**Pioneer Project Final Write-up**

Jerry Hung

**Introduction**

As more and more work pushed upon students in high school, it became harder for them to organize their time efficiently. For me, the circumstance is similar—I was unable to arrange my time appropriately, causing myself many troubles regarding finishing my work and using my time wisely. Hence, I created this project for the students who are not great at organizing time (including myself). Using a DSL allowed me to personalize the features, the outlook, and how an "Academic Calendar" functions, giving me great freedom and flexibility.

The domain of the project will be on academic calendars, geared towards high school students. This domain is great for changing your schedule into one that is easily visualized, giving you more control over your own time. One significant application of this is the iCalendar, which is what my project will be based on, as it is the base for many calendars on the web, including Google Calendar. The project is created as an external DSL of iCal4J, a library that allowed users to develop ics file using java, but in a very tiresome and complicated manner. Which is why the program I wrote is more aiming to simplify the language.

One sentence that explains the concept of my project would be: "An academic calendar that is easily created, visually appealing and has a low threshold to use." When it runs, the user can input two kinds of items, an event that spans for a period, and a to-do item that is "due" before a certain time. For this project, a lot of the specific nouns will be based on time, whether it be the time of the day, a future date, or even a weekday, event, and destination. The biggest problem for the project should be compiling errors, as a part of the project would be to "compile the language from everyday language, via Java, to ics." To prevent this, I need to try to avoid the issue by debugging the program. This project is also a good way for students to familiarize themselves with programming. With this project, the users will learn ideas like syntax, file locations, input files, and other essential pieces of coding. As the project is geared toward regular students, the language is made to be very user-friendly and simple to use.

**Background**

What is a DSL? How does it benefit our lives? DSL is short for Domain Specific Languages. In the computation world, there are mainly two types of languages: general purpose languages like Java, C++, or JavaScript, and domain-specific languages. What differentiates between the two is the scope of functionality. A DSL is primarily a language specialized to a specific domain, or range of interest. Even without knowing it, DSLs have been influencing our lives as even programs like AutoCAD is a DSL.

So, what do DSLs look like? DSLs come in many different shapes and sizes: there are DSLs for creativity like Melodica, Musyc, and ContextFree; DSLs for simulations like AutoCAD, VirtualGlass, and Logism; DSLs to improve upon an existing program like Scalatest or LaTeX. These DSLs, even though their purposes are all different, they all converge to a common idea: a language specified to a specific domain. A good DSL needs to implement and consider many things. First, it must have a purpose to focus on, which is what separates it from a general-purpose program. Next, it should include some specific nouns, adjectives, adverbs, and such dedicated to the domain of the language. Most importantly, a good DSL needs to put its users in mind. As you are not going to be the only one using the DSL, you need to consider the general users and “audience” of the software and make changes according.

When designing a DSL, the process is not just about your own opinion. You also must consider what others think. One of the most important techniques used is to ask questions. Whether by asking yourself what you wanted to have in the language or asking the “audience” about what they think should be in the DSL, it will be helpful to embed the feedback into it. Also, the language needs to be consistent. As a DSL is designed for a specific purpose, the features of the program should be of some use to that purpose, otherwise, it should not be included. It is not uncommon for implementers to create a DSL based on other languages, and if that is the case, it is often a good idea to reuse. One can reuse the existing language definitions, reuse the type systems. In short, reusing prevents the implementer from recreating the wheel, allowing the implementer to spend more time in creating the DSL itself, not the syntax.

One example of a successful DSL is Melodica. Melodica is a program that allowed the user to create music with little squares on a grid. Each square of a column represents a different note, and the music is created by playing the notes from column to column, repeating and allowing the user to add “squares” along the way. This DSL and all of its features are created solely to allow the users to create music easily. The simple concept and interactive design allowed people with no knowledge of music to also use the software. Another example of a DSL is LaTeX (Lamport TeX), a document preparation system. This DSL is designed towards mathematicians and scientists. LaTeX allowed them to type symbols like subscripts, superscripts, scientific symbols, equations with ease. The DSL stands very consistent and very effective in the “role” it is designed for, making this a successful DSL.

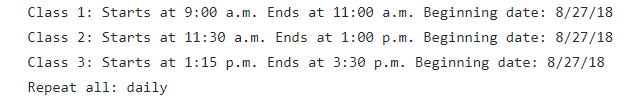
In conclusion, as opposed to general-purpose languages like Java and C++, Domain Specific Languages are designed towards a more specific focus. Creating a DSL requires one to delve deep into what the “focus” or domain is, the functions and features to add to that domain, and the suitability.

**Language design details**

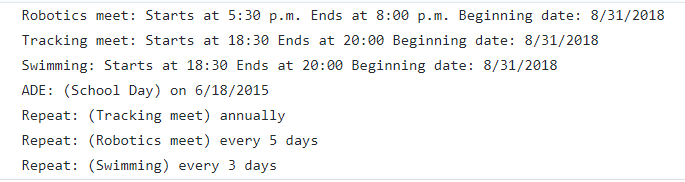
The idea for this project first came to mind when looking at the ideas that professor Wiedermann posted on the instructions. After considering my circumstances and my skill level, I decided on creating an academic calendar. The user (most likely a student) will be using this program as a way of creating an academic calendar in a more "human" fashion. The user, instead of inputting complex code that requires a complicated syntax, can create a calendar with everyday language. For example, if the user wanted to say that there is an English class from 9:00 a.m. to 3:00 p.m. on 8/1/2018, the user will have to input: "English class: Starts at 9:00 a.m. Ends at 3:00 Beginning 8/1/2018. Also, the program essentially uses the first word of each sentence as an identifier, allowing the inputs to be easily parsed. The program includes many features that are user-friendly and will decrease the amount of “writing” to make an event. The program has an ADE event parser, allowing users to create all day events. Combining with the repeat and repeat-all functions, the code can successfully decrease the repetitiveness of creating a calendar.

