VU Objektorientierte Modellierung		Übung 3: Sequenzdiagramm
188.391, SS2019	MUSTERLÖSUNG	Übungsgruppen: 08.04.2019 - 11.04.2019

Hinweise: Die meisten Beispiele bestehen aus Modellierungsbeispielen und dazu passenden Theoriefragen.

- Modellierungsbeispiele: Bilden Sie den Sachverhalt, der in der Angabe geschildert wird, möglichst genau ab. Sollte etwas in der Angabe nicht erwähnt sein, treffen Sie sinnvolle Annahmen.
- Theoriefragen: Nehmen Sie sich bei der Beantwortung die Modellierungsaufgaben der jeweiligen Aufgabe zu Hilfe.

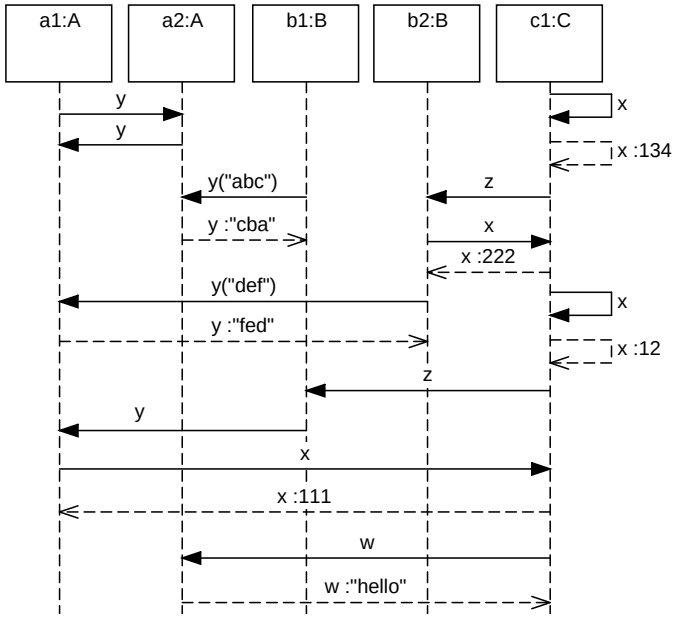
Aufgabe 1: Klassendiagramm aus Sequenzdiagramm

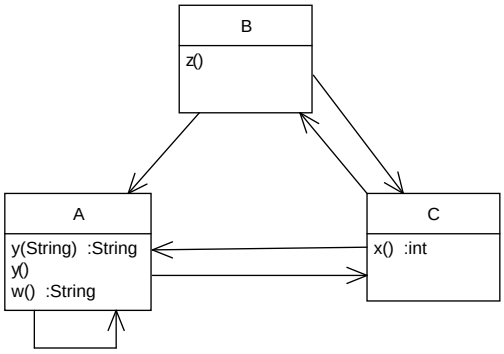
(a) Gegeben ist das nachfolgende Sequenzdiagramm. Modellieren Sie das Klassendiagramm, auf dem das gegebene Sequenzdiagramm basiert.

- Klassen
- 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.

Zusatzfragen:

(b) Wie ist ein Sequenzdiagramm prinzipiell aufgebaut? Welche Elemente kann es enthalten?





Aufgabe 2: Synchrone/Asynchrone Kommunikation

Beschreiben Sie die im folgenden Text vorkommenden Kommunikationsabläufe mittels Sequenzdiagramm. Achten Sie bei diesem Beispiel besonders darauf, ob die beschriebenen Kommunikationsabläufe synchron oder asynchron sind.

