Software Architecture Group
Faculty of Informatics
University of Vienna
http://cs.univie.ac.at/swa

050132 Advanced Software Engineering (UE), Winter Semester 2014/15
Task 4

Task 4 - Domain specific language with Groovy

Task-ID: GRV

General Remarks

- The **deadline** for this work is the 07.12.2014 which can also be found in Moodle (https://moodle.univie.ac.at/course/view.php?id=34717).
  The deadline is firm! No deadline extensions are given.

- You will have to present your solution – so be prepared
  (laptop, pdf of your presentation, running example, ...)
- You can get up to 10 points, which have to be defended during presentation.
- This is an **individual work**. Consequently, group work is not allowed in this assignment.
- If you have problems, do not hesitate to contact the tutor (ase.tutor@swa.univie.ac.at) or us (ase@swa.univie.ac.at) via e-mail.

Submission Guidelines

- The guidelines proposed on https://swa.univie.ac.at/General_guidelines must **always** be satisfied.
- If guidelines are ignored,
  - an assessment of the tasks **cannot be guaranteed** because they are processed electronically.
  - one (1) point will be deducted from the assessment of the task for each unsatisfied guideline.

- Artifacts to upload into Moodle are (see guidelines for more information):
  One (1) ZIP archive containing:
  - The Groovy Project containing all sources
  - Sample DSL instance sources
  - Document about your assumptions (PDF)

Hint:

You can install the Groovy Eclipse Plugin for Eclipse by defining a new Update Site *Help -> Install new Software -> Add...* (have a look at [8] to find the correct URL) and then selecting the newly created Update Site. You can install the required components from there.

This tasks requires knowledge of scripting languages, builder pattern and Groovy. If these topics are new to you, start soon and read the tutorials [2,3,4,5].

Task Definition

Right at the Start

Please, read the guidelines at https://swa.univie.ac.at/General_guidelines, especially the part about naming conventions for your uploads! If you do not follow these guidelines, we will not be able to grade your work, because we will not be able to find your files any more (yes, it is that simple ...).

Software Architecture Group
Faculty of Informatics
University of Vienna
http://cs.univie.ac.at/swa

050132 Advanced Software Engineering (UE), Winter Semester 2014/15
Task 4

a) Grammar

Create a (internal) domain specific language (DSL) for the playlist model from Task 3 using Groovy! Your DSL should be easy to read and understand.

Use Groovy [6,7,8] to create a DSL for the Streaming Domain. **Do not** use the java classes created in previous tasks (you will get problems creating your syntax).

The grammar should be human readable – so everyone who will look at instances written in your grammar should easily understand what s/he is looking at.

Use as much Groovy constructs as possible. You may find the tutorials [2,3,4,5] inspiring, too. You have to use the builder pattern and probably want to use features like operator overloading.

Description of the music streaming library model:

An online streaming music library contains a collection of songs. Each song has a name, an artist, a distributor, a length, a price, and a genre. At least the following genres should exist in our DSL: POP, ROCK, HARDROCK, CLASSIC, FOLK.
Distributors have a unique name, an address, and account information.
Your DSL needs to allow the definition of playlists based on the songs in the library and based on other playlists.
This means that a playlist (which has a unique name) can contain an arbitrary number of songs, and can reuse other playlists by placing the name of the reused playlist somewhere on to the new playlist. It should also be possible to exclude some songs from the reused playlist. Please see the example instance from Task 3 that again is included in the Appendix for details. We are aware that a DSL written in Groovy will not look exactly a DSL written in Xtext. However readability is still very important.

Aside from this description, your DSL does **not** have to model any other elements from previous tasks.
Create your DSL from scratch. Have in mind that you will implicitly create a domain model. Try to structure it in a sound way, so you can also use it without the DSL.

b) Instances

Create a sample instance (which basically is a groovy script) using your DSL grammar. You have to create at least two instances of each of your model's classes. Your script should print one table per playlist that lists the songs in each playlist to the console or to a file (no special formating is required). In the table only songs are listed (songs from referenced playlists are also listed). Furthermore your script should also create an extra table that lists the earnings per distributor over all existing playlists (sum up of the prices) sorted by the amount of earnings (descending).

Software Architecture Group
Faculty of Informatics
University of Vienna
http://cs.univie.ac.at/swa

050132 Advanced Software Engineering (UE), Winter Semester 2014/15
Task 4

Resources and Tipps:

Feel free to make your own assumptions. Mention them in your document.
If you have questions about the requirements, feel free to discuss them with your colleagues and us using the moodle forum.

Checklist

Before uploading your solution, ensure the following checkpoints can be marked positive:

- I followed the guidelines and conventions from https://swa.univie.ac.at/General_guidelines
- I created a Groovy DSL.
- I created an instance of the DSL and included it in my ZIP file.
- My groovy script prints the playlist tables and the sorted earnings table.
- I documented my assumptions in a PDF.

References

[1] Moodle, https://moodle.univie.ac.at
[2] DSLs in Groovy http://groovy.codehaus.org/Writing+Domain-Specific+Languages
[3] Interne DSL mit Groovy (deutsch)
http://raik-bieniek.blogspot.co.at/2011/03/interne-dsl-mit-groovy-teil-1.html (German)
[4] Simple DSL about distances
http://joesgroovyblog.blogspot.com/2007/09/and-miles-to-go-before-i-sleep.html
[5] A Groovy DSL with Builder – (almost at the bottom of the page)
http://www.javabeat.net/groovy-for-domain-specific-languages/
[6] Groovy, http://groovy.codehaus.org/
[7] Groovy with Eclipse, http://www.vogella.de/articles/Groovy/article.html
[8] Groovy Eclipse Plugins, http://groovy.codehaus.org/Eclipse+Plugin

Appendix – Example of a DSL Instance (same as Task 3)

```
Distributors:
      UniversalMusic
      Address: "56 Some Street, 12345 New York, NY, US"
      Account Information:
            IBAN: "US 12 123 123 123"
            BIC: "SOMEUSX1"

      SonyMusic
      Address: "1 Sony Street, 2345 Tokia, JP"
      Account Information:
            IBAN: "JP 12 34 43 55 93"
            BIC: "SOMEJPJ1"

Library:
      Come_as_you_are
      sung by "Nirvana"
      produced by UniversalMusic
      length: 7:23
```

Software Architecture Group
Faculty of Informatics
University of Vienna
http://cs.univie.ac.at/swa

050132 Advanced Software Engineering (UE), Winter Semester 2014/15

Task 4

```
      genre:ROCK
      price: 7.75

      Bohemian_Rapsody
      sung by "Queen"
      produced by UniversalMusic
      length: 5:32
      genre: ROCK
      price: 3.65

      Get_lucky
      sung by "Daft Punk"
      produced by SonyMusic
      length: 3:01
      genre: OTHER
      price: 4.32

      Song2
      sung by "Blur"
      produced by UniversalMusic
      length: 4:23
      genre: ROCK
      price: 3.45

      Roar
      sung by "Katie Perry"
      produced by SonyMusic
      length: 2:35
      genre: POP
      price: 4.67

Playlists:
      SimpleList consists of
      Bohemian_Rapsody,
      Come_as_you_are,
      Get_lucky

      ComplexList consists of
      Song2,
      Playlist SimpleList without Get_lucky,
      Roar

      RockList consists of
      Song2,
      Bohemian_Rapsody,
      Come_as_you_are
```