UNIVERSITÉ CÔTE D'AZUR
L3IA
2022-2023

Julien Deantoni

Language et Compilation

ChartIt

Project delivery

This document concerns the project delivery of the Language and compilation course. It will include the implementation of a DSL focused on the specification and tooling of alanguage to extract data from CSV files and represent them as charts by using the *chartjs* library. Deliveries are expected by email (to Julien Deantoni: firstname.lastname@univ-cotedazur.fr, with [L3IA] as object prefix) followed by "firstname lastname" (where firstname and lastname are your actual names !). The delivery is expected before the **TBF**. The delivery is expected as a PDF report (please let your report be succinct and rigorous).
The report must contain :

- a link to the code of your language (typically a link to the git repo)
- a description of the language proposed:
  - the domain model represented as a class diagram;
  - the concrete syntax represented in a BNF-like form;
  - a description of your language and how it was implemented;
  - a simple description on how you wrote the compiler to obtain the resulting html page
- a set of relevant scenarios implemented by using your language;
- a rationalization of its main usage and all of its features. The associated script and materials will be provided.

Objectives: Define the *ChartIt* language

Your objective here is to define a tooled language allowing

- the loading of file containing data formatted by using the Comma Separated Values format.
- the filtering of columns and rows according to the specification of rules.
- the definition of how the data should be represented by a chart.

The provided language should allow an easy and intuitive manipulation of structured data loaded from the CSV files (see an example of classical CSV files here: `https://perso.telecom-paristech.fr/eagan/class/igr204/datasets`).
Basic scenarios are defined below and must be supported by your language. However, you should define more complex scenarios and keep them in your repository. You should at least provide one program to

illustrate each functionality of your language. Keep in mind that the goal is to provide an easy and intuitive language to obtain charts from raw data.

The programs written in your language should in fine generate an HTML file that contain the required JS script to represent the data. An simple example of such file is provided here `https://www.i3s.unice.fr/~deantoni/teaching_resources/L3IA/Languages/current/TDs/test.html`.

Basic scenarios

1. **Simple Car power in the US**

   Based on the data here: `https://perso.telecom-paristech.fr/eagan/class/igr204/data/cars.csv`, Alice wants to represent a bar chart representing the horse power of cars in the US. One possible result is the following :`https://www.i3s.unice.fr/~deantoni/teaching_resources/L3IA/Languages/current/TDs/scenario1.html`.

2. **Advance car power chart**

   Based on the dataset here:`https://perso.telecom-paristech.fr/eagan/class/igr204/data/factbook.csv`, Bob wants to compare the population number and the cellular number of each country where the data are known. One possible result is the following :`https://www.i3s.unice.fr/~deantoni/teaching_resources/L3IA/Languages/current/TDs/scenario2.html`.

Commmon Parts

The following parts must be available in your DSL:

- **Domain Model**:

  The domain model (a.k.a. abstract syntax) should be clearly identified in the delivered code. It will be provided as a class diagram, together with explanation about the main choices you did to propose easy and intuitive data manipulation.

- **Concrete syntax**:

  The concrete syntax, in a eBNF form, must be clearly identified and used by a relevant set of scenarios. The syntax must introduce the textual constructions you proposed in your language in order to make it easy and intuitive.

- **Validation**:

  Support your end-user by checking that a model is realizable on the expected platform. For instance the file to load actually exists, or the column names used in the programs are actual columns of the loaded file.

- **Code generation**:

  Provide a generator producing *html* code, which, when interpreted by a browser show a chart of the data.

Any of the points above may be discussed between you and me during the course hours or by email.