Final Project
DSL Design: Card Games

Due: (various; see writeup)

Goals for this Assignment

1. Practice with domain-specific language design, applying ideas from the course.

Office Hours

The purpose of office hours is to provide you with a chance to discuss class-related topics, to ask questions about material you find particularly interesting or particularly confusing, and to provide comments about the class.

You can drop by Olin 1253 any time to see if the professor is around; times specifically set aside are Tuesday, Thursday, and Friday from 2:30–4pm. Grutors are available in the Terminal Room on Tuesdays 7–9pm and Sundays 2–4pm. Questions can also be sent to `cs131help@cs.hmc.edu` .

General Instructions

- You are strongly encouraged to work on this project in pairs, but are not required to do so. You are *not* required to work with the same partner as in Assignment 6.
- Submissions must be typed up and submitted as the files `draft.pdf` and `cards.pdf` respectively (using `cs131submit` as Assignment 8). Only one submission is required per group.

Project Statement

You have been hired by a company that wants to make money by allowing people to play card games over the Internet. Since other such systems exist (e.g., Yahoo) this company wants to make theirs more attractive by making it highly configurable; they will allow new games to be added quickly — perhaps even by users. (Eventually they hope to have computerized players that "learn" how to play new games well, but this is beyond the scope of your project.) The idea is that the rules for any particular card game can be loaded from a configuration file. The system will then set up the game, allow actions by users (ensuring that the rules of the game are obeyed), detect when the game is done, and determine the winner(s).

The goal for the project is for you to propose an initial design for a domain-specific language for specifying card games.

You'll have to spend time brainstorming; here are a couple questions to get you *started*:

- *What is the language's structure?* Is it an imperative language where a program loops the players and gets input from them and changes the state of the game appropriately? Is it a declarative language where the programmer specifies constraints on the legal actions from some larger built-in set of possible actions? Is there a fixed event loop that makes calls to your code at various points? Something completely different?

- *What are the types in your language?* Even if the language doesn't have a type system, there are likely to be different "sorts" of things around, such as players and cards.[1] Is the set of cards fixed, or can it be varied? Are cards and players just integers, or different sorts of values?

  Both a hand of cards and a deck of cards can contain zero or more cards; are they the same sort of value, instances of a more general idea (such as a list, set, or stack) or are hands different from decks? How are both related to a discard pile?

  Different games may order cards differently; is there one (or more) built-in orders or are they user-definable? Is the order a property of a specific deck of cards, an independent sort of entity, a function, . . . ?

  What other sorts of things might be relevant to this language?

- *What built-in primitives do you need?* Mathematical, I/O, . . . ?

The expectation is *not* that your language will be able to specify every possible card game, just that it can describe some interesting subset. You are also generating a first proposal for the language; small languages tend to be altered, extended, and improved once as they are implemented and used. ActionScript and Perl, for example, have evolved substantially over several releases.

[1] As a similar example, if one were designing a language like *pic* to draw simple diagrams, one might look at existing diagrams and start thinking about rectangles, circles, labels, directions, distances, bounding boxes, etc.

Milestones

1. By **Thursday, April 10**, you should e-mail the professor if you want a partner but have no one particular in mind; students will be paired up as requests come in.

2. In class **Monday, April 14**, each group must hand in a short writeup with the following information:

   - The name(s) of the student(s) involved.
   - The name of your language.
   - Choose a primary goal and a secondary goal from the following list
     - **Ease of use**. The ability to specify card games without being an expert programmer.
     - **Breadth**. The ability to implement a many different sorts of games.
     - **Elegance**. A clean and general language, rather than a grab-bag of special-case features

   (These may or may not be difficult to achieve simultaneously.)

3. For **Wednesday, April 23 (NO EXTENSION)**, each group should submit `draft.pdf` (as Assignment 8), containing a fairly complete rough draft of the language design. This should include:

   - An unambiguous concrete syntax (including any specifications for valid identifiers, numbers, etc.)
   - A user manual/language specification for the semantics of the language. You do not have to formally specify the semantics, but you should still be as precise and unambiguous as possible. Included in this guide should be information on the types available, the language's constructs and built-in functions, the rules for determining the scopes of variables, and descriptions of the parameter-passing conventions (as appropriate).
   - A couple of interesting game specifications as examples.
   - An explanation why the design is implementable; one approach would be to describe (very generally) how an implementation of your language might work.

4. For the last day of class, **Wednesday, April 30 (NO EXTENSION)**, each group should submit the final language design document as `cards.pdf` (also as Assignment 8).

   On this same day, your group will give a 5-minute in-class presentation. This should cover your design goals, a very brief summary of the language, at least one interesting example, and a critique of your language's strengths and weaknesses.

Grading

Your project grade will be based on the following criteria.

- Is the language appropriate for the goals chosen?
- Is the language design clear? Is it unambiguous?
- Does the language specification include all the required parts?
- Do your descriptions use appropriate vocabulary?
- Do your examples demonstrate the utility and versatility of the language?
- Is the language implementable?
- Are there spelling and grammar errors? (This is the only question where the correct answer is "no").
- Is the presentation the correct length, and does it include the required components?