Objektorientierte Programmierung Blatt 9

Institut für Softwaretechnik und Programmiersprachen | Sommersemester 2024
Matthias Tichy, Raphael Straub und Florian Sihler

Abgabe (git) bis
30. Juni 2024

XML in Java

6 ★ 10 🟦 3 🟢

- XML, XSD, SVG
- XML in Java
- XPath

Aufgabe 1: Ice Cream Van Data

In dieser Aufgabe wollen wir erneut unserem Lieblings-Eisverkäufer helfen. Letzte Woche haben wir ihm bereits geholfen, seine Sales-Informationen in Dateien zu speichern und zu laden. Allerdings ist das Business inzwischen so stark gewachsen, dass weitere Daten gespeichert werden müssen. Deshalb sollen die Daten nun im XML-Format gespeichert werden.

a) XML-Format ★★

Zunächst einmal müssen wir festlegen, wie genau die Daten im XML-Format gespeichert werden sollen. Wir haben folgende Anforderungen:

1. Es gibt eine Menge an Eiswägen.
2. Jeder Eiswagen hat eine eindeutige ID und eine Position.
3. Jeder Eiswagen hat eine Liste an Bestellungen.
4. Jeder Eiswagen hält eine Menge an Eissorten.
5. Für jede Eissorte gibt es eine eindeutige ID und einen Preis.
6. Der Eiswagen hält für jede Eissorte die aktuell vorhandene Anzahl und eine Liste an Orten, an denen der Eiswagen seinen Bestand auffüllen kann.

Überlegen Sie sich, wie Sie diese Anforderungen in XML abbilden können. Erstellen Sie als Beispiel ein XML-Dokument mit zufälligen Daten, das die oben genannten Anforderungen erfüllt.

b) Validate XML with XSD 🟢🟢

Nun wollen wir die XML-Dateien validieren. Hierfür soll eine XSD-Datei erstellt werden. Wir haben bereits eine lückenhafte XSD-Datei erstellt, die Sie erweitern können. Diese finden Sie im vorbereiteten Repository.

Vervollständigen Sie die XSD-Datei, sodass sie die oben genannten Anforderungen erfüllt. Sie müssen hierbei den Teil für das *IceCreamStock*-Element vervollständigen.

c) Compatibility

Nun müssen wir noch sicherstellen, dass XML und XSD zusammenpassen. Hierfür können Sie den folgenden Validator verwenden: `https://www.utilities-online.info/xsdvalidation`. Validieren Sie sowohl Ihr eigenes XML-Dokument als auch das von uns bereitgestellte Beispiel.

Falls es Probleme gibt, korrigieren Sie die XSD-Datei, bis keine Fehler mehr auftreten. Geben Sie an, welche Änderungen Sie vorgenommen haben.

Aufgabe 2: XML in Java

Nun wollen wir die XML-Dateien in Java einlesen.

a) Klassenstruktur

Wenn wir die XML-Dateien einlesen wollen, benötigen wir eine passende Klassenstruktur. Erstellen Sie eine Klassenstruktur, die die oben genannten Anforderungen abbildet. Erstellen Sie auch passende Konstruktoren und Methoden, um die Daten zu setzen und abzurufen.

Orientieren Sie sich an dem XSD-Schema aus der vorherigen Aufgabe.

b) XML einlesen

Erweitern Sie die Klasse `IceCreamBusiness`, die Sie in der vorherigen Aufgabe erstellt haben, um einen Konstruktor, der den Pfad zu einer XML-Datei als Argument nimmt, diese lädt und die Klassen instanziiert. Achten Sie darauf, dass mit den Exceptions, die beim Einlesen und Verarbeiten der XML-Datei auftreten können, korrekt umgegangen wird.

