

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 1 (Abzählbarkeit, Mathematik versus Informatik)

- Begründen Sie formal, warum die Menge $\{1, 2, 3, 4\}$ abzählbar ist.
- Begründen Sie formal, warum die Menge $\text{NatürlicheZahlen} \times \text{NatürlicheZahlen} \times \text{NatürlicheZahlen}$ abzählbar ist.
- Gegeben sei folgende Java-Klasse. Begründen Sie warum `next()` in Java nicht Funktion genannt wird. Überlegen Sie, wie trotzdem der Begriff Funktion für `next` passend sein könnte.

```
public class Zaehler {
    private static int anzahl = 1;
    public int wert;

    private Zaehler(int wert) {
        this.wert = wert;
    }

    public Zaehler next() {
        return new Zaehler(anzahl++);
    }
}
```

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.

Laden Sie von der Veranstaltungsseite das zugehörige Eclipse-Projekt `theorieZeichenWortAufgabe` herunter, das Tests zu den zu entwickelnden Klassen und Spezifikationen der zu implementierenden Methoden enthält.

Generell geben die Tests weitere Hinweise zur gewünschten Funktionsweise.