PTT - Assignment 4
Megamodeling (Gson)

Marco Schanz, Marius Beckmann, Diana Richter, Frank Schaust,
André Thomas, Isabelle Kuhlmann

11.07.2016

1 Formulierung des Usage Scenarios

1.1 Generelle Idee

Eine Android-App nutzt einen beliebigen API-Endpunkt, über welche Ressourcen im JSON-Format abgefragt werden. Da wir in Java nicht mit JSON sondern am liebsten mit Objekten arbeiten wollen, nutzen wir gson zur De- und Serialisierung.
Auf diesen Vorschlag hin haben wir die folgende Rückmeldung bekommen:
"Das klingt gut.
Am besten wäre es also, wenn Sie das Szenario durch eine reale API, ein reales Objektmodell und eine reale Funktionalität illustrieren können.
Generell ist das aber in der Tat ein allgemeines Szenario für GSon."

1.2 Illustriertes Modell

Entsprechend haben wir folgendes festgelegt:
Als API benutzen wir OMDB.
(http://www.omdbapi.com/)
(https://en.wikipedia.org/wiki/Gson)

Als Objektmodell haben wir eine Klasse Movie mit verschiedenen Attributen mit einer toString-Methode.
Reale Funktionalität: Auf Basis einer Anfrage über die API bekommen wir eine .json-Datei zurückgeliefert, die wir dann über Gson als Java-Objekt in unserer Datenstruktur verarbeiten können.

1.3 Beispielnutzung

Wir fragen eine Json-Datei aus der omdb-API ab. Diese wird nach dem Einlesen via gson in ein Java-Objekt Movie m konvertiert. Danach wird das Objekt mit der toString()-Methode ausgegeben.

```
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import java.io.BufferedReader;
import java.io.InputStreamReader;
```

```
import java.net.URL;

public class Main {
    //Liest den durch readUrl gegebenen String ein.
    private static String readUrl(String urlString) throws Exception {
        BufferedReader reader = null;
        try {
            URL url = new URL(urlString);
            reader = new BufferedReader(new InputStreamReader(url.openStream()));
            StringBuffer buffer = new StringBuffer();
            int read;
            char[] chars = new char[1024];
            while ((read = reader.read(chars)) != -1)
                buffer.append(chars, 0, read);

            return buffer.toString();
        } finally {
            if (reader != null)
                reader.close();
        }
    }

    public static void main(String[] args) {
        String s = "";
        try {
                    //liefert einen String
            s = readUrl("http://www.omdbapi.com/?t=inception&y=&plot=short&r=json");
        } catch (Exception e) {
            e.printStackTrace();
        }
                //Initialisiert ein Gson-Objekt
        Gson gson = new GsonBuilder().create();
                //konvertiert den Json-String s in ein Java-Objekt der Klasse Movie
        Movie m = gson.fromJson(s, Movie.class);
                //Ausgabe des Movie-Objektes m
        System.out.println(m);
    }
}
```

Klasse Movie mit toString-Methode für die Ausgabe:

```
/**
 * Created by marco on 06.07.16.
 */
public class Movie {
    private String Title;
    private String Year;
    private String Runtime;
    private String Director;
    private String Actors;
    private String Plot;
    private String imdbRating;

    @Override
    public String toString() {
        return "Title: " + Title
                + "\nYear: " + Year
                + "\nRuntime: " + Runtime
                + "\nDirector: " + Director
                + "\nActors: " + Actors
```

```
                + "\nPlot:␣" + Plot
                + "\nimdbRating:␣" + imdbRating + "\n";
    }
}
```

Ausgabe in der Konsole:

```
Title: Inception
Year: 2010
Runtime: 148 min
Director: Christopher Nolan
Actors: Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen Page, Tom Hardy
Plot: A thief, who steals corporate secrets through use of dream-sharing technology, i
imdbRating: 8.8
```

2 Megamodel

Technological Breakdown:
Platform < Technology
Library < Technology
LanguageProcessor < Technology

ProgrammingLanguage < Language
Java: ProgrammingLanguage - `https:/en.wikipedia.org/wiki/Java_(programming_language)`
MarkupLanguage < Language
Json: MarkupLanguage - `https://en.wikipedia.org/wiki/JSON`
Gson: Library - `https://en.wikipedia.org/wiki/Gson`
Gson implements fromJson
JRE: Platform - `https://de.wikipedia.org/wiki/Java-Laufzeitumgebung`
JRE implements Java
javac: LanguageProcessor - `https://en.wikipedia.org/wiki/Javac`
javac partOf JRE

Concepts:
Protocol < Concept
http: Protocol - `https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol`
Serialization:Concept
Gson facilitates Serialization

Resource < Artifact
File < Artifact
aJsonFile:Resource
aJsonFile ∈ Json
javaClasses:File+
javaClasses ∈ Java

application:File+
application ∈ Java
aMethodCall:Fragment
aMethodCall partOf application
aMethodCall uses Gson

aMethodCall defines fromJson

Transient $<$ Artifact
aJavaObject:Transient
aJavaObject $\in$ Java
fromJson $\in$ Function
fromJson:Json $\rightarrow$Java
fromJson(aJsonFile) $\mapsto$ aJavaObject
aJsonFile correspondsTo aJavaObject