CSCI 4160 Project 6

Due: see class calendar

Goal:

- To perform type checking on the abstract syntax tree, which represents a COOL program.

Description:

In this assignment, you will implement the static semantics of COOL. You will use the abstract syntax trees (AST) built by the parser to check that a program conforms to the COOL specification. Your static semantic component should reject erroneous programs; for correct programs, it must gather certain information for use by the interpreter. The output of the semantic analyzer will be an annotated AST for use by the interpreter.

Typical semantic errors that should be detected in this assignment are:

- Type/Class is not defined
- Types of operands are not compatible
- Argument of 'NOT' has type other than Bool
- Variable/Identifier is not declared
- *new* used with undefined class
- Type of initial expression doesn't match with declared type
- Cannot assign to self

Each type of node in AST has a method called tc_student() in TypeChecking_student.cpp file. In this method, you need to process every language construct contained in this node and report any static semantic errors.

The instructor provides the solution in a library. During the development, if you want to use teacher's solution in some methods so that you can focus on one specific method, you can use *teacher_version* instead of *student_version* in typechecking.cpp file.

Set up environment:

In addition to Visual Studio solution for Windows platform, the project also provides one for Linux platform (i.e. ranger.cs.mtsu.edu). Using the following commands to compile the project in Linux:

```
--work on TypeChecking_Student.cpp file
make                    --compile your project using make command
./main bad.cl      --run your program against bad.cl
```

If you want to use MacOS for the project, you can download the Linux version. You may need to change "g++" at line 9 of **makefile** to the compiler you want to use. The compilation and execution should be similar to the Linux version.

Tips for the project:

- Each type of expression node in the AST has a new member data: ***type***, which should store the actual type of the expression after type checking. Please update ***type*** during the type checking of these expression nodes.
- How to check if a type is Int, Bool, or String?
  In TypeChecking_student.cpp file, several global variables are declared and initialized in method *initialize_constants*. These global variables represent built-in classes, their methods, and special names like 'self', 'self_type'. So to check if a type is Int, Bool, or String, you can compare type with the corresponding global variables representing Int, Bool, or String like the following:
  type == Int or type == Bool or type == Str.

- How to report a semantic error?
  You can always use semant_error() method to report a semantic error like the following:
  ```
  env->semant_error(this) << "error message like Static dispatch to undefined class " <<
                          type_name << "." << endl;
  ```
  where type_name is a variable. The usage of env->semant_error(this) is pretty much the same as cout.

Instructor provided files in the class repository

The following files are provided by the instructor:

- Skeleton source files provided in the sample project are listed below:
  - ErrorMsg.h: contains the definition of error handler
  - Dump.cpp: used to print the abstract syntax tree.
  - Absyn.h: contains class definition for all AST node types.
  - AbsynExtension.h: contains extensions to nodes defined in AST
  - StringTab.h and StringTab.cpp: contains definition of string table.
  - main.cpp: the driver
  - TypeChecking.cpp: the file of deciding which version is used: teacher's or student's.
  - TypeChecking_student.cpp: This is the file you should work on.
  - Semant.h: Contains class definitions you may need for this project.
  - SymbolTable.h and SymbolTable.cpp: definition and implementation of symbol tables
  - bad.cl: COOL program that contains semantic errors.
- Description6.pdf: this file
- Rubric6.doc: the rubric used to grade this assignment.
- bad.txt. sample output when type checking bad.cl file

How to submit

Please submit TypeChecking.cpp and TypeChecking_student.cpp only to D2L dropbox.