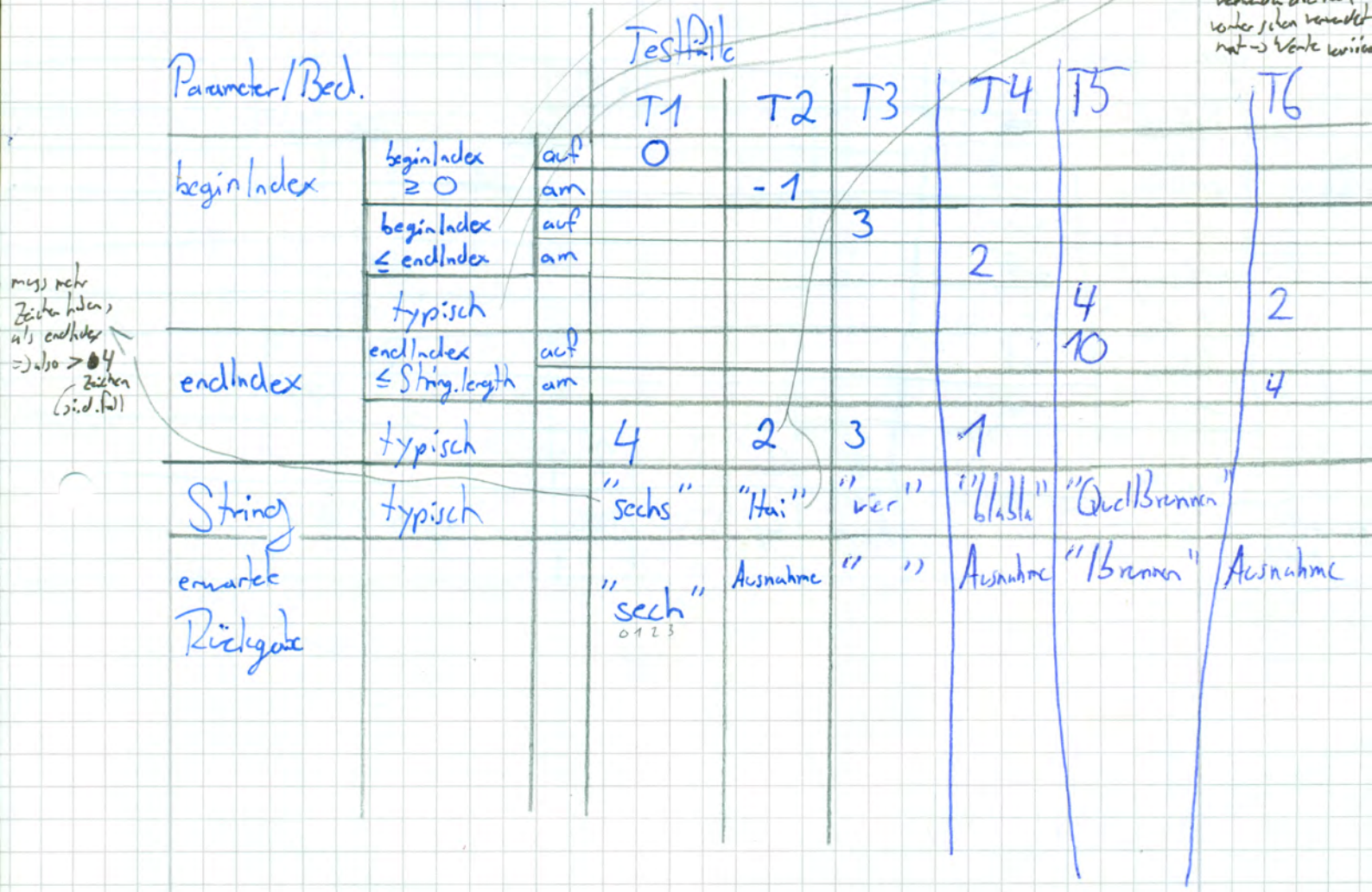
Aufgabe

13.06.14

- Die Klasse String hat die Methode String substring (int .....
- ...
- Ermitteln Sie Testfälle ⇒ siehe Skript!

für die Methode...

Lösung



19.06.14

~~Parameter~~

Aufgabe aus Skript:

Aufgabe:

- In einem Computerspiel erhält der Spieler Nachlass auf die Preis...
- .....

