CS 202 EFFIECIENCY WRITE-UP FOR PROGRAM 4

BY TEJAS MENON; ASSIGNMENT BY PROFESSOR KARLA FANT

This program was a great introduction to the foundations and underlying principles of Java, and it helped build memory and experience with the 'almost obnoxious' assistance of an IDE. In the beginning, the IDE's distracting overlays, formatting and dynamic pop-ups were almost unbearable and I even considered switching back. But once a lot of the program was written and functions and references were numerous, it's assistance for every keyword became indispensable for saving time. I fould myself speeding up at the keyboard very quickly and once some shortcuts of moving around the IDE became second nature, I found the IDE very helpful and empowering.

Additionally, it's advantages for debugging were almost ten fold that of other debuggers, as the learning curve and specificity of the error messages were vastly improved. I also noticed that the shortened time span for writing a program that would have taken longer testing and understanding on vim, made my hours at the program more focused and effective, and I could sit a longer stretch without having to ponder about other non-working aspects about the program. The time for checking of compile time errors and rectifying any simple logical mistakes became almost negligible, and I could afford to forget the syntactical basis of any statement, and rather approach a problem through a higher level of thought. Java in this realm brought further ease to writing a program, as the complete scrapping of the requirement of pointers, the task of passing in or returning by reference and the absence of the delete operation made the entire codebase more readable, maintainable and simplistic. Copying of strings was also not required, as they acted immutably almost like a primitive type, and you could assign one to another without risking data corruption. This made writing code a very instinctive process, and I didn't get lost in changing my style of implementation depending on the syntax, and rather I could mend the syntax to meet my needs. Another such example was returing the 'this' reference from the feature class and having it downcasted in 'car' at runtime. This would not have been possible in C++, as the 'this' pointer would have pointed solely to the base class, and downcasting would have invoked errrors.

Additionally, the static features of any class were extremely helpful in troubleshooting as I didn't need to leave the program to test something quickly, and rather I could quickly build a new static main and run it in the context of the program itself. Java was also a highly documented language, more so than C++ I felt, and the assistance and wide swathe of available operations, libraries and features made it highly versatile, and easier for performing quick higher level operations. Even though this has few drawbacks of not understanding a function or implementation very fully- as it isn't you that made it- it makes good progrmmers those who are familiar and experienced with the common libraries of Java rather than simply being technically fluent with syntactic Java- which isn't very complicated at all. The learning process I found was very different in C++ and Java, in C++ I was looking at ways to make-do with the functions and features I had at disposal, while with Java I was looking at classes, their derivatives, their methods, fields and implementation, and it became a learning of how to work with these overlaying objects or 'robots' rather than simply with basic concepts and functions. The entire hierarchy of the stream objects was very interesting to me- as we had a simiar hierarchy in C++ but it wasn't very accessible or understandable- and I found myself learning better

by looking at the code itself rather than vague/non-specifc information about them being provided to me.