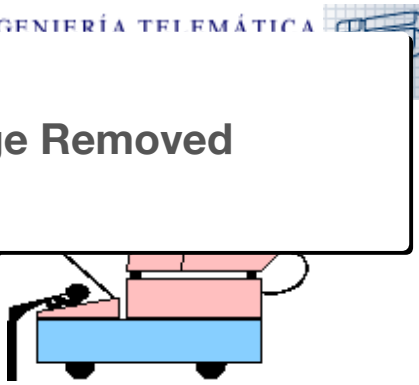

Práctica 3: Generación de código

Comportamiento esperado de la ejecución de un programa fuente

La ejecución de un programa fuente consiste en la ejecución de la secuencia de instrucciones que lo forman. El comportamiento de las instrucciones de asignación, condicional (`if...`) y bucle (`while...`) se explicó en la práctica 1. El comportamiento de la sentencia `print` es equivalente al del método `System.out.println` de Java. El comportamiento de las expresiones se explicó en la práctica 1, salvo en el caso de los operadores lógicos. El comportamiento de los operadores lógicos viene descrito por las siguientes tablas:

a	b	a and b	a or b
true	true	true	true
true	unk	unk	true
true	false	false	true
unk	true	unk	true
unk	unk	unk	unk
unk	false	false	unk
false	true	false	true
false	unk	false	unk
false	false	false	false

a	not a
true	false
unk	unk
false	true

Se supone que las variables al declararse (incluyendo, si los hubiera, los argumentos del programa) se inicializan por defecto a los siguientes valores:

- Variables tipo `int`: 0.
- Variables tipo `string`: `""`.
- Variables tipo `bool`: `false`.

Ejecución del código generado

Al traducir un programa fuente, el código Java generado por el compilador deberá cumplir lo siguiente:

- El resultado de traducir un programa fuente consistirá en una o varias clases Java, que se dejarán en el directorio desde el que se ejecuta el compilador.
- De entre las clases Java generadas, **obligatoriamente** existirá una que se utilizará para ejecutar el código Java generado, cuyo nombre se le pasará al compilador en un argumento al invocarlo, según se explica más adelante, **y que no pertenecerá a ningún paquete**. Por lo tanto, dicha clase Java deberá contener un método `main`.

- El programa obtenido con el compilador se ejecutará de la siguiente forma:

  ```
  java <nombre_clase_generada> [<argumento1> <argumento2> ...]
  ```

  Los argumentos son opcionales. Los argumentos se utilizan para pasárselos al programa a ejecutar en las variables que se definen en la cabecera del programa. Los argumentos se guardan en las variables según el orden en que éstas se declaran. Por ejemplo, supongamos que tenemos un programa en lenguaje PF2016, cuya cabecera es:

  ```
  PF2016 Ejemplo(arg1, arg2, arg3)
  ```

  Supongamos que al traducir este programa a Java la clase que se utiliza para ejecutar el código generado se llama `EjemMain`. Supongamos que ejecuto dicha clase Java, pasándole 2 argumentos, de la siguiente forma:

  ```
  java EjemMain Hola Mundo
  ```

  El efecto de esta ejecución es que las variables en la cabecera del programa se inicializan de la siguiente forma: `arg1="Hola"`, `arg2="Mundo"` y `arg3=""`. Nótese que por lo tanto no es un error que se pasen menos argumentos que variables se declaran en la cabecera, sino que las variables a las que no se les asigna un argumento se inicializan a `""`. Tampoco es un error pasar más argumentos que variables hay en la cabecera; los argumentos que sobren simplemente se ignoran.

Requisitos que deben cumplir los ficheros JLex y CUP y las clases Java a entregar

Para esta práctica se deberá desarrollar una clase `Main` (puede utilizar como punto de partida la proporcionada en la práctica 1 y/o la que haya desarrollado en la práctica 2). La ejecución de dicha clase se realizará de acuerdo con lo siguiente:

```
java Main <nombre_fichero> <nombre_clase_generada>
```

Donde `<nombre_fichero>` es el nombre del fichero (con el path si es necesario) del programa fuente a traducir y `<nombre_clase_generada>` es el nombre de la clase Java generada al ejecutar el compilador que contiene su propio método `main`.

