Design Decisions and Challenges

The first and most challenging decision I had to make was deciding what underlying data structure I was going to use to accomplish the goals of my program. Solely based on the tasks my program was going to use I knew I needed to have a binary tree structure. However, the Tree Map and Tree Set frameworks were great for searching for data but being red-black trees they self-level. This would not work for my program since the structure of the tree needs to stay the same.

Therefore, I decided to use a HashMap instead. This allowed me to specify integer keys and have low latency looking up data. To maintain the tree structure, however, I had to determine a way to calculate where left and right children are located. I used the equations below to construct keys I knew would not overlap, which would lead to collisions:

Another benefit to using a HashMap is that it is serializable, which means that the process of saving and loading data would be very simple. Creating save and load methods were pretty simple to begin with. However, later in my development process I decided that the Learning Tree class should also have data on its save location and subject. As such, instead of just saving and loading HashMap data I needed to save and load the entire object. Saving was not an issue, but when I loaded the object there was no way to directly load everything into the current object calling the Load method. Instead I had to use a temporary object and then get all of the properties and load them into the current object.

Similar to the Learning Tree data, I had to make a decision on how I wanted to store player and high score data. I decided early on that I wanted my game to represent an old school arcade game where users could reuse their name but have separate high score entries. As such, I choose to just use a simple Array List and then sort the list whenever a player’s score is updated. To save this data I just used the same save/load type method from Learning Tree.

Going along with the arcade game feel I also decided that if a high score file does not already exist to initialize it with fake player scores to give players a goal to achieve. My original proposal had the score be equivalent to the number of times a player “stumped the bot.” However, to make the scores more impressive I decided to increment them by 5 so that a high score of 100 would seem impressive but also not take too long.

I decided not to make separate windows for admin control and player control since the native way you teach the learning tree is to play with it. So instead I made an admin control button that is password protected so that an admin can play and then switch to admin to teach the learning tree when they like.

I did, however, decide to make custom windows that pop-up for creating a new game and for admin control of adding a new question to the learning tree. This was a challenge since I had to figure out ways to communicate between the two windows. I had started to do research in multi-threading to have my main program wait for the popup to finish. It was pretty complicated so instead I decided just to call methods between the two windows.