# Scala

I am personally thrilled by Scala, a hybrid language which is the bridge between two programming paradigms. That is the connection between OOP (Object-Oriented Programming) and functional programming. I am intrigued to witness Scala running on the Java virtual machine where it runs side-by-side with existing deployments. Although I am not a Java programmer, I think Scala's syntaxes are easy to grab and use. Ability to apply functional programming ideas to my code, I guess, is really helpful. I actually loved the use of "VAR" and "VAL" for defining mutable and immutable variables respectively. Because this will make our code more readable, and implementation of functions would be easier too that will also help us to be on the safe side of our program. I believe that the biggest problem faced by concurrency-minded programmers in OOP today is the mutable state which Scala solves to some extends through its functional programming paradigm. Meaning, functional programming helps resolve these issues by eliminating mutable state from the program. Like Ruby and IO, everything is an object in Scala, with some small exceptions. As I have observed the use of tuples in functional programming, like Prolog, Scala also offers tuples. Unlike Ruby, I actually loved the way Scala's method definitions have parameter types and names which will make us easier to understand the code. While addressing about collections types in Scala, I truly liked the use of Sets, Lists, and Maps as well as interesting Anything/Nothing class. Need to mention, I am fascinated to notice the support of higher order functions in Scala. In addition to that, I absolutely enjoyed the use of anonymous functions (also called closures) such as .filter(), .reduce(), .map(), .foldLeft() and many more. These functions not only treated me to create immutable state but also made my life (as a programmer) easier as we could replace many lines of code blocks in one or two sentences. Refactoring is so easy through Scala which will better design our code for concurrency. Clearly speaking, Scala's flexible syntax and operator overloading made it an ideal language over Java and other languages too.

Although Scala's local type inference generally works well but I consider it has limitations too. In Scala, I was not able to use 0 or Nil at all as they are the substitute for neither True nor False which I perceived quite weird. I similarly discovered Scala's syntaxes demanding, academic and hard on the eyes whilst correlating to that of other languages syntaxes. Sometimes, I assume that the use of both mutable and immutable states sum up to concurrency bugs.

I, in conclusion, think that I would use Scala in terms of concurrencies, pattern matching, and core designing features.