Übungsblatt 5: XML-Schema, XPATH & SVG

Abgabetermin: 02.6.2019, 23:59 Uhr

Aufgabe 1: Sokoban – XML und XSD

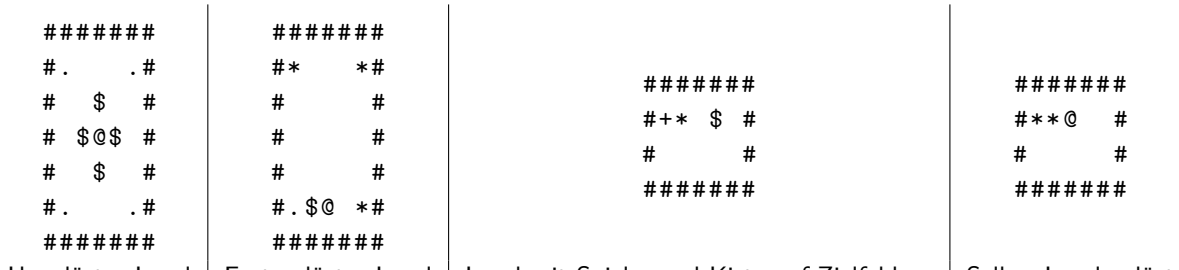
$(2 + 1 + 1.5 + 1)$

In dieser Aufgabe sollst du ein XML-Schema (XSD) für Sokoban Levels implementieren.

Das meistgenutzte Format für die Darstellung von Sokoban Levels ist ASCII-Text. Dabei werden die Felder folgendermaßen kodiert (Quelle: `http://sokobano.de/wiki/index.php?title=Level_format` (Mai 2019)):

Wand	`#`
Zielfeld	`.`
Spieler	`@`
Spieler auf Zielfeld	`+`
Kiste	`$`
Kiste auf Zielfeld	`*`
Freies Feld	*Leerzeichen*

Hier sind Beispiele, wie Levels aussehen können:



Ungelöstes Level	Fast gelöstes Level	Level mit Spieler und Kiste auf Zielfeldern	Selbes Level gelöst

Dieses Darstellungsweise ist sehr simpel und erlaubt es nicht zusätzliche Daten anzugeben. Für das Sokoban-Projekt soll ein XML-Format verwendet werden, das die folgenden zusätzlichen Daten darstellen kann:

- Namen von beliebig vielen Autoren
- Name des Levels
- Schwierigkeitsgrad (kann folgende Werte haben: `EASY`, `MEDIUM`, `HARD` und `IMPOSSIBLE`)
- Größe des Levels (Breite und Höhe)
- Level im obigen ASCII-Text Format

Hier ist ein Beispiel (kann im Moodle heruntergeladen werden):

```
<?xml version="1.0" encoding="UTF-8"?>
<sokoban:Sokoban xmlns:sokoban="https://pvs.informatik.uni-ulm.de/sokoban">
  <Author>Stefan Koegel</Author>
  <LevelName>PvS Test Level</LevelName>
  <Difficulty>EASY</Difficulty>
  <LevelData width="7" height="7">#######
#.    .#
#  $  #
# $@$ #
#  $  #
#.    .#
#######</LevelData>
</sokoban:Sokoban>
```

Im folgenden ist eine lückenhafte XSD angegeben (kann auch im Moodle heruntergeladen werden), die die obigen Anforderungen an das Format erfüllen und das Beispiel validieren soll.

Fülle die Lücken in der XSD und verwende die Seite http://www.utilities-online.info/xsdvalidation/ um zu überprüfen ob deine XSD zum obigen Beispiel XML passt.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="https://pvs.informatik.uni-ulm.de/sokoban"
           xmlns="https://pvs.informatik.uni-ulm.de/sokoban"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="difficultyEnum">
    <xs:restriction base="xs:string">
      <!--Hier fehlt die Enumeration für die Schwierigkeit-->
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="sokobanString">
    <xs:restriction base="xs:string">
      <xs:pattern
        value="<!--Hier fehlt das Pattern für die in einem Level erlaubten Zeichen-->"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="Sokoban">
    <xs:complexType>
      <xs:sequence>
        <!--Hier fehlen die Element für Autor, Level Name und Schwierigkeit-->
        <xs:element name="LevelData">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="sokobanString">
                <!--Hier fehlen die Attribute für Breite und Höhe eines Levels-->
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
```

