
Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing-Wiss. und Informatik
- Software-Entwicklung -

Theoretische Informatik

Sommersemester 2025
1. Aufgabenblatt

Vorbemerkungen:

Nutzen Sie die unter C:\kleukersSEU von Ihnen installierte Software, siehe auch http://kleuker.iui.hs-osnabrueck.de/kleukersSEU/index.html. Sollten Sie in einer anderen Veranstaltung Eclipse nutzen, verwenden Sie zumindest einen anderen Workspace.
Weitere Hinweise zur Nutzung von Eclipse befinden sich auf der Web-Seite der Lehrveranstaltung http://kleuker.iui.hs-osnabrueck.de/querschnittlich/SEU.pdf

Generell hat jedes Aufgabenblatt einen theoretischen Teil und mehrfach einem praktischen Teil. Der theoretische Teil hat u. a. den Anspruch auf die Klausur vorzubereiten. Der praktische Teil beschäftigt sich mit der Nutzung von Werkzeugen und der Umsetzung der Definitionen und Algorithmen der Vorlesung in Java, wodurch ein vertieftes Verständnis aufgebaut werden kann. Der theoretische Anteil sollte als verpflichtend angesehen werden. Praktische Teile, die auf eigene Gefahr bei Überlastung weggelassen werden könnten, sind mit 👍 gekennzeichnet („Daumen hoch für ihr Engagement bei diesen Aufgaben.“).

Aufgabe 0.1
Geben Sie das Lösungswort des Quiz aus der Lernnotiz an.

Aufgabe 1 (Grundbegriffe, Abzählbarkeit, Sprachen, Mathematik versus Informatik)

a) Gegeben seien folgende Mengen M1= {}, M2={a,b,c}, M3={5,10}. Berechnen Sie
   i) M2 ∪ M1
   ii) M3 ∩ M3
   iii) M3 – M2
   iv) M1 × M2
   v) M2 × M3
   vi) (M2 × M3) ∩ (M3 × M2)
   vii) Pot(M2) // Potenzmenge von M2

b) Begründen Sie formal, warum die Menge {1, 2, 3, 4} abzählbar ist.

c) Begründen Sie formal, warum die Menge
   NatürlicheZahlen × NatürlicheZahlen × NatürlicheZahlen abzählbar ist.

d) Sind folgende Mengen Alphabete?

	ist Alphabet	ist kein Alphabet
{a,b,c}		
{a,a,b,c,c}		
{}		
{ε}		
alle ganzen Zahlen des Intervalls [-42,42]		
alle rationalen Zahlen des Intervalls [-42,42]		
alle reellen Zahlen des Intervalls [0,1]		
{x | x∈ NatuerlicheZahlen mit x+2 < 2}		

e) Gegeben seien die folgenden 4 Worte über dem Alphabet {a,b,c}, geben Sie jeweils die Konkatenation der folgenden Worte untereinander an.

°	ε	a	aa	bab
ε				
a				
aa				
bab				

f) Gegeben seien die folgenden Sprachen über dem Alphabet {a,b}, welche der Sprachen enthält das Wort a^{42}?
   i) Sprache1={aaaa}
   ii) Sprache2 = {aaa}
   iii) Sprache1*
   iv) Sprache2*
   v) (Sprache1Sprache2)*
   vi) (Sprache1Sprache1)*

g) Gegeben sei folgende Java-Klasse. Begründen Sie warum next() in Java aus Sicht der Mathematik nicht Funktion genannt wird. Überlegen Sie, wie trotzdem der Begriff Funktion für next passend sein könnte. Überlegen Sie jeweils welche Definitionsbereiche und Werte- (oder Ergebniss-)bereiche eine Rolle spielen könnten.

```
public class Delta {
  private int delta;

  public int next(int wert) {
    return wert + (delta++);
  }
}
```

Aufgabe 2 (Umsetzung informeller Definitionen)
Mit den Implementierungsaufgaben können Sie im Detail überprüfen, ob Sie die Definitionen und Algorithmen im Detail verstanden haben.
Konzept: Die Modellierung soll einem domain-orientierten Ansatz folgen, d. h. jeder Fachbegriff wird als Klasse (oder Methode) repräsentiert, elementare Typen nur auf unterster Modellierungsebene genutzt. Generell hat jede Klasse zumindest eine equals(.)- und eine toString()-Methode.

Anforderungen:
Zu entwickeln ist eine Klasse Zeichen zur Repräsentation von Zeichen. Dabei kann ein Zeichen sich auch aus mehreren klassischen Buchstaben zusammensetzen, z. B. „a“ aber auch „blubb“. Es ist garantiert, dass es jedes Zeichen nur einmal gibt, d. h. es kann aus einem String ein Zeichen erzeugt werden, sollte es aber schon ein Zeichen zu diesem String geben, muss das identische (==) Zeichen zurückgegeben werden.
Ein Leerzeichen soll als Zeichen möglich sein, dabei soll “ „ und „/space“ zu einem identischen Objekt gehören.
Für neue Berechnungen sollen alle bisher eingerichteten Zeichen gelöscht werden können.
Bei der Erstellung von Zeichen wird darauf geachtet, dass die zugehörigen Strings kein Präfix von einander sind, d.h. es darf nicht zwei Zeichen „xa“ und „xab“ geben, was zu einer Exception führen soll. (dies steht nicht in der formalen Definition, macht aber viele spätere Aufgaben deutlich einfacher).