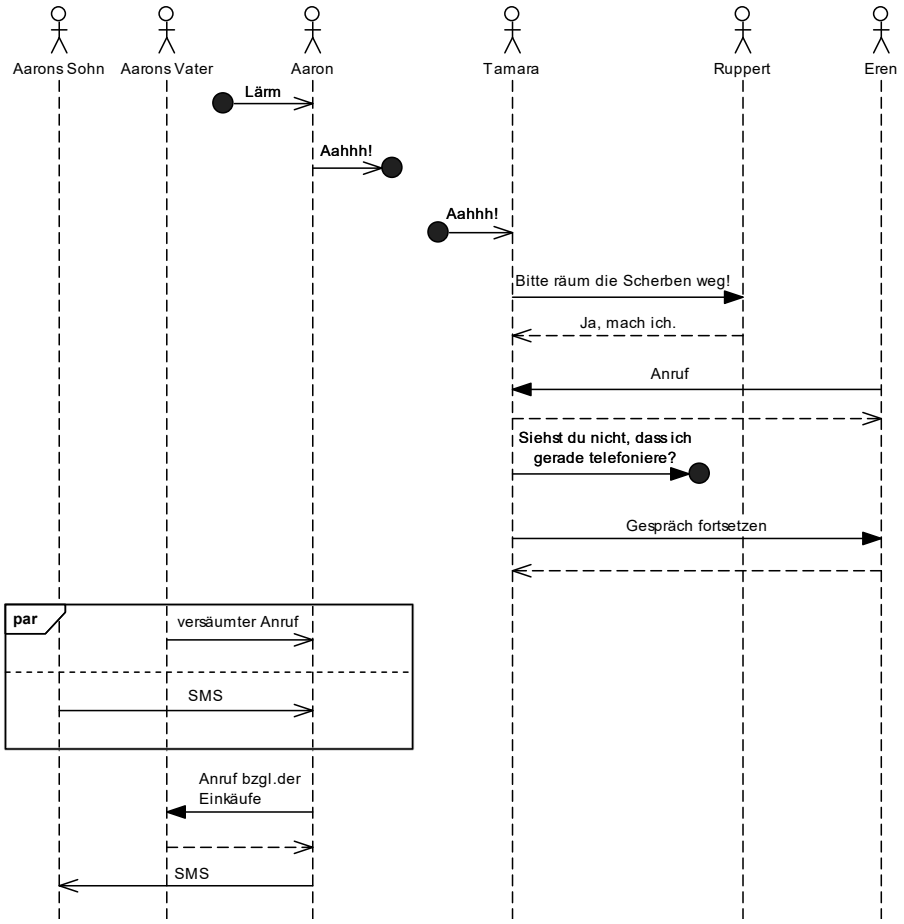
(Teil a) Montagmorgen wird Aaron durch ein lautes Geräusch aus dem Schlaf gerissen. Erschrocken fährt er hoch und beginnt laut zu schreien „Aahhh!“. In der Wohnung neben ihm erschrickt Tamara von dem plötzlichen Lärm und lässt ihre Kaffeetasse fallen. Da gerade Ruppert vom Reinigungsdienst ihre Wohnung putzt, bitte sie ihn, auch den verschütteten Kaffee und die Scherben wegzuräumen. Dieser sagt: „Ja, mach ich!“ und macht sich an die Arbeit.
Kurz darauf läutet ihr Telefon und Eren ist am anderen Ende der Leitung. Sie sprechen über eine dienstliche Angelegenheit. Währenddessen dreht Ruppert den Staubsauger auf. Tamara kann Eren nicht mehr hören und schreit zu Ruppert „Siehst du nicht, dass ich gerade telefoniere?“. Ruppert reagiert nicht und saugt weiter, vermutlich hat er Tamara nicht gehört.

(Teil b) Sie geht ins Nebenzimmer und telefoniert weiter mit Eren. Währenddessen hat sich Aaron von seinem Schock erholt und ist aufgestanden, er nimmt sein Handy und sieht am Display, dass er eine neue SMS von seinem Sohn und einen versäumten Anruf hat. Umgehend ruft er seinen Vater zurück und sie reden über die anstehenden Einkäufe. Aaron verabschiedet sich von seinem Vater und legt auf. Dann schreibt er seinem Sohn eine Nachricht per SMS und macht sich fertig für die Arbeit.

Zusatzfragen:

(c) Was sind die Unterschiede zwischen synchronen und asynchronen Nachrichten?

(d) Welche weiteren Arten von Nachrichten gibt es und wie sehen diese aus?