Furthermore, as the program automatically creates a .ics file at a default location, the user only needs to load the data into an online calendar that supports iCal for the calendar to function. Finally, as the project is based around iCal, this is also built on upon the iCal4j. This project turns a text file into a calendar file. Compared to the messy iCal4J, this project is easier to use and more user-friendly.

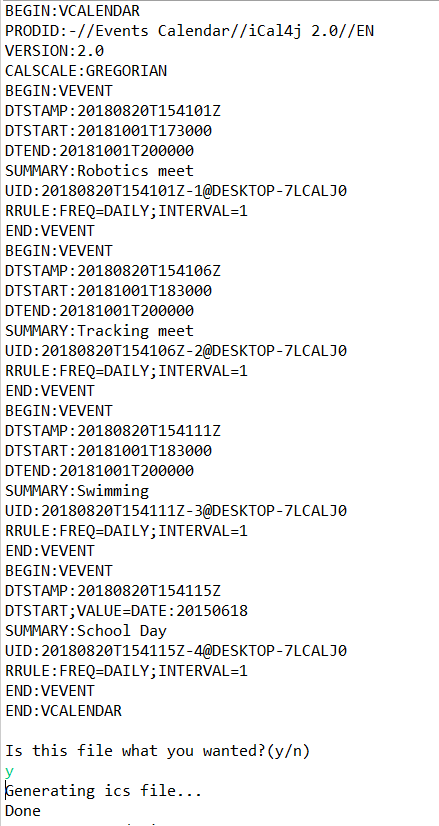
**Example program(s)**

In a sense, this project is a translator of a sort. The code translates “everyday language” into a machine language that is hard and tiresome for some people to understand. Some example inputs of the program include:

This piece of code is what an average student would input into the program. The first three lines regarding Class 1, 2, and 3 are to create an event, specifying the time starting and ending.



This second piece of code is to simulate one who is packed with extra-curricular activities, creating events like tracking meet, swimming meet and so on. Attached below is one of the sample outputs to the code, as one can see, even though the code looks repetitive, there are many details that one needs to be careful of. Otherwise, the iCalendar code will not be able to function. Which is why the language is created to parse each line of the input, taking segments of information required to build a calendar, create each event and finally publish it and allow the user to put it on their calendar.



**Language Implementation**

At the beginning stages of my program, it seemed more like an internal DSL as the program is based on iCal4J and is just working on making it easier to use. Then, as the project progressed further, I made the program read a file and turn it to another format, and as the program progresses further, I am inching closer and closer to an external DSL, which is what I picture the project to be. For the host language, I chose java mainly because of my familiarity with it and how the class-oriented language fits in very well with the idea of a calendar. In a nutshell, what this program is striving for is for students to be able to use this program with little knowledge of programming. Which is why I have set a lot of details on the calendar itself as made that if they wanted to change them, they will have to do some work to change it. Parsing is one of the biggest aspects of this project. In a sense, this project is a translator, which meant that parsing from one language to the other is very important. One technique I used is to use certain symbols as “identifiers”, allowing the program to parse the code, trim the sentence down, and get the information needed. Then, after the details are snipped from each sentence, it is just a simple job of organizing the events, plugging in the variables….and so on. As Java is used as the host language, I did the program in a object-oriented form, also using ArrayLists to help organize the events.

**Evaluation**

My program, compared to some other programs I wrote for coding class or AP projects, is relatively DSL-y. At first, it is closer to an internal DSL, but after taking in advice from the professor, the program turned out into something more like an external DSL. Compared to a general-purpose language which completes a specific task, this program functions like a compiler from “natural” language to iCalendar language.

What worked well with the language is that I am proud of its completion. When I first started this project, the technical and in-program difficulties stumped me many times, even the smallest of errors is causing the compiler to not work, making a considerable part of the program focused on debugging the program. Another thing I am proud with is how I remembered Hashmap and utilized it to organize my “CEvents.” I am also glad to learn about regular expressions, which I trust will benefit me greatly in the future.

One thing I am disappointed with is that I am unable to implement the “automatic error detection” for the program. Trying to detect and analyze a correct line is a lot of work to parse, and adding an element of randomness and irregularity increases the complexity greatly, and will push the “natural language” input into more of the actual language we speak in conversations. Furthermore, the GUI for my code could be improved by implementing a graphic console for the program, making this program less text-based and more visual based.

As this is mainly a calendar designed to students, I aimed the evaluation to my peers and myself, I compared mainly to google calendar on the difference. Of course, Google's interactive calendar is easier to use, but being able to read from a text file could save a lot of time for some users. I have not made any significant changes as most of the code is written according to plan. For example, I created various implementations like repeating daily, weekly, all day and other functions from the inputs taken from my peers.

I ran into a lot of trouble when parsing the code. Sometimes to RRule gets added too many times, other times the key of the Hashmap are the same and the code did not execute properly. One of the methods I used to solve the problems is making use of the print function, allowing me to print out the inputs and outputs at each location, allowing me to pinpoint on what the problem is. All in all, the syntax and unfamiliarity with the iCal4J library has caused plenty of trouble for the completion of the project.