c) XML Validierung

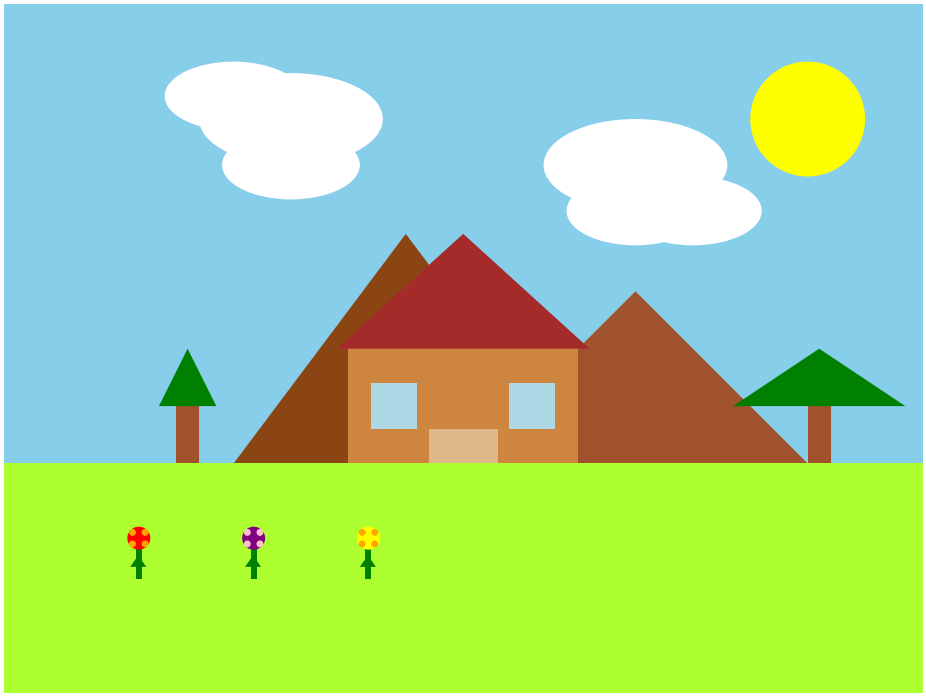
Verwenden Sie die XSD-Datei, um die eingelesenen XML-Dateien zu validieren.

d) Tests

Nun sollen Sie selbst Tests schreiben, um sicherzustellen, dass die XML-Dateien korrekt eingelesen werden. Verwenden Sie hierfür die bereitgestellten XML-Dateien und erstellen selbst einige weitere XML-Dateien, die Sie testen wollen. Testen Sie auch, ob die Validierung korrekt funktioniert und ob Exceptions korrekt geworfen werden.

Sind Sie in der Lage, eine XML-Datei zu erstellen, die von XSD als valide erkannt wird, aber von Ihrer Klassenstruktur nicht eingelesen werden kann? Kann Ihre Klassenstruktur Werte speichern, die von dem XSD-Schema nicht erlaubt sind?

Aufgabe 3: SVG and XPATH



SVG ist ein XML-basiertes Dateiformat, das zur Darstellung von Vektorgrafiken verwendet wird. XPath ist eine Query-Language, die verwendet wird, um Teile eines XML-Dokuments zu selektieren. In dieser Aufgabe wollen wir die Mächtigkeit von XPath nutzen, um Informationen aus einem SVG-Dokument zu extrahieren und dann in Java zu verändern.

Da SVG ein XML-basiertes Dateiformat ist, können wir diese wie in der vorherigen Aufgabe einlesen und verarbeiten. In den folgenden Aufgaben sollen Sie Methoden schreiben, die eine SVG-Datei einlesen, diese wie in der Aufgabe beschrieben verarbeiten und die veränderte Datei wieder speichern.

a) Clouds in the Sky

Zunächst wollen wir nur Elemente in der SVG-Datei selektieren und noch nicht verändern. Schreiben Sie eine Methode, die alle Elemente in der SVG-Datei selektiert, die eine Wolke darstellen.

b) Plants

Schreiben Sie eine Methode, die alle Elemente in der SVG-Datei selektiert, die alle Pflanzen, also Bäume und Blumen, darstellen.

c) Specific Plants

Schreiben Sie eine Methode, die einen `Integer` zwischen 1 und 3 als Argument nimmt und alle Elemente in der SVG-Datei selektiert, die Teil der Blume mit dieser Nummer sind. Hierbei ist die linke Blume die Nummer 1, die mittlere die Nummer 2 und die rechte die Nummer 3.

d) Renovate the House

In dem gegebenen SVG-Dokument sehen Sie ein Haus. Leider gefällt mir die Farbe nicht mehr. Ich hätte gerne ein rotes Haus mit einem schwarzen Dach, weißen Fenstern und einer grünen Tür.

e) Improving Nature 

Die Bäume im SVG-Dokument sind nicht schön. Entfernen Sie die Blätter der Bäume und ersetzen Sie diese durch mehrere grüne Kreise, die die Blätter schön darstellen.

f) Bigger Nature 

Die Bäume im SVG-Dokument sind zu klein! Vergrößern Sie die Bäume um 20%. Achten Sie darauf, dass die Bäume immer noch auf dem Boden stehen.

Versuchen Sie die XPath-Expressions so zu schreiben, dass sie für das initiale SVG-Dokument und das veränderte SVG-Dokument mit schöneren Blättern funktionieren.