Aufgabe 3: Berechnung von Traces

(a) Beschreiben Sie alle möglichen Ereignisfolgen des nachfolgenden Diagramms.

(b) Welche der folgenden Traces sind möglich? Warum/warum nicht?

a → b → c → d → e → f → g		☒ nein
a → b → c → d → e → f → g → i → h	☒ ja	
a → c → b → d → e → f → g → i → h	☒ ja	
a → b → d → c → e → f → g → i → h		☒ nein
a → d → b → c → e → f → g → i → h	☒ ja	

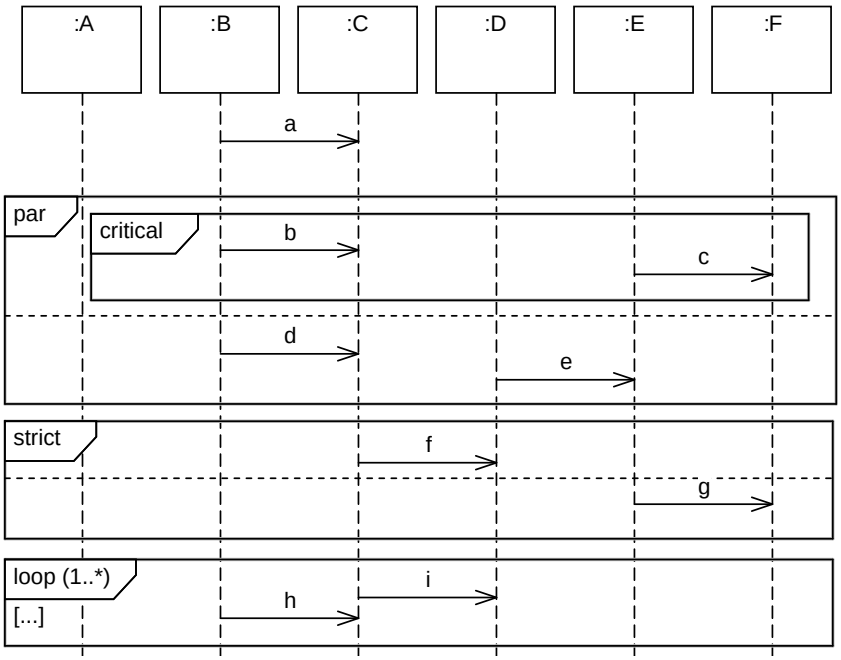
a → b → c → d → e → f → g → i → h → i		☒ nein
a → d → e → b → c → g → f → i → h		☒ nein
a → c → b → e → d → f → g → i → h	☒ ja	
a → e → c → b → d → f → g → i → h	☒ ja	
a → b → c → d → e → f → g → h → i		☒ nein

Zusatzfragen:

(c) Erklären Sie die Funktionsweise der Operatoren seq und strict.

(d) Erklären Sie die Funktionsweise der Operatoren par und critical.

(e) Erklären Sie die Funktionsweise des loop-Oerators.



Die Nachrichten finden nach folgendem Schema statt:

- `a`
- `par`: Die Nachrichten der zwei `par`-Abschnitte können beliebig kombiniert werden, solange die folgenden Einschränkungen berücksichtigt werden:
  - Erster `par`-Abschnitt: `critical`; die Reihenfolge von `b` und `c` ist egal (da sie sich keine Lebenslinie teilen), dazwischen dürfen aber keine anderen Nachrichten kommen.
  - Zweiter `par`-Abschnitt: `seq` (default-Ordnung); die Reihenfolge von `d` und `e` ist egal (da sie sich keine Lebenslinie teilen).

Mögliche Sequenzen zum Beispiel:

`b` → `c` → `d` → `e`

`d` → `c` → `b` → `e`

NICHT möglich zum Beispiel:

`b` → `d` → `c` → `e`

- `strict`: genau diese Reihenfolge muss eingehalten werden

  `f` → `g`

- `loop`: `i` muss jedenfalls vor `h` kommen, da sich die beiden eine Lebenslinie teilen; Wie oft `i` → `h` stattfindet ist abhängig von der Bedingung, jedoch mindestens ein Mal (wegen der Untergrenze `1`).

