Names: Iris Lian, Zeb Keith-Hardy, Michael Li
Course: CS461
Date: February 24, 2019

Project 12: Enhance the IDE so that it can perform semantic analysis of the AST

Overview

In this project we focused on implementing the SemanticAnalyzer class. This class deals with the semantic error that are not discoverable by parser. The Semantic Analyzer is consisted of five essential pieces: 1)Add build-in class to the classMap 2) add user-defined classes and build the inheritance tree of ClassTreeNodes 3) build the environment for each class (add class members only) and check that members are declared properly 4) check that the Main class and main method are declared properly 5) type check everything. The majority of the task is in step 2), 3) and 5) since the step 1) is already implemented and step 4) is an re-iteration of project 11 tasks.

Beside these five steps, we also needed to setup the buttons and errorhandler so that the semantic errors are printed out in console in an elegant way.

Implementation

For this project we exclusively used visitors to accomplish all of the tasks. To generate the ClassMap we used a InheritanceTreeVistor to place the classes in the map and check for cyclic inheritance. To generate the environment for each of the classes we used and EnvironmentBuilderVisitor which created the symbol tables for each of the classes. To check the main method we used the MainMainVisitor from last project but modified it so that it would also check the inheritance tree for the Main class. Finally of course we used the TypeCheckerVisitor to type check pretty much every node in the AST.

Elegance and Inelegance

This project is elegant because we strictly followed the Visitor pattern when implementing analyze method in SemanticAnalyzer class. Each visitor has an intuitive name that indicate its job in the SemanticAnalyzer.

We also continue to use threading in our GUI when running our growing compiler. This meant creating another inner class in the ToolbarController, CheckTask, which is responsible for generating the class tree and returning it so that later we can use it.

Division of work

For the division of work we split into two groups: Michael and Iris implemented the InheritanceTreeVisitor and the EnvironmentBuilderVisitor and Zeb implemented the MainMainVisitor from project 11 as well as the TypeCheckerVisitor. The group also worked together to debug the TypeCheckerVisitor in step 5) which is the most complicated and work-intensive class in this project.