**Universidad del Valle de Guatemala**

**Facultad de Ingeniería**

**Departamento de Ciencias de la Computación**

**CC3032 - Construcción de Compiladores**

Aspectos Semánticos YAPL

Se resumen algunas consideraciones semánticas a tratar dentro de la fase de Análisis Semántico dentro del Compilador de YAPL:

* Un programa en YAPL es una secuencia de 1 o más definiciones de clases, con atributos y métodos.
* Todo programa en YAPL debe de contener una clase Main.
* La clase Main debe contener un método main sin parámetros formales.
* La clase Main no puede heredar de ninguna otra clase.
* Le ejecución de un programa en YAPL, comienza evaluando (new Main).main()

Tipos de Datos.

* Los tipos básicos del lenguaje son **Int, String** y **Bool**
* Los tipos básicos se pueden utilizar en la definición de clases, creando nuevos tipos derivados a partir de ellos.
* Las clases que definen los tipos básicos no pueden ser padres de alguna otra clase.

Ámbito

* Un atributo dentro de una clase, debe ser declarado antes de su uso.
* Un método dentro de una clase puede ser llamado de forma recursiva.
* Existen dos ámbitos dentro de una clase, el ámbito **Global** y el ámbito **Local**. Un ámbito Global es el definido dentro de la sección del cuerpo de la clase, mientras que un ámbito Local se crea con las instrucciones Let ó dentro de la definición de un método, dentro de la clase.
* Todos los atributos y métodos dentro de una clase poseen acceso *público.*
* Los identificadores de un ámbito local ocultan la definición de identificadores en el ámbito global.
* Ningún identificador puede ser definido más de una vez dentro de un mismo ámbito.
* Si B hereda de A y B sobreescribe un método de A, este método debe de poseer la misma firma con la que fue declarado en A.
* No es posible la herencia múltiple de clases y herencia recursiva.

Valores *default*

* Los objetos creados a partir de la clase Int, poseen valor default **0**.
* Los objetos creados a partir de la clase String, poseen valor default **“”** {cadena vacía}.
* Los objetos creados a partir de la clase Bool, poseen valor default **false.**

Casteo

* Es posible el casteo implícito de Bool a Int. (False es 0, True es 1)
* Es posible el casteo implícito de Int a Bool (0 es False, cualquier valor positivo es True)
* No es posible el casteo explícito.

Expresiones de Asignación

* Una asignación tiene la forma <id> <- <expr>
* El tipo estático <expr> debe coincidir con el tipo declarado para el <id>, o ser de un tipo heredado a partir del tipo de <id>
* El valor de <expr> del lado derecho se convierte en el valor del objeto <id>
* El tipo de dato de la asignación es el tipo de dato de <expr>
* Si el lado izquierdo de la asignación hace referencia a algún atributo de una clase, este atributo debe de encontrarse definido dentro de la clase.
* Tanto el lado izquierdo como el lado derecho de la asignación permiten identificadores recursivos {i.e. [class1].[class2]....[classn].[atributo]}

Llamadas a métodos y valores de retorno

* Los argumentos de los métodos que son de tipos básicos, se pasan por valor.
* Los argumentos de los métodos que son tipos derivados, se pasan por referencia.
* Los argumentos formales del método son considerados variables locales del método.
* Los argumentos son evaluados de izquierda a derecha.
* El tipo de la expresión de retorno del método debe coincidir con el tipo de retorno declarado con el método.
* El valor de la expresión de retorno será devuelto al método llamador y asignado al lado izquierdo de una expresión (si este fue llamado dentro de una expresión de asignación)

Estructuras de Control

* El tipo de dato estático de la <expr> utilizada en una estructura de control **if** o **while** debe ser de tipo **Bool**.
* El tipo de dato del condicional **if** es el tipo de dato del **bloque** que sea un supertipo de ambas ramas del condicional.
* El tipo de dato de la estructura **while** es **Object**.

Expresiones

* Los operadores aritméticos se aplican a Objetos creados a partir de la clase **Int.** El tipo de dato del resultado es del tipo **Int.**
* Los operadores de comparación se aplican a Objetos que del mismo tipo de datos estático o que sean Objetos de clases heredadas de la misma clase. El tipo de dato del resultado es del tipo **Bool.**
* La operación unario ~ aplicado al tipo **Int** devuelve como resultado un valor del tipo **Int.**
* La operaión unaria **not** aplicado a una expresión de tipo de dato **Bool**, devuelve una expresión de tipo de dato **Bool**.

Clases especiales

* Existe una clase especial IO que define funciones de entrada y salida de valores tipo **Int** y **Bool.**
* La tabla de símbolos debe de contener de forma predefinida la definición de las clases **IO, Int, String** y **Bool** junto a sus métodos ya definidos.