Aufgabe 4: Kombinierte Fragmente

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

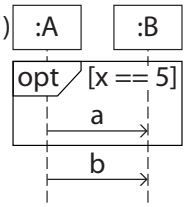
Zusatzfragen:

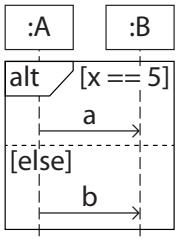
(b) Erklären Sie die Funktionsweise der Operatoren alt und opt.

(c) Erklären Sie die Funktionsweise der Coregion.

(d) Erklären Sie die Funktionsweise des break-Operators.

(a.1)

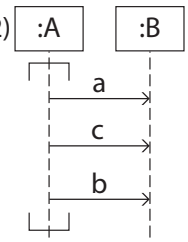


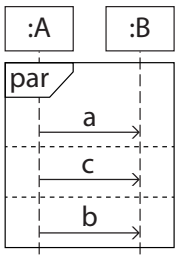


☐ äquivalent

☒ nicht äquivalent

(a.2)

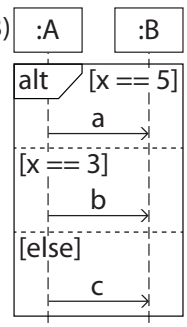


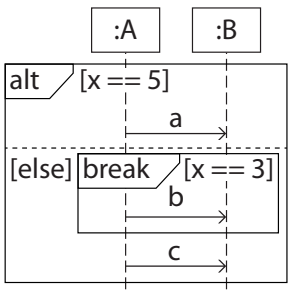


☒ äquivalent

☐ nicht äquivalent

(a.3)





☒ äquivalent

☐ nicht äquivalent

Aufgabe 5: Ausschnitte

Wie können Sie die folgenden Sachverhalten in einem Sequenzdiagramm abbilden? Modellieren Sie die geschilderten Sachverhalte.

(a) Der Client schickt eine Nachricht A zum Server. Dieser muss innerhalb von 30 Sekunden nach Erhalt der Nachricht antworten.

(b) Der Client schickt eine Nachricht B zum Server. Dieser antwortet nach 10 Sekunden.

(c) Der Client schickt Nachricht C um 23:55 zum Server und erwartet keine Antwort.

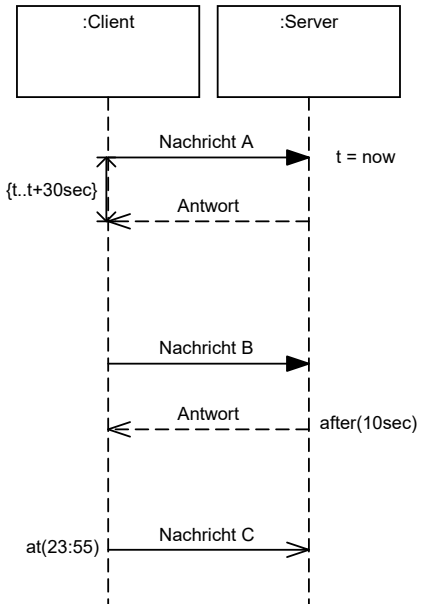
(d) Der Client schickt die asynchronen Nachrichten D, E, F und G zum Server. Nachricht E ist für die Programmiererin nicht relevant, die Nachrichten F und G hingegen sind sehr wichtig.

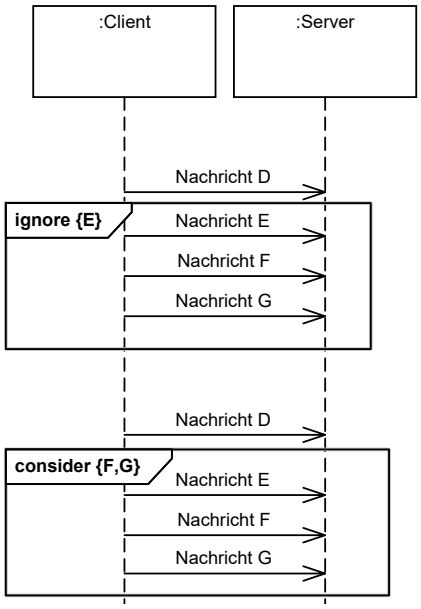
(e) Der Client schickt die Nachricht H, woraufhin der Server antwortet. Anschließend schickt der Client noch die asynchrone Nachricht J. Der Modellierer will sicher gehen, dass die Programmiererin den Ablauf ganz genau so implementiert.