El programa deberá de levantar una excepción que no deberá ser capturada en el caso de que el programa fuente tenga algún error léxico, sintáctico o semántico. Si el programa fuente no contiene errores, el compilador debe depositar en el directorio actual uno o varios ficheros que contendrán la o las clases Java generadas para el programa fuente.

Obligatoriamente las clases generadas por CUP deberán pertenecer a un paquete llamado Parser y la clase generada por JLex deberá pertenecer a un paquete llamado Lexer.

Las clases Java que desarrolle en esta práctica **obligatoriamente** deberán estar organizadas en los siguientes paquetes:

- Paquete `Errors`, que debe contener todas las excepciones utilizadas (incluidas las clases proporcionadas en la práctica 1). Si las excepciones utilizasen alguna clase auxiliar, dicha clase deberá pertenecer asimismo a ese paquete.
- Paquete `AST`, contendrá todas las clases necesarias para representar árboles de sintaxis abstracta correspondientes a programas del lenguaje de programación fuente.
- Paquete `Compiler`, contendrá cualquier otra clase que necesite el compilador, incluyendo las de la tabla de símbolos.
- Opcionalmente, paquete `GeneratedCodeLib`, que contendrá las clases Java que usted desarrolle para que sean utilizadas por el código generado (si es que éstas existen).

La clase `Main` no pertenecerá a ningún paquete.

Orden de compilación

Antes de entregar la práctica deberá cerciorarse de que su práctica puede compilarse correctamente con `javac` siguiendo el siguiente orden:

1. Clases del paquete `Errors`.
2. Clases del paquete `GeneratedCodeLib` (si es que éstas existen).
3. Clases del paquete `Compiler`.
4. Clases del paquete `AST`.
5. Clases `parser` y `sym` generadas por CUP.
6. Clase `Yylex` generada por JLex.
7. Clase `Main`.

Aquellas prácticas que no se puedan compilar en este orden serán calificadas con 0.

Ayudas y sugerencias

Se proporciona un juego de tests con el que probar la práctica. De entre ellos solamente deberían de producir un error los tests en carpetas cuyo nombre comienza por `ErrSem`, `ErrSint` y `ErrLex`. Puede ocurrir que alguno de los ejemplos `ErrLex` de error de sintaxis (y no léxico). Los tests de las carpetas cuyo nombre comienza por `ErrSem` deberían producir un error semántico.

Las carpetas que contienen programas fuente que no tienen errores (es decir, carpetas cuyo nombre empieza por `Ejem`) contienen además un fichero de nombre `ejecucion.txt` que contiene el resultado esperado de una o varias ejecuciones (con diferentes argumentos) de la o las clases Java generadas para el programa fuente contenido en la carpeta.

Se advierte que se realizarán tests adicionales a los proporcionados a las prácticas recibidas, por lo que se recomienda a los alumnos que planifiquen tests complementarios a su práctica.

Ficheros a entregar

Se deberán entregar exclusivamente los siguientes ficheros:

- fichero `Yylex` en formato JLex para el analizador léxico
- fichero `parser` en formato CUP para el analizador sintáctico
- fichero `java.zip`, que al descomprimirlo genere una carpeta de nombre `java` cuyo contenido debe ser **exactamente** el siguiente:
  - fichero `Main.java` que contenga la clase `Main`.
  - Carpeta `Errors` que contenga las clases Java del paquete `Errors` (proporcionar ficheros Java, no ficheros `.class`).
  - Carpeta `Compiler` que contenga las clases Java del paquete `Compiler` (proporcionar ficheros Java, no ficheros `.class`).
  - Carpeta `AST` que contenga las clases Java del paquete `AST` (proporcionar ficheros Java, no ficheros `.class`).
  - Opcionalmente, carpeta `GeneratedCodeLib` que contenga las clases Java del paquete `GeneratedCodeLib` (proporcionar ficheros Java, no ficheros `.class`).

Last Revision: 03/03/2016 17:17:00