Zu entwickeln ist eine Klasse Wort, dabei besteht ein Wort aus einer endlichen Folge von Zeichen. Zur Erstellung eines Wortes wird eine Folge von Zeichen oder ein String übergeben. Dazu wird eine Methode benötigt, die einen String in die zugehörigen Zeichen umrechnet. Sollte dabei ein Zeichen nicht existieren, wird eine Exception geworfen.
Für Worte sind mehrere Methoden zur Bearbeitung zu entwickeln, diese umfassen die Berechnung der Länge, des Zeichens an Position i, des Teilworts ab der Position i bis zur Position j, die Ersetzung eines Zeichens an der Position i durch ein Wort, das vorne und hinten Dranhängen eines Zeichens, das Löschen eines Zeichens an der Position i, die Berechnung

aller Positionen eines Zeichens in einem Wort, eine Ausgabe, die das leere Wort als Epsilon (`"\u03B5"`) ausgibt, eine Ausgabe, die Leerzeichen als /space ausgibt und die Möglichkeit über die Zeichen des Wortes zu iterieren.

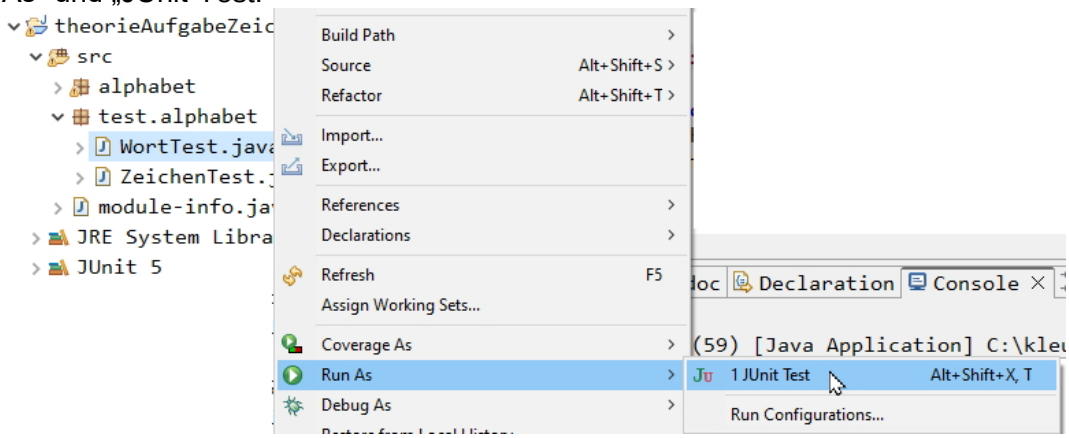
Umsetzung:
Laden Sie das zugehörige Eclipse-Projekt theorieAufgabeZeichenWort herunter, das Tests zu den zu entwickelnden Klassen und Spezifikationen der zu implementierenden Methoden enthält. Programmieren Sie die dort angegebenen Methoden so aus, dass die Anforderungen und Tests erfüllt sind.

Um Zeichen eindeutig umzusetzen, sollen diese durch in einer Klassenmethode der Klasse Zeichen erzeugt werden. Ein Zeichen wird wie folgt konstruiert. Damit wird der eigentliche Konstruktor innerhalb der Methode zeichen(.) aufgerufen. Es ist dabei sinnvoll, dass der Konstruktor der Klasse Zeichen die Sichtbarkeit private hat (warum?).

```
Zeichen z = Zeichen.zeichen("x");
```

Die Klasse Zeichen muss dann alle bereits erzeugten Zeichen verwalten können. Eine dabei sehr praktische Klasse ist HashMap<Key,Value>, mit er zu Key-Werten sich genau ein Value-Wert gemerkt werden kann. Im Beispiel ist damit HashMap<String,Zeichen> interessant.
Ein Wort ist generell eine Folge von Zeichen, die in Java z. B. durch die Klasse ArrayList<Zeichen> sinnvoll repräsentiert werden kann. Beachten Sie, wenn Sie eine String in ein Wort umwandeln wollen, dass die einzelnen Zeichen kein gemeinsamen Präfix haben und so schrittweise geprüft werden kann, ob ein Teilstring für ein Zeichen steht.

Generell geben die Tests weitere Hinweise zur gewünschten Funktionsweise.
Zum Ausführen der Test machen Sie z. B. ein Rechtsklick auf der Datei und nutzen dann „Run As“ und „JUnit-Test.



Sollen zu einem gescheiterten Test Detailinformationen angezeigt werden, wird auf den kleinen Kasten, dritter von rechts geklickt. Die Detailinformationen stehen danach im Ausgabefenster.