```
  </xs:element>
</xs:schema>
```

Aufgabe 2: Sokoban – XML einlesen (5 + 1 + 1 + 3 + 1)

In dieser Aufgabe sollst du Sokoban Levels aus XML Dateien einlesen.

Hinweis: Den Code aus der Vorlesung zum Thema XML findest du im Moodle.

1. Erzeuge in deinem Sokoban Package eine Klasse `SokobanLevel`. Die Klasse soll einen Konstruktor haben, der einen Pfad (als `String`) zu einer XML Datei nimmt, das XML einliest und damit die Felder der Klasse befüllt. Entsprechend braucht die Klasse eine Liste für die Namen der Autoren, ein Feld für den Namen des Levels, ein Feld für den Schwierigkeitsgrad und ein `char[][]`-Array für das eigentliche Level.
2. Achte darauf, dass du `Exceptions` sinnvoll behandelst. Zum Beispiel: wenn der Pfad falsch ist oder das XML nicht der XSD entspricht
3. Überprüfe ob der Sokoban Level den Größenangaben in der XML Datei entspricht. Wirf eine passende `Exception` falls dies nicht der Fall ist.
4. Übernimm den Code aus deiner Sokoban.java Klasse und ergänze ihn um mit dem erweiterten Format spielen zu können.
5. Füge in deiner Consolenanwendung die Möglichkeit hinzu ein Level aus einem angegebenen Dateipfad zu laden.

Aufgabe 3: XML Validität (10 {2 pro erkannten Fehler mit Erklärung, 1 für valide erkannt})

Lade dir die Beispiel XML-Dateien aus dem Moodle herunter (`Sokoban-A.xml` bis `Sokoban-F.xml`).

Drei dieser XMLs entsprechen nicht den Anforderungen aus der ersten Aufgabe und sollten von deiner XSD als falsch erkannt werden. Welche sind das? Wieso sind sie falsch?

Eine dieser XMLs ist valide zur obigen XSD, sollte aber von deiner Klasse `SokobanLevel` als fehlerhaft erkannt werden. Welche ist das? Wieso ist sie falsch? Könnte die XSD diese Fehler auch erkennen?

Welche XML Dateien sind nicht Fehlerhaft?

Aufgabe 4: XSD & DTD (3 {6x0.5})

DTD ist eine Alternative zu XSD um XML-Dokumente zu validieren.

Nenne drei Unterschiede zwischen DTD und XSD. *Hinweis: Ist eines der Formate ausdrucksmächtiger als das andere?*

Aufgabe 5: SVG und XPATH (4)

SVG ist ein Bildformat, das auf XML basiert. Kreise[1] und Linien[2] werden als XML-Elemente gespeichert. Farben, Liniendicke und andere Eigenschaften werden als Attribute der XML-Elemente gespeichert. Sieh dir die Datei `picture.svg` aus Moodle, in einem Texteditor und einem Programm das SVG-Bilder darstellen kann (z.B. Firefox), an.

Da SVG-Biler im Grunde XML-Dateien sind, kannst du sie, wie in der ersten Aufgabe dieses Übungsblattes, einlesen und wieder abspeichern.

In dieser Aufgabe sollst du die Datei `picture.svg` (Abbildung 1 zeigt das Originale und veränderte Bild) einlesen, mithilfe von `XPATH` verändern und als `modfiedPicture.svg` wieder abspeichern. Sieh dir dazu `javax.xml.xpath.XPathFactory`

[1] https://developer.mozilla.org/en-US/docs/Web/SVG/Element/circle
[2] https://developer.mozilla.org/en-US/docs/Web/SVG/Element/line

und `javax.xml.xpath.XPathExpression`, die Vorlesungsfolien und die Code-Beispiele aus der Vorlesung (können im Moodle heruntergeladen werden) an.

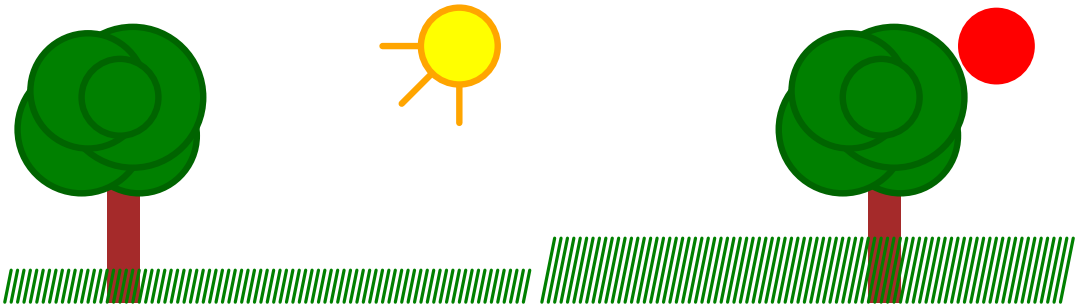


Abbildung 1: `picture.svg` aus Moodle und `modifiedPicture.svg` Bild.

- Erzeuge eine `XPathExpression`, welche die drei *Sonnenstrahlen* im Bild findet und lösche diese dann. *Hinweis: Sieh dir die Methoden `getParentNode()` und `removeChild()` an. Sieh dir die `ids` der <g>-Element in `picture.svg` an.*
- Erzeuge eine `XPathExpression`, welche die *Sonne* im Bild findet. Ändere die Farbe der *Sonne* auf rot und entferne den orangenen Rand. *Hinweis: Sieh dir die Methode `setAttribute` von `org.w3c.dom.Element` an.*
- Erzeuge eine `XPathExpression`, welche den gesamten *Baum* im Bild findet und verschiebe ihn um 350 Pixel nach rechts. *Hinweis: Sieh dir `https://developer.mozilla.org/en-US/docs/Web/SVG/Element/g` und das `transform`-Attribut an.*
- Erzeuge eine `XPathExpression`, welche alle *Grashalme* im Bild findet. Verändere die x2- und y2-Attribute der `Grashalme` so, dass das `Gras` doppelt so hoch wird.

Abgabe

Beachte bei der Abgabe folgendes:

- Die Bearbeitung kann in Gruppen erfolgen, muss jedoch nicht. Wichtig ist, dass, sollte die Abgabe in einer Gruppe erfolgen die Namen der Mitglieder erkenntlich sein sollten.
- Die Abgabe erfolgt über Moodle.
- Der abgegebene Code muss kompilieren.
- Alle von Ihnen erzeugten Klassen und Methoden müssen ausreichend mit JavaDoc und normalen Kommentaren versehen sein.
- Alle Fragen müssen hierbei sinnvoll beantwortet sein.

Notenbonusklausur 1

Die erste Notenbonusklausur findet am Mittwoch den 29.5.2019 an Stelle der Vorlesung von 12.30 Uhr bis 13.30 Uhr statt. Die Ergebnisse werden im Moodle veröffentlicht werden sobald die Korrektur beendet ist.

Fragen

Wenn du Fragen hast, nutze bitte das Forum im Moodle Kurs oder frage deinen Tutor.