Dominion Project

Disclaimer

The project was developed for Software Engineering course, Polytech Nice 2023.
The technologies used were mainly: Eclipse Modelling Framework, Xtext, Java.

Author: Tymoteusz Ciesielski, tymotheus.c@gmail.com

Prof: Frederic Mallet

Description

As a project we were supposed to describe our own Metalanguage in a domain chosen by us. It could be an attempt to describe a turing complete programming language like Python, or a Domain Specific Language (DSL) (as I chose) or something "unusual" like defininig your metalanguage of cooking.

My contribution towards the project

I chose describing the model of the card game "Dominion" as I think that this game is both complex enough, and simple in its rules which makes it good to model. The complete manual of the game can be found here: https://cdn.1j1ju.com/medias/59/e6/c2-dominion-rulebook.pdf
The language created by me is MDL - MetaDominionLanguage.
I created:

- The Metamodel of the game, split into 3 class diagrams: dominion game (main), players turn (strategy), action diagram (helper)
- Concrete textual syntax for the model (MDL), using the Xtext framework
- Plugins modelling the setup of the game alongside the initial user interface
- Simple tests and sample players strategies

Using the program

The program should be used by the end user in such manner:

1. User (non-tech person) installs eclipse and the plugins provided by me.
2. User can write his own strategy into .dml file, following easy-to-use syntax close to the natural language. They can do it in the MDL editor provided by me (syntax colouring and auto-completion provided) or in any other editor of their choice. This way the whole strategy created by a player can be packed in a small and concise form.
3. The strategy file gets translated into the abstract tree, which detects concrete objects defined by my metamodel.
4. Using the Dominion.ui plugin provided, user can call the Dominion menu, which should launch the benchmark of their strategies against each other or against some predefined ones. (Model explorer on the left of Eclipse -> right click on the correctly created .mdl file **containing a strategy element**, **on the program element of the abstract tree**. You should see a popup Dominion menu, and option "Start a game")

5. Alternatively a user can model any situation in the game or try to play the game interactively, though it was not created with such idea.

Structure of the project

- Dominion project contains the metamodel as well as all the java classes automatically generated for it
- Dominion.editor is the editor plugin allowing to build custom setups for the game as tree structures
- metaDominoLang.mydsl/.ide/.tests/.ui/.ui.tests contain all the structural files related to the MDL language. In metaDominoLang.mydsl\src\metaDominoLang\MetaDominoLang.xtext I define the syntax of the language
- DominionGameFunctions is a plugin where I should define all the helper functions. For now those are the functions generating program, starter setting for a game, or drawing cards for the player from his deck, handwritten in Java.
- Dominion.ui should be the user interface, to interact with all the functionalities. For now it implements custom windows and handlers, allowing the user to read the created strategy, and set up the game with its starting settings.

Whole library of the cards (and library of abilities), possible to use should be defined either as a java library or in a separate .mdl file, allowing the end user to access it

What needs to be implemented?

- In the metamodel, there should be "Greater or equal", "Lesser or equal" expressions implemented in players turn, currently the user can only build strategies with money equalities.
- Shuffling the deck. I did not manage to implement it, as I was receiving errors that shuffling List<> object requires it not to have duplicate elements (weird)
- DominionGameFunctions should be extended for much more functions related to the game, so it can be fully modelled. Functions for buying cards, discarding them etc.
- Ideally, the program would allow to run the full benchmark with the whole dynamic of the game

What other flaws I am aware of?

There are couple of inaccuracies of the project, some of them I am aware of but probably they are not the only ones:

- I probably commited a sin, because I implemented toString() methods of CardImpl and PlayersHandImpl in the src-gen files that should not be touched
- I was trying to either assign strategy created by a user to the game created by generateProgram() function, or to call generate2PlayersGame() for the program created by a user but both this approached generated the same error, so for now the strategy lives in a separate "DominionProgram" object compared to the benchmark programs
- Action diagram definitely needs refactoring, as I changed my approach a bit after the talk with the prof, and moved most of the player's turn logic into the players turn diagram.

- Many references between models are not fully logically correct. For example Dominion Game should always have a trash pile (even if it is never used), or according to the rules, supply piles should start with 10 cards (not any number). But as I read in a tutorial, this loosening the restrictions a bit helps with early modelling and I had the same impression
- xtext syntax should look better, I mostly worked to be able to obtain nice-looking strategy files and that was my main goal. I also wanted to obtain some minimal working demo functionality

Other remarks

MDL was my 3rd attempt to create a language built upon this model, .dmn and .dmns languages were abandoned.
I estimate I spent more than 50 hours on this project.