Ty Marking 949-274-0522

Mr. Gula

August 28, 2017 The Gula Residence

1:00pm-4:00pm

Job Shadow Mtg. #2

Of Scala and Long Nosed Dogs

Preface: Due to the nature of Jim Gula’s workplace (his home) and his employment status (retired), the content of this report won’t have much social or workplace exposition and instead would easily will fall almost solely in technical realm but I hope to keep any boring parts to a minimum. At least there’s Tyler, he’s really cool.

The first thing that came running out of Mr. Gula’s door to greet me was his beautiful dog Tyler accompanied milliseconds later by a few nautical miles of silky flowing hair. Tyler’s nose could make a fool of even the most adept artist’s depth perception, I could never quite fully grok its length. I continued to be surprised throughout the afternoon by just how far it protruded and how it gave Tyler an almost aquiline appearance. But enough about the one reader friendly relief, I have to save somethings about him for later right? Mr. and Mrs. Gula were most kind and homely in welcoming me emanating an aura of welcoming and homeliness. Having proceeded to the work den, we got down to business. Inside the rectangular room were the two main computers flanked by bookshelves most impressively filled with books form the art of sailing to medical books about concepts I didn’t even understand the name of to CS books spanning the ages. On the rightmost computer was the friendly face of a wall of definitions wrapped by the countless little icons that eclipse (the development software) never seems to want to give you the name of when you hover over them. And so began the conversation about Mr. Gula’s main project, and cross-platform accounting software for developing communities electrical grids written in Scala. Unfortunately, it wasn’t in a runnable state due to Mr. Gula being in the middle of the completely asinine but admiral task of writing his own custom version of HTML. Now this may sound like mealy a time intensive project, just write some interpretations for each of the different elements and tags and linearly run through the input displaying the output on the screen in progressively recursioned blocks, easy right? Even if that was the case it most certainly would not be easy, but that isn’t even the case. Because each HTML element must be displayed differently based off the available space and what comes after it and how much space that would leave for the next element and what the CSS file says and the user preferences and a million and one other things, things get crazy fast. To skip over a lot of technicality, Mr. Gula explained how he was doing a fancy thing (using a state machine) and a useful feature in Scala (tail end recursion) to solve the problem in a dynamic way. Tail end recursion was something that when I saw what was happening I instantly thought of all the times I could have used it and how it can make problems that are seemingly impossible due to memory constraints almost trivial. I am even considering switching a possible upcoming project of mine from java to Scala for this feature. Basically, instead of having a possibly exponential amount of function (a figurative black box that takes an input and returns an output) awaiting a return in the stack (memory), it allows for them to be popped off and their expected return to be routed directly to where they were called from. If function A calls function B­­passing it an input, if function B is a traditional recursive function, it might end up calling n instances of itself, then calling function C and passing the value returned from C back up through each call of B all the way to A. So basically A→B1→B2→B3→…Bn→C for a tradition recursive function. With tail end recursion, the middle steps (all the B calls) are popped off and the output of C would be directed directly back to A. In symbol form: A→B1→B2 but then B1 would be popped off and you’d be left with A→B2, then B2 calls B3 or A→B2→B3, B2 pops of, A→B3 and on and on until you have A→Bn→C which is boiled down to A→C. If n is a stupidly large number, with traditional recursion, the stack (memory) would become filled up with all these references to functions and where they are supposed to return to that a Stack Overflow error is thrown. With tail end recursion, the stack of functions never grows large and memory is conserved. There are some technical downsides in how you must write the function and it is not supported on every platform, but when there is the opportunity for tail end recursion to work, it works very well. Anyways enough technical jargon, did I mention how long Tyler’s nose (or is it a snout?) is? It seemed to be the length of my forearm. I mean that dog could reach the bottom cookie in the tallest of cookie jars.