CSCI 4160 Project 7

Due: see class calendar

Goal:

- To develop a COOL program interpreter by evaluating the abstract syntax tree, which represents a COOL program.

Description:

Similar to project 6, you need to implement interp methods for each type of expressions. The following changes has been made:

- class ValueEntry: classes to hold different types of values
- class Environment
  - `SymbolTable<Symbol, ValueEntry>  var_table;`
    `Var_table is now defined to hold value, instead of type information for each variable`
  - `ValueP var_value_lookup(Symbol s);`
    `look up the value of a given variable s`
  - `ValueP var_value_probe(Symbol s);`
    `check the value of a given variable s in latest scope`
  - `void var_value_change(Symbol s, ValueP v);`
    `Modify the value for the given variable s`
  - `EnvironmentP instantiation();`
    `Duplicate the environment`
    `deep copy of var_table, but shallow copy for all other member data`
- class InheritanceNode
  - `EnvironmentP instantiation();`
    `Create an environment for an object from environment of the class`
  - `EnvironmentP getEnvironment()`
- class ClassTable
  - `ValueP interp();`
    `For each class, using provided initial value to initialize attribute in Environment. Find Main class, then find main method and interp main method.`
- Interpreter_student.cpp: All your implementation should go in this file.
- Interpreter.cpp: You can modify this file to decide which version: student or teacher version should be used to test your solution.

To debug your solution, you can come up with your own COOL program to contain features you want to test. You can also test your interpreter against COOL sample programs provided in class repository.

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

- How to create an object from a class?
  First, get the InheritanceNodeP from the class name by looking up class symbol table, and call InheritanceNode::instantiation() to create an object from it. Please be noted that an object is represented by an Environment object.

Instructor provided files in the class repository

The following files are provided by the instructor:

- Skeleton source files you need to know:
  - 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
  - Interpreter.h: the file defining values of variables
  - Interpreter.cpp: the file of deciding which version is used: teacher's or student's
  - Interpreter_student.cpp: This is the file you should work on.
  - Semant.h: Contains class definitions you may need for this project
  - good.cl: provided test file
- Description7.pdf: this file
- Rubric7.doc: the rubric used to grade this assignment.

How to submit

Please submit Interpreter.cpp and Interpreter_student.cpp only to D2L dropbox.