CSE 401 Project Report, Spring 2023
AN
Kyle Mumma, Ashwin Banwari
kmumma1, ash1152

What works:
We implemented everything in the spec to a point that we thought works, from testing various programs using these features, including:

- integer expressions
- prints
- object creation, vtables, object fields
- methods, local variables, method calling on objects, parameters, assignment statements
- control flow, while loops, if statements, boolean expressions in the context of evaluation. All our booleans get evaluated even for conditionals, but we stop evaluation short when possible.
- classes and instance variables, extended classes, method overriding and dynamic dispatch
- arrays, array assign, lookup, bounds checking

We were able to get implemented and do basic tests for pretty much everything that is on the spec and we thought we needed. And we were able to successfully compile and run the following minijava programs, comparing their output to the java compiler to verify correctness.

works, outputs compared against java compiler:
BinarySearch.java
BubbleSort.java
Factorial.java
LinearSearch.java
QuickSort.java

what doesnt work:
~Everything~ does not work however, as with most code. When we first finished and ran more tests we encountered more bugs that we had to fix as the tests got more complex. Many we were able to solve before the deadline but not all. We are not able to run the following minijava programs, they all compile, but segfault at runtime.

doesn't work, segfault:
BinaryTree.java
LinkedList.java
TreeVisitor.java

We are not sure what the cause of the errors in these cases are. It would require stepping through the programs to figure out what is causing the errors.

testing:
As discussed above we mainly tested features as we implemented them, in the various cases we could think of. This is usually where we caught all our bugs. Once we had everything implemented we ran the sample programs given to us for exhaustive testing. It likely would have been good to create permanent .java files for our different tests, or perhaps do more test driven development, but we didn't.

extra features:
We didn't really implement any extra features than what was required. We added bounds checking at compile time to arrays, for index out of bounds errors, but I think this was required. We also formatted our assembly labels, method labels, and vtables in a decently human readable way.

division of work:
We mostly worked independently on separate tasks, collaborating when necessary if things overlapped or required parts that the other worked on. Collaboration was a bit more difficult since Kyle fell ill during the project, but we were able to figure it out.

Ashwin did most of:
- initial integer expression evaluation and printing
- method calling, method parameters, method local variables, this
- assignment statements
- class fields

Kyle did most of:
- object creation with new
- class vtables, extended vtables, dynamic dispatch
- control flow: if, while, boolean expressions
- arrays: assign, lookup, length, bounds checks

Most of the debugging and fixes were done by both of us, fixing issues as we found them.

conclusions:
I think we had a good division of work, it allowed us both to get broad exposure to the different parts of the compiler and have good collaboration skills. I think our testing was ok but could have been better.

We could have formatted testing in a more structured way to help guide us more, such as test driven development, or at least saving all of our tests, rather than testing as we go and never saving any of them. We also should have started the project earlier, we didn't start until fairly late and around that time Kyle got sick with covid which completely interfered with our ability to finish in time. Luckily with the extension Hal gave us, we were able to finish most of the major features. But still we didn't have enough time after feature implementation to do exhaustive testing and debugging, which is why not all of the programs were compiling. If we started just a

few days earlier, we likely would have been able to successfully get all the sample programs working.

Additionally we could have structured our assembly generations a bit better, when going back to old assembly generating code, it takes a long time to decode what it's doing, especially when it gets long. We could have figured out a way to do this better.

We really liked this project checkpoint, it was both of our favorites from the entire project, but here are our thoughts. We think that this is the most fun, but also the most difficult checkpoint. We think that the time and difficulty of this checkpoint could be stressed more, perhaps even split up into two similar to how semantics was done. In fact we thought this checkpoint took longer and was more difficult than semantics, so we were surprised that it wasn't split up while semantics was.