Byung Hun Lee

CS 408

Raheja

CS 408 Term Paper

Our team chose Apex as the language to evaluate in this group project. Apex, marketed as “The World’s First On-Demand Programming Language” is basically Java combined with a built-in database, support for data between users, user interface, and external application support. Well, there were several small differences besides the differences listed above. While the purpose of this class was to learn the differences between programming languages and learn the specific implementations required by programming languages, it was certainly a different experience learning via power-point and learning via hands-on research.

To begin with, I will discuss the language features as briefly as possible. As discussed in the opening paragraph, Apex was created by the team in Salesforce as a propriety language for their customers; and the software team in Salesforce decided to use Java as a basis for their new language, Apex. In terms of readability, comprehending the language will be easy if you already know Java as I would say about 70 if not 80 percent of Java’s syntax are also present in Apex and implemented in a similar if not same way. In that way many of the readability, writability, reliability gimmicks present in Java are also present in Apex. Even if a reader does not know Java I believe the language does not have a high learning curve. The lack of operator overloading and pointers found in languages like C++ weakens writability but enhance readability. Like Java, Apex has the try/catch and the optional finally for exception handling, boosting reliability and writability. Although many things are similar between Java and Apex there are some apparent differences. For one thing, certain data types are missing in one language and is replaced by others, such as the Currency data type in Apex and the Character data type in Java. Furthermore, user input/output is practically non-existent in Apex and the language is case insensitive which is harmful for reliability and readability, although it does make writing the code easier at times.

More importantly however, are the features present in Apex that was added on to fulfill Salesforce client’s needs. One big difference is the existence of data manipulation language, or DML for short, in Apex. DML, and by extension, the Database class, were made to support database operations such as insert, update, and delete. While Java also support some SQL operations, it requires extra difficulties to perform said operations unlike Apex. Furthermore, since Apex was created to be used to support the client’s needs, it has built-in user interaction and form handling such as triggers as well as the ability to use Apex in external applications using SOAP API and Rest Architecture without the need for extra classes. Another interesting difference in Apex compared to many other object-oriented programming language, is the addition of the “with sharing” and “without sharing” keywords. Normally, paradigms that adhere to OOP will follow the rules of sharing, or enumeration, where one class or object cannot view or change other class’ private variables unless said class use getters or setters. Apex breaks that rule by allowing other class’ to view and change said variables by using system context. A class must use the “with sharing” keyword to force Apex to follow enumeration rules for that class. Lastly, unlike Java and other languages which require external compilers to run, such as JVM for Java, Apex runs entirely on Salesforce server so there is no need for any downloads or path/variable setup to be made. This, of course, presents other dilemmas such as the requirement of a stable yet fast internet connection.

In all honesty, we had trouble coming up with an application that was not too simple, but also contained the important and unique features found in Apex. We decided that the features that we had to highlight was database operations, user and form interaction, and external application support. In other words, use insert/update/delete, triggers, and either SOAP or REST API in our code. Our solution? A monopoly like game that would have ‘players’ (or in the case of our application, account objects), fight over dominance. Using REST API, we sent a GET request to the Force.com server with the requested number of players. The API would receive the request and, following OOP principles, create a game driver that implemented the game interface. In the driver, several players/accounts would be created with a set amount of money and inserted into the database. The players would take turns randomly taking or giving money to other players. After everyone’s turn is over and the rankings have been updated in code, the database would be updated using the update statement. Whenever the accounts were updated, the trigger involving accounts would activate and print out all player’s current rankings. After all the turns were passed the application would print everyone’s standings one last time, delete all the accounts involved in the game, and print the runtime. Although the application was overall simpler then I would have liked, it was an application that displayed the abilities of the Apex programming language.

As for my part of the project, I was originally in charge of coming up with the idea and implementing the application. After having some trouble coming up with what I thought would be an acceptable application we came up with the application described above. After finishing the application, I assisted my partner, Nandita, with finishing the power point. Nandita finished most of the power point and all I had to do was add in some extra details to some features I thought had to be described more in-depth. For example, I added the security features of Apex involving the “with sharing” and “without sharing”. I also thought it would be necessary to go just a little in depth to Apex’s method of parallel processing, or asynchronous programming. The only other detail that I added was how Apex allows external applications to run its codes using SOAP and REST API.

