**CSE 401 Project Report, Autumn 2024**

AM
Vova Trifonov, Jaechan Lee
vladtrif, chan0369

**What languages features work**

We completed all the requirements mentioned in the assignment instructions. This includes arrays, if-else statements, subclasses, method overriding, etc…

**What language features weren't implemented, don't work, or don't quite work**

We did not attempt anything outside of the requirements mentioned in the assignment instructions. For example, we did not attempt to implement instanceof, casting, method overloading, nested classes, interface, abstract class, String types, etc…

**A summary of the test programs you've tried and the results**

In previous checkpoints, we mainly just tested our code on the SampleMiniJava programs that are given to us and visually checked the string output by the scanner, parser, type-checker, etc. From the feedback we had gotten previously, our compiler tended to accept too many programs, obviously because we hadn't extensively tested on the ill-formed programs. After each grading, we revisited our graded implementation and fixed everything by testing our compiler against ill-formed programs.

For the code-generator, we first got System.out.println working, and while debugging, we tried to isolate each feature that is likely to cause bugs. In the end, we compiled and ran all the SampleMiniJava programs. The actual output produced by our compiler and the expected output compiled by Java exactly matched.

**Any extra features**

N/A

**How work was divided**

We pair-programmed in the CSE lab together so we did the entire part of this project together in person.

**Conclusion**

Overall we are happy with the basic functionality of our compiler, however our code could've been organized better. We didn't design our APIs in a way that is anywhere close to compatible with real Java, which albeit wasn't strictly required, could have been something to strive for. For example, we exploited the fact that there can only be three nested levels of symbol tables by creating a separate class for each of them, such that adding support for nested classes would require re-structuring all of our APIs. In general, we think our APIs are functional but not clean; we would have revisited them if we had more time.

We also could have thought of more systematic ways of testing, as most problems that arose in previous checkpoints could have easily been caught if we tested our program more thoroughly.

Aside from the language features, there were places where our generated code could have been optimized. For example, our generator fully evaluates all the booleans in the conditionals instead of utilizing a control flow; and has lots of push & pop operations to simplify having to keep track of registers (in particular, we don't use the last five registers conventionally reserved for passing arguments).