Especificaciones de reglas semánticas c/. nomenclatura

Regla Semántica	Descripción
Program Structure	Programa.val := Clase_1.val Clase_2.val ... Clase_n.val
Main Class	Programa.val := Clase_Main.val Clase_2.val ... Clase_n.val
Main Method	Clase_Main.val := atributos métodos método_main()
Inheritance	Clase_Main.val := atributos métodos
Basic Types	Clase.val := "Int" atributos métodos | "String" atributos métodos | "Bool" atributos métodos
Scope	Atributo.val := "Int" id | "String" id | "Bool" id
Identifier Definition	Ámbito.val := identificador_1 identificador_2 ... identificador_n
Method Overriding	Clase_B.val := Clase_B_atributos Clase_B_métodos método_A()
Inheritance Limitations	Clase.val := atributos métodos
Default Values	Objeto_Int.val := 0 | Objeto_String.val := "" | Objeto_Bool.val := false
Type Casting	Bool_a_Int.val := if Bool_val then 1 else 0 end | Int_a_Bool.val := if Int_val != 0 then true else false end
Assignment Expressions	Asignación.val := id <- Expr.val
Method Calls and Return Values	LlamadaMétodo.val := método(Argumentos.val)
Control Structures	Control.val := if Bool_expr.val then Bloque_1.val else Bloque_2.val end | while Bool_expr.val do Bloque.val end
Expressions	Expr.val := Expr.val " + " Term.val | Expr.val " - " Term.val | Expr.val " * " Term.val | Expr.val " / " Term.val | Expr.val " < " Term.val | Expr.val " > " Term.val | Expr.val " <= " Term.val | Expr.val " >= " Term.val | Expr.val " == " Term.val | Expr.val " != " Term.val | " ~ " Expr.val | " not " Expr.val
Special Classes	Clase_IO.val := métodos_entrada_salida

Especificaciones de reglas semánticas sin nomenclatura

Regla Semántica	Descripción
Program Structure	Define la estructura del programa compuesta por una serie de clases. Cada clase contribuye a la totalidad del programa.
Main Class	Establece la estructura del programa con una clase principal y otras clases. La clase principal suele contener el método principal que inicia el programa.
Main Method	Define la clase principal con sus atributos, métodos y una invocación al método principal que es el punto de entrada del programa.
Inheritance	Define la clase principal únicamente con atributos y métodos, sin referencia al método principal.
Basic Types	Describe una clase con un tipo específico (entero, cadena o booleano) y sus respectivos atributos y métodos.
Scope	Establece el alcance de varios identificadores dentro de un contexto particular.
Identifier Definition	Define un atributo con un tipo específico y un identificador. Puede ser un entero, una cadena o un valor booleano.
Method Overriding	Define una clase B con sus atributos y métodos, e incluye un método A, posiblemente indicando la sobreescritura de un método heredado.
Inheritance Limitations	Define una clase general con sus atributos y métodos.
Default Values	Establece los valores por defecto para los objetos de tipo entero, cadena y booleano.
Type Casting	Define las reglas para convertir entre booleanos y enteros, utilizando una lógica condicional.
Assignment Expressions	Describe la asignación de una expresión a un identificador, estableciendo su valor.
Method Calls and Return Values	Define cómo se llama a un método con argumentos dentro del programa.
Control Structures	Describe las estructuras de control, como las condiciones if-else y los bucles while, utilizadas para guiar la ejecución del programa.
Expressions	Define una variedad de expresiones matemáticas y lógicas que pueden ser utilizadas en el programa, incluyendo operaciones aritméticas y comparaciones.
Special Classes	Define una clase especial para manejar métodos de entrada y salida, como lectura y escritura en archivos o consola.

Universidad del Valle de Guatemala
Compiladores
Sección 40

Laboratorio 1
Andrés de la Roca 20332
Jun Woo Lee 20

Reglas de tipos

- **Enteros (int)**
  - Aritmética:
    - Los enteros se pueden sumar, restar, multiplicar y dividir según las operaciones aritméticas estándar.
  - Comparación:
    - Los enteros se pueden comparar usando operaciones como menor, mayor, menor o igual, mayor o igual, igual o diferente de formas como: < , > , <= , >= , = , !=
  - Conversión a Booleano:
    - Un entero se puede convertir a un booleano en caso de que si es 0 se convierte en “false” y si es cualquier número no 0 se convierte en “true”

- Cadenas (string)
  - Concatenación:
    - Se pueden concatenar cadenas usando el operador +, como ejemplo: “cadena1” + “cadena2” termina como cadena1cadena2
  - Valor Predeterminado:
    - En caso de que no tenga un valor, se le asigna un valor predeterminado de cadena vacía de “”.

- Booleanos (bool)
  - Comparación:
    - Los booleanos pueden ser comparados por igualdad o desigualdad como “while bool” o “while not bool”
  - Valor Predeterminado:
    - En caso de que no tenga un valor, se le asigna un valor predeterminado de true
  - Conversión a entero:
    - Un booleano se puede convertir a un entero de forma de que si es true, se convierte a un 1, y si es false, se convierte a 0.

Universidad del Valle de Guatemala
Compiladores
Sección 40

Laboratorio 1
Andrés de la Roca 20332
Jun Woo Lee 20

Implementación inicial tabla de simbolos

```python
from prettytable import PrettyTable

class SymbolTable():
    def __init__(self) -> None:
        self.pretty_table = PrettyTable()
        self._symbols = []
        print('Iniciando nuevo ambito/scope')

    def add(self, type, id, size, isParameter):
        self._symbols.append({
            'Type': type,
            'ID': id,
            'Size': size,
            'IsParameter': isParameter
        })

    def lookup(self, id):
        symbols_copy = self._symbols.copy()
        symbols_copy.reverse()
        for symbol in symbols_copy:
            if symbol['ID'] == id:
                return symbol
        return 0

    def getsize(self):
        return sum(symbol['size'] for symbol in self._symbols)

    def totable(self):
        self.pretty_table.field_names = ['Type', 'ID', 'Size','IsParameter']
        for i in self._symbols:
            self.pretty_table.add_row(list(i.values()))

        print(" -- Simbolos -- ")
        print(self.pretty_table)
        self.pretty_table.clear_rows()
```