Lehrstuhl für Softwaretechnik und Programmiersprachen
Professor Dr. Michael Leuschel

# Compilerbau - Wintersemester 2021/22

## Praktisches Übungsblatt 2

Besprechung der Aufgaben am 05.11.21 um 16:30 Uhr in 25.12.02.55 und gleichzeitig online per BBB
(evtl. auch früher)
Fragen an Lukas.Lang@hhu.de
Die Bearbeitung ist freiwillig.
Bringen Sie Ihre Lösung mit zum Übungstermin

### Aufgabe 2.1

Implementieren Sie in Java einen deterministischen endlichen Automaten. Sie erhalten einen JUnit Testcase für den Automaten, der eine Binärzahl akzeptiert, die durch 5 teilbar ist.

1. Implementieren Sie dazu eine Klasse Node, die einen Zustand des Automaten repräsentiert und das folgende Interface implementiert:

   ```
   public interface INode {
   public abstract void addTransition(Node dest, char c);
   public abstract INode getNext(char c);
   public abstract String getName();
   public abstract boolean isFinal();
   public Iterable<? extends IEdge> getEdges();
   }
   ```

   Beachten Sie, dass die Methode getNext(char c) null zurückliefern soll genau dann, wenn es keine Transition für c gibt.

2. Schreiben Sie eine Klasse DFA, in der der Automat kodiert ist, sowie eine Klasse Edge welche die Kanten modelliert.

   ```
   public interface IEdge {
       public INode getStart();
       public INode getEnd();
       public char getChar();
   }
   ```

   Die Kodierung des Automaten soll im Konstruktor der Klasse erfolgen. Dem Konstruktor der Klasse wird hierzu eine Menge von Knoten und Kanten sowie ein Startknoten übergeben.

```
import java.util.Set;

public class DFA {
    private Set<INode> nodes;
    private Set<IEdge> edges;
    private Node start;

    public DFA(Set<INode> nodes, INode start, Set<IEdge> edges){
       // TODO:
    }

    public INode getStart(){
       // TODO:
    }

    public String accept(String word, INode current) {
       // TODO:
    }

    public Iterable<? extends INode> getNodes() {
       // TODO:
    }
}
```

3. Die Klasse soll eine Methode accept( String word) haben. In dieser Methode werden der Reihe nach die besuchten Knoten ausgegeben und anschließend entweder ablehnen oder akzeptieren. Die gesamte Ausgabe muss in einer Zeile stehen und zwischen der Knotenliste und dem Ergebnis muss ein Leerzeichen sein. Im obigen Beispielautomaten:
   Die Eingabe „10010“ liefert den String:

   ```
   "CBEDDA ablehnen"
   ```

   Die Eingabe „11001“ liefert den String:

   ```
   "CBABEC akzeptieren"
   ```

4. **Hinweis:** Die Methoden 'Iterable<? extends IEdge> getEdges();' in Node und 'Iterable<? extends INode> getNodes()' in DFA müssen Sie nur implementieren, wenn Sie die mitgelieferte Klasse ViewUtil zur Visualisierung ihrer Automaten benutzen möchten!