Parameter/Bedingung			T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
preis	preis ≤ 0	auf	0							
		an		1						
	preis ≥ 500	auf			500					
		an				499				
	typisch	~~=~~					20	2	200	10
erfahrung	erfahrung ≤ 0	auf					0			
		an						1		
	erfahrung > 0	auf							/	
		an							/	
	erfahrung > 10	auf							11	
		an								10
	typisch		4	2	3	2				
Rückgabe = Nachlass			Ausnahme	4	5500	1996	0	4	4600	200

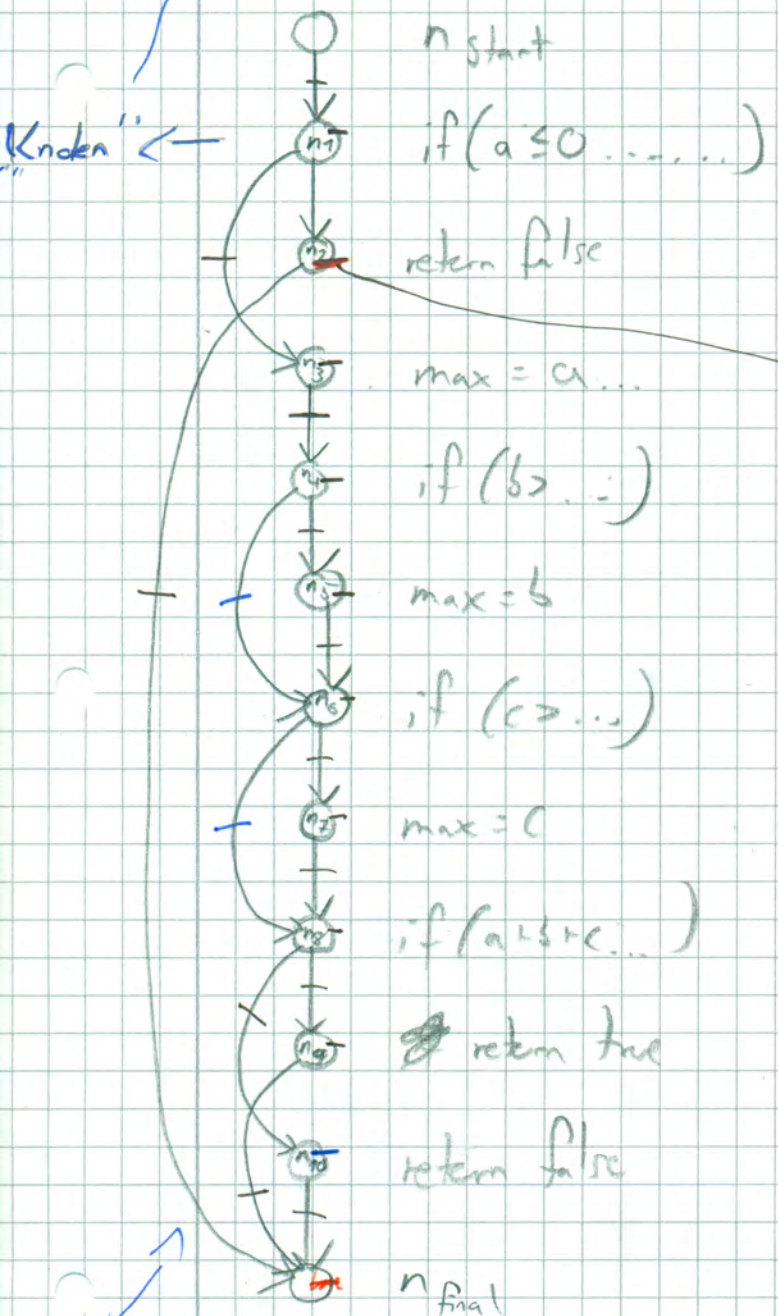
SWE V
26.06.2014

Aufgabe (aus Skript)

Drei Zahlen können die Seiten eines Dreiecks bilden, wenn die Summe der beiden kleinen ....

public boolean proofDreieck .....

