Praktikum PSZ WiSe25

Aufgabenblatt 2
(Ausgabe 2025-11-10 für Praktikumstermin 10.11., 24.11. und 8.12.)

Prof. Dr. Bettina Buth <Bettina.Buth@haw-hamburg.de>

Bearbeitungshinweise:

- Die Bearbeitung der Aufgaben findet idealerweise in **festen Dreier- oder Vierergruppen** statt, Einzelbearbeitung ist auch möglich oder Zweierteams.
- Rückfragen zu den Aufgaben bitte über den Kanal Allgemein in MS-Teams TES
- Es gibt **für diese Aufgabe im WiSe 25 prinzipiell Anwesenheitspflicht** beim Praktikum!

Ziel des Praktikums:

- Rust Programmierung vertiefen
- Weiteres Verständnis für Konzepte Speichersicherheit, Typsicherheit und Thread-Sicherheit aufbauen

Vorbereitung vor dem Praktikum

- Rust-Syntax und Verwendung von Nebenläufigkeit recherchieren

Abgabe

- Abgabe der Implementierung in Teams bis 7.12.2025
- Diskussion der Implementierung und der Erfahrungen in der Vorlesung am 8.12.

Zusatzinformationen: Hilfreiche Links

- The Rust Book https://doc.rust-lang.org/book/
- Ev Ältere Version des Rust Books als pdf https://lise-henry.github.io/books/trpl2.pdf

Abgabetermin:
Code in Teams einstellen bis 7.12.2025
Vorstellung der Lösungen und Auswertungen am 8.12.

Rust Coding - Elevator

Aufgabe 1.1: Implementierung

Implementiert in Rust eine Fahrstuhlsimulation zu den folgenden Anforderungen:

F1	Das Fahrstuhlsystem besteht aus 3 Fahrkabinen, die zwischen Ebenen 0 und 3 fahren und beliebig vielen Passagieren
F2	Fahrstühle fahren unabhängig voneinander
F3	Fahraufträge werden durch Knöpfe in der Ebene oder in der Fahrkabine erteilt
F3.1	In den Ebenen können Anforderungen für AUF und AB gewählt werden
F3.2	In den Kabinen können Zielstockwerke gewählt werden
F4	Fahrkabinen zeigen Ihre Position in der Kabine an
F5	Fahrkabinen haben Türen, die in Ebenen geöffnet und geschlossen werden
P1	Passagiere werden auf den Ebenen erzeugt und wählen zufällig AUF oder AB als Fahrauftrag in der Ebene
P2	Passagiere betreten Fahrkabinen wenn sie in der Ebene halten und die Türen in den Zuständen offen, öffnend und schließend sind.
P3	Passagiere wählen zufällig Zielebenen in der Kabine
P4	Passagiere können Fahrkabinen nur bei geöffneter Tür verlassen
Z1	Zustände der Fahrkabinen: Kabine steht in Ebene x, Kabine fährt von Ebene x zu Ebene y, Kabine hält in Ebene x,
Z2	Zustände der Türen: Türen geschlossen, Türen öffnend (in Bewegung von geschlossen zu offen), Türen offen, Türen schließend (in Bewegung von offen zu geschlossen)
Z3	Zustände eines Passagiers: idle auf Ebene x, betritt Fahrkabine, wählt Zielebene, ist in Fahrkabine, verlässt Fahrkabine
C1	Eine zentrale Steuerung gibt Fahraufträge aus der Ebene an Fahrkabinen und überwacht die Bewegungen der Fahrkabinen und Fahraufträge
C2	Die zentrale Steuerung steuert die Fahrkabinen zu Ebenen entsprechend der vorliegenden Aufträge
C3	Die zentrale Steuerung steuert das Öffnen und Schließen der Türen
C4	Die zentrale Steuerung überwacht die Anzahl der Passagiere pro Kabine
S1	Türen von Kabinen sind während der Fahrt nicht offen
S2	Kabinen die auf Ebene 0 stehen können nicht abwärts fahren
S3	Kabinen, die auf Ebene 3 stehen, können nicht aufwärts fahren

S4	Türen von Kabinen, die in Ebene x stehen, schließen sich nach Timeout
S5	Es dürfen nicht mehr als 2 Passagiere gleichzeitig transportiert werden.
S6	Wenn zuviele Passagiere in einer Fahrkabine sind, wird das geeignet angezeigt und die Türen schließen erst wenn die maximale Anzahl an Passagieren nicht überschritten ist
S7	Wenn beim Schließen der Türen Passagiere oder andere Hindernisse in der Tür entdeckt werden, dann öffnen sich die Türen wieder
N1	Bei der Implementierung sollen die Hauptkomponenten in separaten Threads realisiert werden und weitgehend unabhängig von einander agieren Bsp: wenn sich die Türen in Kabine 1 schließen kann Kabine 2 trotzdem fahren

Ergänzt und verfeinert die Anforderungen um geeignete Anforderungen insbesondere für Zuverlässigkeit und Safety 😊

Aufgabe 1.2: Evaluation

Notiert Eure Beobachtungen bei der Implementierung, z.B.

- Welche Frameworks wurden verwendet; ev auch welche Pattern?
- Welche Probleme gibt es bei der Umsetzung von Datenstrukturen und Kontrollfluss in Rust?
- Wo werden Typsicherheit, Speichersicherheit und Thread-Sicherheit bei der Implementierung sichtbar?
- …

Viel Spaß!