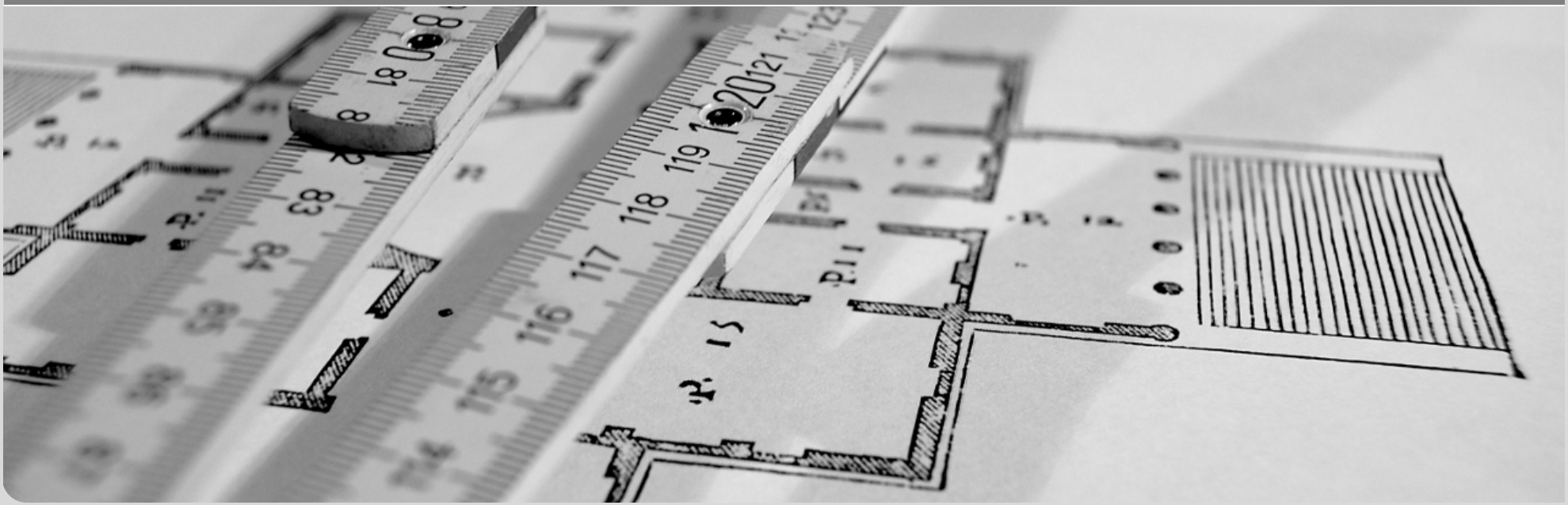
Karlsruher Institut für Technologie

SDQ Software Design and Quality

Programmieren
11. Tutorium

Robin Rüde | 2. Februar 2015



Gliederung

1

Organisatorisches

Organisatorisches

- Übungsschein bestanden: 59 oder mehr (evtl auch weniger).
- Vergangene private Tests werden demnächst öffentlich gemacht.
- Musterlösungen kommen immernoch... demnächst
- Private Tests von Blatt6A wurden geändert, erlauben jetzt beliebige Reihenfolge

2

Parser

Parser

- Top-Down-Parser
- Bestimme Grammatik dessen, was geparst werden soll
- Links-Faktorisierung der Grammatik durchführen (als erstes immer ein Terminal)
- nun kann mit einem lookahead Symbol die nächste Regel bestimmt werden.
- Danach Aufgabe: JSON-Parser schreiben

Beispielgrammatik:

```
Document → Element
Element → Array|String
Array → '[' Element (',' Element)* ']'
String → '"' (a-z)* '"'

Beispiel: ["abc", "a", ["b", "c"], ["a", ["b"]], "d"]
```

Parser: Beispiel

üblicher Parseraufbau

- peek() - Funktion: gibt nächstes Zeichen ohne es zu lesen. (lookahead)
- read() - Funktion: liest nächstes Zeichen und erhöht Eingabeposition.
- expect(char) - Funktion: if(read()!=char) throw new Exception();
- parseDocument()-Funktion die als erstes aufgerufen wird.
- für jede Art von möglichen Element eine parseX()-Funktion
- jede parseX()-Funktion ruft rekursiv weitere auf.
- Rückgabewerte:
  - Validieren: boolean oder void
  - Parsen: Objekte (Abstrakter Syntax-Baum)

Parser: Beispiel

Karlsruher Institut für Technologie

Document → Element

```
void parseDocument() {
    parseElement();
}
```

Element → Array|String

```
boolean parseElement() {
    switch(peek()) {
        case '[': parseArray(); break;
        case '"': parseString(); break;
    }
}
```

Parser: Beispiel

```
Array → '[' Element (',' Element)* ']'
<=>

Array → '[' Elements ']'
Elements → Element ',' Elements
Elements → Element
```

```
void parseArray() {
    expect('[');
    parseElement();
    while(peek() == ',') parseElement();
    expect(']');
}
```

```
String → '"' (a-z)* '"'
```

```
parseString() {
    expect('"');
    while(peek() != '"') read();
    expect('"');
}
```

Parser: Beispiel

```
class Parser {
    void parseDocument() { return parseElement(); }
    void parseElement() {
        switch(peek()) {
            case '[': parseArray(); break;
            case '"': parseString(); break;
        }
    }
    void parseArray() {
        expect('[');
        parseElement();
        while(peek() == ',') parseElement();
        expect(']');
    }
    void parseString() {
        expect('"');
        while(peek() != '"') read();
    }
}
```

Aufgabe

Schreibe einen Parser für JSON. JSON (vereinfacht):

```
Document → Element
Element → Array|Object|String
Array → [ Element (, Element)* ]
Object → { Attribute (, Attribute)* }
Attribute → String : Element
String → " (char)* "
```

Beispiel:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

3

Suchen

Lineare Suche

- Gegeben: Array a mit n Elementen
- Gesucht: Ist x in a? Wo ist x in a?
- Idee: einfach einmal Array durchlaufen

Binäre Suche

- Gegeben: Array a mit n Elementen. a sortiert
- Gesucht: Ist x in a? Wo ist x in a?
- Idee: Sortierung macht das finden einfach. Beginne in der Mitte.
- Prüfe, ob das gesuchte Element kleiner oder größer ist
- Kleiner -> suche in der linken Hälfte weiter
- Größer -> suche in der rechten Hälfte weiter
- Gleich: gebe das gewünschte zurück

4

Sortieren

Bubblesort

- Array besteht aus 2 Teilen
- Links unsortiert, Rechts sortiert
- Wenn das aktuelle Element größer als das folgende ist: vertausche diese
- Bis man am Ende des Arrays ist. Das größte Element steht nun rechts.
- Beginne wieder von vorn

SelectionSort

- Links sortiert, rechts unsortierter Teil des Arrays
- Wähle aus dem rechten Teil das kleinste Element aus
- schiebe dieses an den Anfang des rechten, unsortierten Teils
- das ausgewählte Element ist nun an der richtigen Stelle

InsertionSort

- Links sortiert, rechts unsortiert
- Wähle das erste Element des rechten Teils.
- Sortiere dies in den linken sortierten Teil ein

Weitere (bessere) Algorithmen: Quicksort, Mergesort

Ende

Vielen Dank für eure Aufmerksamkeit!
Fragen?