Teamwork wise I believe my partner, Nandita, and I worked very well together. We knew what needed to be done, what needed to be said, and who needed to do what for the project. During our group discussion we were easily able to come to an agreement on who was going to do the power point and who was going to do the application. While doing our individual parts of the project we would ask each other about what should be included in the application, and what should be included in the slides. During our practice runs we came to a consensus on what parts of the slides need to be covered in depth and which part of the slides needed to be skipped, skimmed, or deleted. We were also able to agree on who should say what during the presentation to ensure smooth transition. The fact we were able to agree to meet before our presentation date to practice is almost a miracle considering we were both busy college students; and I have almost never been part of a group that agreed to meet for a practice run, true story. Overall, I must say our teamwork was excellent with little to no conflict while finish the project.

Now onto what I learned and experienced during this group project. To begin with I should mention that after finishing this project, I realized that it was an overall enjoyable experience having to learn not only a new language but comparing the differences and similarities said new language to other languages I know. When I first learned of how similar Apex was compared to Java I had a little bit of dilemma since I thought, “this is Java”. It took some time to learn and understand the differences that the Apex developers added and changed between Java and Apex to suit their and their costumers’ needs. One of the surprising things I experienced was finding out that Apex, while being described as an object-oriented programming language, by default, ignored encapsulation rules. I had finished writing the application when I realized that the keyword “with sharing” was needed for the program to follow encapsulation rules. It was a pleasant surprise for me to find out and learn the unique features of Apex that Java does not have or is hard to implement in Java, despite how similar the two languages were. One of those differences was the database operations found in Java using DML and database class. I knew Java had some support for database operations, but I had been told that Java’s implementation was not the easiest, so I had skipped out on learning Java’s database operations. However, since Apex had database operations and since Apex was so like Java syntax wise, it was necessary for our group to compare the two programming languages including the different implementation of database operation. This requirement forced us to learn the different SQL operations on Java which was a something I did not expect to have when we first started this project. Another bonus experience was learning the database operations itself. I had never taken database as a class and only had basic knowledge on how SQL operations worked from YouTube videos. Java’s implementation was, as I was told, too convoluted for me to learn but Apex’s database operations were very easy for me to learn and implement. Furthermore, our group did not have to set up our own database since Force.com had its own database and Apex was already connected to said database from the start.

Another new experience for me was using REST API. I knew of its existence and our software engineering group used said API for our website using Java, but I was not the one who implemented the REST architecture, although I did touch briefly upon it as a homework assignment. Learning Apex forced me to understand how REST API worked and how to implement it in Apex for our application.

The project, while it had many enjoyable experience, was not all fun and games unfortunately. Triggers, while fun to play around with, provided its share of difficulties. During the development of the application, I ran into an obstacle where either the trigger would not activate, or the insert command would not be carried out. Several hours of debugging later I realized where the problem was, and the solution was changing both the trigger class and the game driver class so that the conflict could be resolved (The problem was that the game driver class could not activate the insert command while the trigger would activate after the insert). Another ‘fun’ experience was trying to use the cURL command line operation so that the REST API call could be activated via command line. I wanted to call it via command line to avoid having to use Workbench during the application demo. To begin with, the cURL command had to be installed on my Windows computer and correctly connected; after that came the fun part of connecting to SalesForce.com via cURL and making the correct REST API call. The entire process took between 4 and 5 hours and the quest ended up as a fool’s errand as I realized that setting up the cURL command would take too long to perform during the demo as the session key would change every time I logged into Salesforce. Thankfully Workbench still show cased the ability of REST API.

Despite everything however, including the difficulties, the project was still a valuable learning experience that taught me new things. A new programming language is important of course but it also reminded me of the things a programming language needed to do and have as a programming language, such as data types, paradigms, etc. and the differences between languages, no matter how similar they may seem. Furthermore, it taught me new skills such as SQL operations, and cURL. The fact our team worked well together was a extra bonus.