Lösung:



"Knoten" ← [Jeder Befehl ein Knoten]

Anweisungsüberdeckung → alle Knoten müssen markiert werden

a = 4 b = 5 c = 6 –

erwartete Rückgabe: true

→ n_2 noch nicht markiert

⇒ a = 0 b = 5 c = 6 –

erwartete Rückgabe: false

→ n_{10} fehlt noch:

a = 2, b = 4, c = 10 –

erwartete Rückgabe: false

⇒ alle Knoten einmal markiert (außer start & final)

⇒ Anweisungsüberdeckung erfolgreich!

alle Zweige müssen markiert werden

Zweigüberdeckung (die selben Testfälle wie Anweisungsüberdeckung)

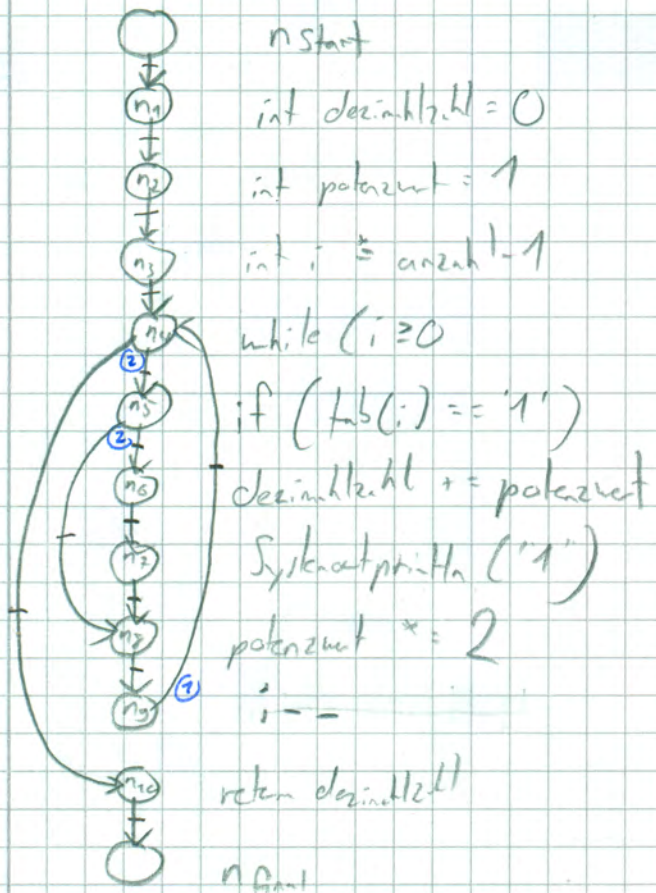
UND: a = 8 b = 5 c = 3 – erw. Rückgabe: false

Aufgabe (aus Skript)

SWE V
27.06.14

Das nachfolgende Programm wandelt eine Binärzahl in eine.....

Lösung: (Kontrollflussgraph)



Hilfe:
- an Ende der Schleife wieder zu Beginn der Schleife (4)
- Fallunterscheidung mit 2 Ausgänge (2)

Zweigüberdeckung: (3)

tab = {'1', '0'} anzahl = 2

erwartete Rückgabe = 2

boundary interior Pfadtest:

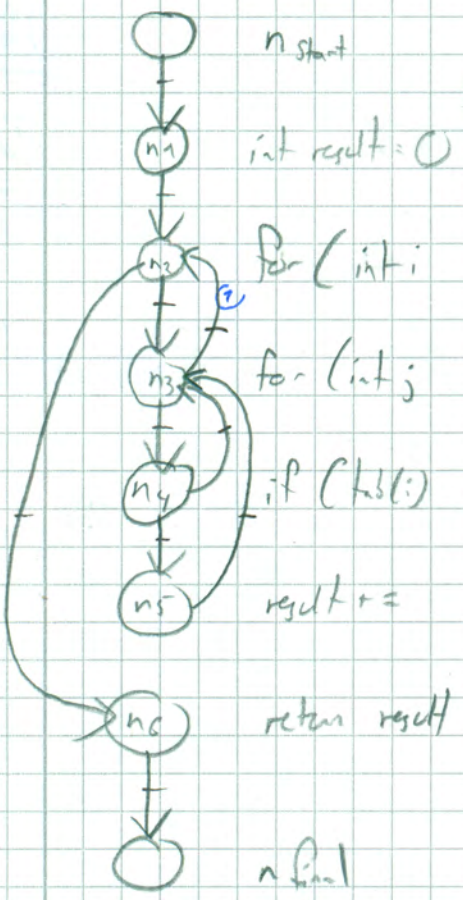
tab	anzahl	erw. Rückgabe
{ } (Schleife nicht durchlaufen)	0	0
{'0'} (Schleife 1x mal durchl.)	1	0
{'1'}	1	1
{'0', '1'}	2	1
{'1', '0'}	2	2
{'1', '1'}	2	3
{'0', '0'}	2	0

(1) innere Fallfälle nehmen, die den Code am weitesten eingerückt durchlaufen

Aufgabe (aus Skript)

Das nachfolgende Programm berechnet die Summe von ....

