## *Parte 3*

**Realizar un analizador semántico que revise los siguientes puntos:**

1. Variables no utilizadas.

Ej:
int x;
int y;
y = 10;
System.out.println(y);
la variable x nunca es utilizada después de su declaración.

2. Si existen variables no utlizadas, entonces remover esta expresión del árbol. (optimización de la R.I.)

3. Variables duplicadas.

Ej:
int x;
String x;

4. Asignación de variables del tipo incorrecto (type checking). Incorporamos subtipos (herencia) y scope checking.

Ej:
Factorial fac;
int x;
fac = 10;
x = “uno”

5. Paso de parámetros de tipo incorrecto (type checking), de tamaño incorrecto o tipo de retorno incorrecto

Ej:
fac.getResult(fac);

*donde el método getResult tiene la siguiente firma: int getResult(int res)*

6. Type checking entre expresiones. (verificar variables y tipo de dato retornado por un método)

Ej:
String x;
int y;
y = 10 + 3 * x;

7. Chequeo de métodos existentes

Ej:
MiObjeto a;
a.miMetodo()
donde el método miMetodo() no fue declarado para la clase MiObjeto.

8. Type checking en la expresión de retorno de un método

Ej:
```
public int miMetodo(){
        String b;
```

```
        return b:
}
```
Donde la variable b no es del tipo especificado como retorno del metodo (se especificó int)

Para los puntos del 3 al 8 el analizador no debe abortar el proceso de compilación, sinó que debe buscar la mayor cantidad de errores posibles. Una vez recorrido todo el arbol, imprime un sumario de los errores encontrados.

*Tip*: haga un visitor que mantenga una tabla de símbolos y los errores encontrados; al final de la visita, imprime los errores.

**Forma de presentación.**

Los archivos en el classroom y un informe explicando lo que logró y lo que no.

## *Parte 4*

**Realizar un generador de código jasmin:**

Crear un visitor que genere código válido de jasmin. El generador de código deberá crear un archivo .j del mismo nombre de la clase Main.

**Forma de presentación.**

En el exámen final, deberán presentar el compilador con todas sus partes unidas y funcionales a partir de un main, que realice las tareas de Análisis Léxico, Sintáctico, Semántico y Generación de Código.

Tip: A partir de TestParser.java pueden modificarlo para que realice cada proceso de compilación secuencialmente.