Einführung in die Programmierung

Prof. Dr. Detlef Krömker
Alexander Wolodkin

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Übungsblatt Nr. 4[1]

Ausgabe: 15.11.2017
Abgabe: 02.12.2017
16:00 Uhr

Personenaufzug

Hinweis: *Dieses Aufgabenblatt ist mit einer Bearbeitungszeit von zwei Wochen dafür ausgelegt in einem Zweierteam gelöst zu werden. Neben der Implementierung wird bei der Bearbeitung Wert auf die folgenden Punkte gelegt: Dokumentation[2], Strukturierung und Einhalten des Style-Guides[3].*

Achtung: Achten Sie darauf die Variable `__author__` in **allen** Quellcode Dateien **korrekt** zu setzen. Abgaben die nicht dieser Vorgabe entsprechen werden **nicht** bewertet! Quellcode muss als *.py* Datei und alles Weitere als *.pdf* Datei abgegeben werden. Bei Abgaben mehrer Dateien müssen diese als *.zip* zusammengefasst werden. Abgaben, die nicht diesen Regeln entsprechen, werden ebenfalls **nicht** bewertet! Außerdem muss ihr Name in jeder abgegebenen *.pdf* Datei zu finden sein.

Σ __ / **20**

Aufgabe 4.1: **Allgemeine Informationen**

Dies ist eine Aufgabe aus dem richtigen Leben, auch wenn sie schon tausende Male mehr oder weniger optimal gelöst wurde.

Simulieren und visualisieren Sie durch ein Python 3.6-Programm einen Zwillings-Personenaufzug (zwei kooperierende Fahrkörbe) für ein 6 stöckiges Haus.
Die `Bezeichnungen der Stockwerke` sind K (unten), E, 1, 2, 3, 4. Auf jedem Stockwerk befinden sich jeweils zwei Anforderungstaster für Auf- und Abwärtsfahrt (`Ausnahmen:` In K nur aufwärts und in 4 nur abwärts) und für jeden Fahrkorb eine Anzeige in welche Richtung er fährt und in welchem Stockwerk er sich befindet. In jedem Fahrkorb gibt es 6 Taster zur Anforderung der anzufahrenden Stockwerke. Gibt man das Stockwerk an, auf dem der Fahrstuhl steht, wird diese Eingabe ignoriert. Die schon gewählten Zielstockwerke werden im Fahrkorb angezeigt. Erreicht der Fahrkorb ein Zielstockwerk, so hält er zum Ein- und Aussteigen 1-3 Takte (gleichverteilt auswürfeln!)an.

Taktsteuerung: Wir realisieren die Fahrstuhlsteuerung taktgesteuert. Der Takt wird durch die Eingabe von <Return> fortgeschaltet. Vor dem Return können entsprechend folgender Kodierung die Anforderungen angegeben werden:

> Fahrt-Anforderung auf jedem Stockwerk **::= ('K' | 'E' | '1' | '2' | '3' | '4') ('h' | 'r'), also immer 2 Zeichen**,
> (Ausnahmen: In 'K' nur 'h' und in 4 'r'), z.B. 'Eh' für Erdgeschoß Anforderung „hoch" oder '3r' für drittes Stockwerk und Fahranforderung „runter"

[1] Es dürfen keine Lösungen aus dem Skript, dem Internet oder anderen Quellen abgeschrieben werden. Diese Quellen dürfen nur mit Quellenangaben verwendet werden und es muss ein hinreichend großer Eigenanteil in den Lösungen deutlich zu erkennen sein.

[2] `http://www.python.org/dev/peps/pep-0257`

[3] `https://www.python.org/dev/peps/pep-0008/`

Fahrt-Anforderung im Fahrkorb A oder B ::= ('A' | 'B')('K' | 'E' | '1' | '2' | '3' | '4'), also immer zwei Zeichen, z.B. 'A1' bedeutet im Fahrkorb A wurde das Fahrziel 1. Stockwerk gedrückt.

