Hierbei handelt es sich weder um eine Veröffentlichung der Übungsleitung noch des Lehrstuhls für Software Engineering.
Dieses Dokument ist ein inoffizielles Übungsblatt für Studierende der Gruppe 17. Obwohl ich mich um Korrektheit bemühe, kann ich keine absolute Fehlerfreiheit garantieren. Wem ein Fehler auffällt bitte kurzen Hinweis per Mail an m.schwarz@tum.de.

Klassendiagramme, Generics und Listen

Die Universität möchte die Stühle, die es gibt, modellieren. Hierzu soll es zuerst einmal eine Klasse Stuhl geben, die nicht initialisiert werden darf. Spezialisierte Stühle sind Hocker und Drehstühle, diese Klassen sollen initialisierbar sein. Außerdem gibt es bestimmte Drehstühle, die auch höhenverstellbar sind. Diese implementieren das Interface Hoehenverstellbar.
Jeder Stuhl hat eine Nummer, sowie einen zugeordenten Raumnamen. Drehstühle verfügen zusätzlich über die Methode *public void drehen()*, höhenverstellbare Stühle über die Methode *public void verstelleHoehe(int i)*.

1.) Diagramm

Geben Sie ein auf diesen Sachverhalt passendes Klassendiagramm nach den Regeln der Vorlesung an.

2.) Generics und Listen

Nachdem die Universität nicht nur Stühle, sondern auch Mitarbeiter, Studierende, Bücher usw. verwaltet, soll eine Liste implementiert werden, bei der erst später festgelegt werden muss, was darin verwaltet wird. Es soll aber sichergestellt werden, dass z.B. in einer Liste von Büchern keine Drehstühle eingefügt werden. Die Klasse soll Liste heißen und folgende Methoden haben:

public boolean istLeer()	true $\Leftrightarrow$ Liste leer.
public T get(int i)	Gibt das Element an Stelle i zurück; Wenn es kein solches Element gibt, wird eine IllegalArgumentException geworfen.
public void hEinfuegen(T element)	Fügt das Element hinten in der Liste ein.
public int laenge()	Gibt die Länge der Liste zurück,

Beachten Sie: Die Liste soll beliebig erweiterbar sein.

a) Geben Sie eine mögliche Implementierung an.
b) Intialisieren Sie nun jeweils Listen für Stühle, Drehstühle, höhenverstellbare Stühle und eine Liste, in der Objekte jedes beliebigen Typs verwaltet werden können.

Kontrollflussdiagramme und MiniJVM

Listing 1: Ein Algorithmus

```
1  public int func(int a,int b)
2  {
3      int i=0;
4      while(a >= b)
5      {
6          a = a-b;
7          i = i+1;
8      }
9
10     return i;
11 }
```

a) Geben Sie ein Kontrollflussdiagramm an.
b) Geben Sie eine Übersetzung dieses Programms in Code für die MiniJVM an. Gehen Sie davon aus, dass a und b in Zellen 0 bzw. 1 gespeichert sei. Die Rückgabe erfolge durch Speichern nach Speicherzelle 3.
c) Was macht der Algorithmus ?