Lettuce Wrap: An Interactive Language Interpreter

Reece Suchocki, Spencer Wilson

May 3, 2023

1 Final State of System Statement

In our final iteration of Lettuce Wrap for CSCI 5448 Object-oriented Analysis & Design we present several new features. On the front-end, we were able to implement drop-down menus for a user to switch between scoping conditions, type conditions, and lazy/eager conditions. We've built a drawer component that contains our complete grammar for using the tool and improved the user interface layout. On both front-end and back-end, code documentation has been completed with references and helpful code comments. On the back-end, we have added a binary sequence operator (e1 ; e2), identified OO patterns with an **OO PATTERN** keyword, and addressed a dearth of TODO comments.

Scoping conditions: *Lexical / Static - Dynamic*
Type Conditions: *None - Implicit*
Lazy / Eager Conditions: *Lazy - Eager*

To name a few features which are not yet implemented, we would like to expand upon the expression identification or highlighting of partials. We attempted this in encoding highlights within our JSON structure but struggled with the logic. We hope to pick this feature back up in future iterations of the total product. Relating to 3155, the tostring() and string concatenation have extra parenthesis on the string literals when rendered in our output boxes. This is strictly cosmetic, but still something we hope to address as we continue development of the Lettuce Wrap for CSCI 3155.

2 Final Class Diagram and Comparison Statement

Our final class diagram can be seen in the index section of this document here: 5. To view our original UML class diagram for project 5 on Lucid Chart, click *here*. To view our final UML class diagram on Lucid Chart, click *here*. Some exciting key changes in our diagrams include a refined definition of the View and ViewController. The addition of a bianary sequence operator (e1 ; e2)

are show in the bottom center with green markings. Additionally, we see the introduction of more clases to manage the interpreter logic and to provide a more maintainable and extensible code base. The drop down collection, seen top-right in green, and also the Project 6 introduction of error classes. One additional pattern which should be noted, but not captured in the class diagram, as it exists in our testing framework, but we do use the simple factory pattern.

The following patterns can be identified in our UML:

1. MVC-ception: React.js has it's own MVC pattern in the framework. We also separate our code into an MVC architecture.
2. Composite pattern for the expr data structure.
3. Interpreter pattern integrated into the composite pattern for expression evaluation.
4. Template pattern implemented to integrate with the step-function of the interpreter to encapsulate variance in language semantics to relevant evaluation condition as needed while leaving core logic of the interpreter in-situ where possible.

3 Third-Party code vs. Original Code Statement

Our Scala back-end code is inspired by various iterations of the CSCI 3155 course offered at CU Boulder (Principles of Programming Languages) from 2015 - 2023. The parser is mostly a rewrite of code available at LettucePlaygroundScala and pppl-lab5. The remainder of the work is original authorship and varies in novelty. The interpreter with lexical scope, implicit type conversions, and eager evaluation is a re-architecture of code authored for that a version of the course self-authored and not plagiarized.

The syntax is largely borrowed from the PL course, Fall 2023, version. Small step semantics for dynamic scope at function call site is a novel effort demonstrating past knowledge. This also inspired the novel development to parse all functions as a closure from the start, while it's not valuable in the lexical scoping version, it is there for multi-paradigm evaluation on the same initial parsing of the expression. The program architecture to use the template pattern, optimize code reuse, and write an interpreter flexible to many different semantics/evaluation conditions is novel work.

Great Scala Tutorials Which Helped Us (hyperlinked):

1. Learn Scala
2. csci3155_notebooks

3. cats (for the truly dedicated)
4. csci3155-notes
5. Programming in Scala - fifth edition (Martin Odersky)

Great Tutorials Relevant to our Front-end (hyperlinked):

1. React Grid
2. React Drawer
3. React Select
4. React Button
5. Using the state hook
6. Classes - JavaScript

Several of the above links are explicitly referenced within our code. When using Material-UI template components, changes made are noted with proper citations. To initiate the react boilerplate project and Scala Play framework, we exectued 'npx create-react-app react-app', and 'sbt new playframework/play-scala-seed.g8'. This created many of the backgrounds Scala and React files which we built on.

4 Statement on the OOAD Process Overall

List three key design process elements or issues (positive or negative) that your team experienced in your analysis and design of the OO semester project.

1. **Green-red testing.** In developing the back-end, we intentionally performed some TDD. Authoring a test, then coding the implementation to pass that test without breaking our other tests. This was a solid experience. Unfortunately, we got complacent and did not keep this process throughout the full implementation of the back-end, so our test coverage is somewhat lacking. We enjoyed this process.
2. **OO Pattern Design Ideation.** We used the project 5 framework described in class. We used all the different diagrams and used them to cross-pollinate ideas and go deep into our design before authoring much code at all.
3. **Design & UI Sketching.** We opened up the conversation to our family and friends as to what a good UI would look like for the Lettuce Wrap. We went through a fair amount of sketching and design before we decided on the application layout. We did well here to not stick too hard to one design in particular and kept evolving the interface as requirements broadened.

5 Class Diagram

Click [HERE] to view full UML class diagram on Lucid Chart, or see next page.

Project 5 OOAD_Semester_Project Class Diagram

Spencer Wilson & Reece Suchocki | May 3, 2023

Project 7 OOAD_Semester_Project Class Diagram

Spencer Wilson & Reece Suchocki | May 9, 2023