VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

PRINCIPLES OF PROGRAMMING LANGUAGES - CO3005

ASSIGNMENT 3

Static checker

HO CHI MINH CITY, 01/2023

ASSIGNMENT 3

Version 1.1

After completing this assignment, you will be able to

- explain the principles how a compiler can check some semantic constraints such as type compatibility, scope constraints,... and
- write a medium (300 - 500 lines of code) Python program to implement that.

1 Specification

In this assignment, you are required to write a static checker for a program written in MT22. To complete this assignment, you need to:

- Read carefully the specification of MT22 language
- Download and unzip file assignment3-initial.zip
- If you are confident on your Assignment 3, copy your MT22.g4 into src/main/mt22/parser and your ASTGeneration.py into src/main/mt22/astgen and you can test your Assignment 2 using MT22 input like the first three tests (400-402).
- Otherwise (if you did not complete Assignment 2 or you are not confident on your Assignment 3), don't worry, just input AST as your input of your test (like test 403-405).
- Modify StaticCheck.py in src/main/mt22/checker to implement the static checker and modify CheckSuite.py in src/test to implement 100 testcases for testing your code.

2 MT22 Semantic

Most of semantic description has been given out in MT22 Specification. However, some of them are discussed later because of confusion between two phases: syntax analysis and static checking. The below description will be used in assignment 3 and 4.

2.1 Implicit conversion of `integer` and `float`

In MT22, using integer for float alternatives is allowed but the opposite side is not. For example:

```
1 a: float = 1;
2 b: float = a + 2;
3 c: integer = 2.3;
```

In line 1, integer literal 1 implicitly conversed to 1.0 (in float). In line 2, integer with value 2 automatically changed to 2.0 when adding to float variable `a`. However, the assignment in line 3 is not allowed.

Every operator (in section 6 - MT22 Specification) which works with two types: integer and float has to follow the rules:

- If at least one of operands has the type `float`, the result type will be in `float`.
- Otherwise, the result type of expression will be in `integer`.

2.2 Declarations with `auto` type

- A **variable** of type `auto` indicates an automatic type which is to be inferred by the value given on the right hand side.

  For example, in the following code, `a` is of type `integer`, `b` is of type `string`, and `c` is of type `boolean`:

  ```
  a: auto = 10;
  b: auto = "hello";
  c: auto = a < 100;
  ```

  So that, the initialization of variable declaration is required if the type is **auto**.

- A **function** has the return type of `auto` which is to be inferred by its first usage (function call in expression or in call statement).

  For instance:

  ```
  foo: function auto (a: integer, b: integer) {}
  ```

  There are three cases of usage:

```
1 a: float = foo(1, 2);
2 b: integer = foo(1, 2) + 1;
3 foo(1, 2);
```

  - Case 1: The return type of foo will be inferred by the type of left hand side type, `float`
  - Case 2: Each operator works with the same types of operands so the right side of `+` which is in `integer` leads the return type of function `foo` to be in `integer`.
  - Case 3: The `foo` in call statement will be inferred as a procedure in `void` type.

2.3 Inheritance features

In MT22, a function (called **child function**) can inherit some parameters which could be declared with the keyword `inherit` from another function (called **parent function**). This inheritance is single but hierarchical. In function's body, it can activate the call to parent function with `super(<args-list>)` or prevent automatically prevent default activation by calling `preventDefault()`. These rules are as follows:

- If the parent function has a non-empty parameter list, the first statement in child function must be `super(<args-list>)` with the list of parameters is suitable for calling the parent function or `preventDefault()` to prevent the activation.
- Otherwise (an empty parameter list), the activation of parent function is implicit. To stop this automatical calling, the first statment in child function must be `preventDefault()`.

With the parameters with keyword `inherit`, the child function inherits them in its scope, so the redeclaration will be invalid.

3 Static checker

A static checker plays an important role in modern compilers. It checks in the compiling time if a program conforms to the semantic constraints according to the language specification. In this assignment, you are required to implement a static checker for **static scope** MT22 language.