Lösung:



Wenn innere for-Schleife beendet → zurück zur äußeren

Zweigüberdeckung

tab = {2, 0} , element = 2 , Rückgabe = 4

Boundary interior Pfadtyp:

tab	element	Rückgabe
{ }	0	0
{0}	1	0
{3}	1	9
(1,2)	2	9

{1,0}	2	1
{0,0}	2	0
{0,1}	2	1

03.07.2014

Anwendungsfall Ware kaufen (Sicke Skript)

	Spieleranzahl nichts kaufen	Spieler hat nicht genug Goldstücke	Gewicht zu groß	Spieler will weiter Waren kaufen
1	w	/	/	/
2	f	w	/	w
3	f	w	/	f
4	f	f	w	w
5	f	f	w	f
6	f	f	f	w
7	f	f	f	f

benötigte Testdaten: Spieler hat 1 Goldstück, Esel trägt 99kg

Ware	Preis	Gewicht
Ware 1	1	1
Ware 2	2	1
Ware 3	1	2

~~TP1: Spieler hat nicht genug~~

Testfall 1:

Spieler fragt nach Waren	System zeigt Ware 1, Ware 2, Ware 3
Anwendungsfall Ende.	

Testfall 2:

Spieler fragt nach Waren	System zeigt Waren
Spieler möchte Ware 2 kaufen	Fehlermeldung, nicht genügend Goldstücke
Spieler will Ware 1 kaufen	Erfolgsmeldung, Spieler hat 0 Goldstücke, Esel frisst 100kg

Testfall 3:

Spieler fragt nach Waren	System zeigt Waren
Spieler möchte Ware 2 kaufen	Fehlermeldung, nicht genügend Goldstücke
~~Spieler will~~ Anwendungsfall Ende	

Testfall 4:

Spieler fragt nach Waren	System zeigt Ware 1, Ware 2, Ware 3
Spieler will Ware kaufen	Fehlermeldung Ware zu schwer
Spieler will Ware kaufen	siehe TF2

Testfall 5:

Schritte 1 und 2,3,4 aus Testfall 4 — Anwendungsfall Ende

Testfall 6:

Spieler will Karten kaufen — System zeigt

Spieler will Karte 1 — Erfolgsmeldung

Spieler will Karte 2 kaufen — Fehlermeldung

Testfall 7:

wie TF6 nur nach Erfolgsmeldung -> Anwendungsfall Ende, da Spieler keine Karten mehr kaufen will.

Anwendungsfall: Wire verkaufen

	Spieler will nichts verkaufen	gebotener Preis zu niedrig	Preis für andere Wire	weitere Wire verkaufen
1	w	/	/	/
2	f	w	/	/
3	f	w	w	w
4.	f	w	w	f
5	f	w	f	w

04.07.14

SWE ↓ nicht klausurrelevant

Testfall	Spieler will nicht kaufen	angebotener Preis zu niedrig	Preis für andere Ware frei	weitere Waren kaufen
1	w	/	/	/
2	f	w	f	f
3	f	w	w	w
4	f	w	w	f
5	f	f	f	/
6	f	f	w	f

Bsp. d. Spiels:

Ware	gebotener Preis	Gewicht	Goldstücke 5
Ware 1	1 (zu niedrig)	1	Goldstücke : 5
Ware 2	2 (ok)	2	

TF1:

Spieler fordert Liste der Waren in seinem Besitz an

System zeigt Ware 1 und 2 an

Anwendungsfall Ende

TF2:

Sp fragt ...

Sp fragt nach Preis für Wa 1

Anwendungsfall Ende

System zeigt ...

~~System zeigt~~

System zeigt 1 Geldstück

TF3:

Sp fragt ...

Sp fragt nach Preis für Wa 1

Sp fragt nach Preis für Wa 2

Sp wählt Wa 2

System zeigt ...

System zeigt 1 Geldstück

System zeigt 2 Geldstücke

System zeigt für Fkt 1 Hy, Sp hat 2 Geldstck

Aufgabe: (Skript)

Legen Sie die Reihenfolge fest ... (sortiert nach Wichtigkeit)

- Reisen (1)
- Waren kaufen (2)
- Waren verkaufen (4)
- Tätigkeit annehmen (6)
- essen (5)
- Tag beenden (3)