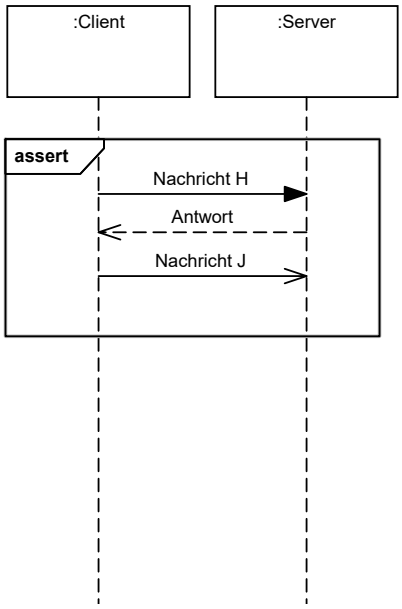
Zusatzfragen:

(f) Welche Konzepte werden benötigt, um die jeweilige Aufgabenstellung zu lösen?

(g) Was versteht man im Kontext des Sequenzdiagramms unter Zustandsinvarianten?





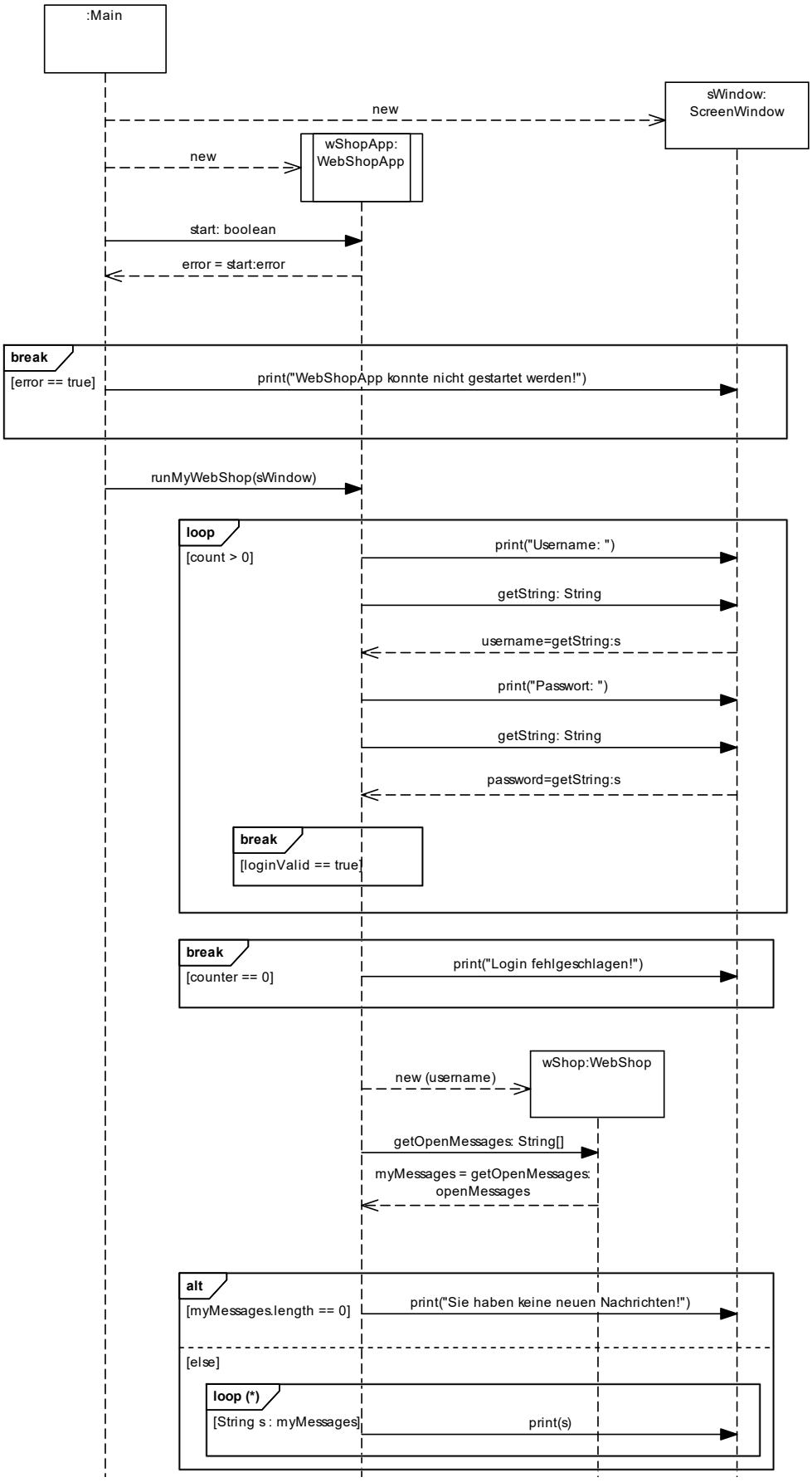


Aufgabe 6: Darstellung von Programmabläufen mittels Sequenzdiagramm

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. „...“ markiert vernachlässigte Codeteile, die nicht modelliert werden müssen.

```
1  class Main {
2    ScreenWindow sWindow = new ScreenWindow();
3    WebShopApp wShopApp = new WebShopApp();
4    boolean error = wShopApp.start();
5
6    if (error) {
7      sWindow.print("WebShopApp konnte nicht gestartet werden!");
8      exit;
9    }
10   wShopApp.runMyWebShop(sWindow);
11   ...
12 }
13
14 class WebShopApp extends Thread {
15
16   public boolean start() {
17     boolean error;
18     ...
19     return error;
20   }
21
22   public void runMyWebShop(ScreenWindow sWindow) {
23     int count = 3;
24     boolean loginValid;
25     ...
26     while (count > 0) {
27       sWindow.print("Username: ");
28       String username = sWindow.getString();
29       sWindow.print("Passwort: ");
30       String password = sWindow.getString();
31       ...
32       if (loginValid) {
33         break;
34       }
35       ...
36     }
37
38     if (count == 0) {
39       sWindow.print("'Login fehlgeschlagen!"');
