CSE 401 Project Report, Spring 2018
BA
Sarah Oslund, Johan How
sfoslund, howj

What language features work?
All of the main language features work.

What language features weren't implemented, don't work, or don't quite work?
As far as we know, none, but we have had problems with missing edge cases in previous portions of the project.

Summary of the test programs we've tried

For each part of the project, we tested it on all the sample MiniJava programs in the SamplePrograms folder. We also wrote our own test cases, putting them in the OurSampleProgram folder. We tried to write an exhaustive test suite for our programs, mainly using clear-box testing, trying to hit every case possible. For example, in the code generation part of the project, we wrote 12 different test cases for extended classes, each testing a different part of our compiler including dynamic dispatch, parameters, and object fields.

Any extra features or language extensions?
None

How was work divided?

For each part of the project except for part V, we met up early to discuss the project part and how we would split up the work. We mainly split up the work, occasionally there were some things we worked together on.

Scanner
We worked together on both the .jflex and .cup files.

Parser
Worked together on the .cup file and to get the AST to print to the terminal the correct format.

Semantics

Johan: checking for duplicate names. Checking that a class does not extend itself, directly or indirectly. Methods are called with correct number of arguments, method in a subclass overrides check.
Sarah: Building symbol tables, calculating type information for nodes, checking components of expressions/ statements are the same type (for &&, +, etc) or the right type (for if, while, etc)

Codegen
Johan: Initial setup of this part of the project, and figuring out how to run gcc with boot.c to generate .s files. Integer Expressions & System.out.println, Control Flow, some of Variables, Parameters, & Assignment.
Sarah: Objects (including creation, vtables, and fields), Method calls, parameters, variable assignment, class/ instance variables, extended classes, booleans, and arrays

Additions
Worked together on ADDITION-NOTES.
Johan: extending the scanner, parser, and semantics of our compiler to work with double.
Sarah: Extending the code gen part of our compiler to work with double

Brief conclusions

What was good?

We had good expectations as a team. We were able to stay on schedule, and submit our projects before they were due, not using any late days. We had good communication about our progress on our respective parts of our project. In general we worked together well and were able to split the work between ourselves fairly.

What could have been better?

Our testing could have been better. We failed to identify some edge cases in the parser and semantics parts of our projects. Specifically, we had problems with the parser part of the project because there were many edge cases that we did not think to test for. These problems were persistent throughout the whole quarter meaning in pretty much every part of the project we were docked points for something wrong with our parser, we would fix it, and then on the next part we would be docked points again on another problem with the parser.

What you would have done differently or would have liked to have seen changed about the project?

Something we would do differently is to write JUnit test suites for each part of the project, so running tests would be more convenient and we would be sure not to miss a case when we updated our project. This could have saved us much effort and time in the long run. In addition to this, we could have written more black-box tests, as we missed some edge cases we might have identified otherwise. In terms of changes to the project itself, it would have been nice if problems with previous portions of the project did not play such a large part in the grading of subsequent parts. It was frustrating that for every part of the project, about half of the points we got off ended up being due to problems with our parser. Particularly with the code generations portion, when we went back to fix the problems that were identifier we only had to add about 4 lines to our code generation visitor (which accounted for about 6 points we got off) but we had a lot more work to do on other parts (which accounted for the other 11 points).