In einer Zeile (vor dem <Return>) können mehrere Anforderungen erfolgen, getrennt durch ein Leerzeichen. Jeder Fahrkorb hält in seiner Fahrtrichtung an jeder nächsten Anforderung.

Prioritätsregeln:

(a) Die Fahrt-Anforderungen werden in der Reihenfolge der Angabe bearbeitet, zuerst die am weitesten links stehende und die in der i-ten Eingabe vor der in der i+1. Eingabe.

(b) Die Anforderungen im Fahrkorb haben Vorrang vor den Anforderungen auf den Stockwerken, wenn diese Fahrt-Anforderungen im selben Takt auftreten und werden in jedem Fall bedient.

(c) Fahrkorb A übernimmt die Anforderungen mit höherer Priorität als (also vor) Fahrkorb B.

(d) Wenn ein Fahrkorb keine offenen Anforderungen mehr hat, übernimmt er sofort die nächste Anforderung von Stockwerken.

(e) Jeder Fahrkorb wechselt die Fahrtrichtung nur dann, wenn er mit der aktuellen Fahrtrichtung keine Anforderung mehr erfüllen kann.

Aufgabe 4.2: **Teilaufgaben** Punkte: ____ / 20

(a) (2 Punkte) **Entwickeln Sie** ein Zustandsübergangsdiagramm für einen Fahrkorb: In welchen Zuständen kann er sein? Was sind die Bedingungen für einen Übergang in einen anderen Zustand. Dieses Diagramm soll Sie bei der Entwicklung von Lösungsideen unterstützen. Es muss nicht die letzten Feinheiten oder Bedingungen der endgültigen Lösung enthalten.

Fertigen Sie eine Skizze an: Zustände sind Ellipsen/Ovale. Übergänge sind Pfeile. Schreiben Sie ggf. die Übergangsbedingung an jeden Pfeil. Unbeschriftet heißt: Der Übergang findet beim nächsten Takt statt.

(b) (3 Punkte) **Entwickeln Sie** eine Visualisierung dieser Zustände für beide Fahrkörbe auf der Konsole. Skizzieren Sie diese am besten auf Karopapier: Paper Mock-up!

(c) (2 Punkte) **Entwerfen Sie** eine strukturierte Lösung für dieses Problems. Teilen Sie ihr Programm in mindestens zwei Module auf. Begründen Sie Ihre Wahl. (Wir wissen auch, dass diese Aufteilung nicht unbedingt sein müsste. Aber zur Übung sollen Sie das einmal machen!)

(d) (8 Punkte) **Entwickeln Sie** ein Python 3.6 Programm, dass die Simulation und Visulisierung unter Nutzung der in 3. gewählten Module umsetzt. Lesen Sie die Fahrt-Anforderungen von der Konsole ein und fangen Sie Falscheingaben ab und geben Sie ggf. möglichst helfende Fehlermeldungen an den Benutzer zurück. Geben Sie eine Zustandsvisualisierung in jedem Takt (also nach jedem Return) aus. Sind ggf. weitere Visualisierungen von internen Zuständen sinnvoll?- Welche sind das und welche Lösung haben Sie dafür?

Hinweis: Von den 8 möglichen Punkten entfallen 3 Punkte auf das Einhalten der Programmierrichtlinien!

(e) (3 Punkte) **Geben Sie** geeignete Testdaten (Sequenzen von Fahrtanforderungen) an und begründen Sie deren Wahl!

Lesen Sie den Spiegelartikel „Hoch, runter und nie stecken bleiben“, siehe `http://www.spiegel.de/wissenschaft/mensch/aufzugs-mathematik-hoch-runter-und-nie-st` `html` (oder auch im Reading, da ohne Werbung ;-))).

(f) (2 Punkte) Jetzt wissen Sie, dass Sie ein wirklich schwieriges Problem lösen sollen. **Überlegen und implementieren Sie** (trotzdem) generelle Verbesserungen für die o.g. Prioritätsregeln. Geben Sie an, was die Verbesserungen sind und illustrieren Sie dieseVerbesserung(en) an mindestens einem Beispiel.