40       exit;
41     }
42
43     WebShop wShop = new WebShop(username);
44     ...
45     String[] myMessages = wShop.getOpenMessages();
46
47     if (myMessages.length == 0)
48     {
49       sWindow.print("'Sie haben keine neuen Nachrichten!"');
50     }
```

```
51      else
52      {
53        for (String s : myMessages)
54        {
55          sWindow.print(s);
56        }
57      }
58      ...
59    }
60
61  class ScreenWindow {
62    ...
63
64    public void print(String m) {...}
65
66    public String getString() {
67      String s = new String();
68      ...
69      return s;
70    }
71  }
72
73  class WebShop(username) {
74    private String[] openMessages;
75
76    public String[] getOpenMessages() {
77      ...
78      return openMessages;
79    }
80    ...
81  }
```


:Main
sWindow:
ScreenWindow
new
wShopApp:
WebShopApp
new
start: boolean
error = start:error
break
[error == true]
print("WebShopApp konnte nicht gestartet werden!")
runMyWebShop(sWindow)
loop
[count > 0]
print("Username: ")
getString: String
username=getString:s
print("Passwort: ")
getString: String
password=getString:s
break
[loginValid == true]
break
[counter == 0]
print("Login fehlgeschlagen!")
wShop:WebShop
new (username)
getOpenMessages: String[]
myMessages = getOpenMessages:
openMessages
alt
[myMessages.length == 0]
print("Sie haben keine neuen Nachrichten!")
[else]
loop (*)
[String s : myMessages]
print(s)