The input of the checker is in the AST of a MT22 program, i.e. the output of the assignment 2. The output of the checker is nothing if the checked input is correct, otherwise, an error message is released and the static checker will stop immediately.

For each semantics error, students should throw corresponding exception given in StaticError.py inside folder src/main/mt22/checker/ to make sure that it will be printed out the same as expected. Every testcase has at most one kind of error. The semantics constraints required to check in this assignment are as follows.

3.1 Redeclared Variable/ Parameter/ Function

The declaration must be unique in its scope. Otherwise, the exception `Redeclared(<kind>, <identifier>)` is released, where `<kind>` is the kind (`Variable`/`Parameter`/`Function`) of the identifier in the second declaration.

3.2 Undeclared Identifier/Function

- The exception `Undeclared(Identifier(), <identifier-name>)` is released when there is an identifier is used but its declaration cannot be found. The identifier can be a variable

or parameter.

- `Undeclared(Function(), <function-name>)` is released if there do not exist any function with that name. The function usage (for inheritance or invocation) could be allowed before its declaration.

3.3 Invalid Variable/Parameter declaration

- The exception `Invalid(Variable(), <variable-name>)` is released when a variable is declared in type `auto` without the initialization.
- The exception `Invalid(Parameter(), <parameter-name>)` is released when a parameter is redeclared in a function with parent function having the same parameter in name with inherit keyword.

3.4 Type Mismatch In Expression

An expression must conform the type rules for expressions, otherwise the exception **TypeMismatchInExpression**(<expression>) is released. The type rules for expression are as follows:

- For an array subscripting (index operator) E1[E2], E1 must be in array type and E2 must be a list of integer.
- For a binary and unary expression, the type rules are described in the MT22 specification and implicit conversion in subsection 2.1.
- For a function call <function-name>(<args>), the callee <method name> must have non-void as return type. The type rules for arguments and parameters are the same as those mentioned in a procedure call.

3.5 Type Mismatch In Statement

A statement must conform the corresponding type rules for statements, otherwise the exception **TypeMismatchInStatement**(<statement>) is released. The type rules for statements are as follows:

- The type of a conditional expression in an if/while/do-while statement must be boolean.
- In for statement, the type of a scalar variable, update expression must be integer.
- For an assignment statement, the left-hand side can be in any type except void type and array type. The right-hand side (RHS) is either in the same type as that of the LHS or in the type that can coerce to the LHS type.

- For parameter passing, the rule for an assignment is applied to parameter passing where a parameter is considered as the LHS and the corresponding argument is the RHS.
- For a return statement, the return expression can be considered as RHS of an implicit assignment whose LHS is the return type.

3.6 Break/Continue not in loop

A break/continue statement must be inside directly or indirectly a loop otherwise the exception **MustInLoop**(<statement>) must be thrown.

3.7 Illegal Array Literal

The exception **IllegalArrayLiteral**(<array literal>) must be thrown unless all literals in an array literal must be in the same type.

For example, `{1, 2.0}` is an example of this error.

3.8 Invalid first statement

In inherited functions, the first statement follows the rule in subsection 2.3, otherwise, the exception `InvalidStatementInFunction(<function-name>)` is released.

3.9 No entry point

There must be a function whose name is **main** without any parameter and return void type in a MT22 program. Otherwise, the exception **NoEntryPoint()** is released.

4 Submissions

This assignment requires you submit 2 files: StaticCheck.py containing class StaticChecker with the entry method check, and CheckSuite.py containing 100 testcases.

File StaticCheck.py and CheckSuite.py must be submitted in "Assignment 3 - Submission".

The deadline is announced in course website and that is also the place where you MUST submit your code.

5 Plagiarism

You must complete the assignment by yourself and do not let your work seen by someone else. If you violate any requirement, you will be punished by the university rule for plagiarism.

6 Change log

- Void type with call statement.