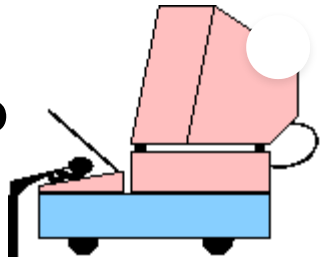
Práctica 3: Generación de código



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...then`) y bucle (`while...do`) es análogo al de sus equivalentes en lenguaje Java. El comportamiento de las sentencias `printInt` y `printBool` se explicó en la práctica 1. El comportamiento de las expresiones se explicó en la práctica 1.

Un programa en lenguaje fuente no espera recibir ningún dato de entrada para ejecutar. Por lo tanto, dado un programa fuente existe un único resultado que dicho programa puede producir. Se supone que las variables al declararse se inicializan por defecto a los siguientes valores:

- Variables tipo `int`: 0.
- Variables tipo `intset`: conjunto vacío.
- Variables tipo `bool`: `false`.

revisar los valores por defecto!

este año no recibe ningun argumento nuestros programas

Ejecución del código generado

El código generado por el compilador constará de una o varias clases Java. En todo caso, existirá **obligatoriamente** entre las clases generadas una cuyo nombre coincida con el del programa fuente (es decir, el identificador inmediatamente a continuación de la palabra clave `pf2024`), **y que no pertenezca a ningún paquete**. Esta clase debe tener un método `main` que se utilizará para ejecutar el código generado. Al ejecutar esta clase Java no hace falta pasarle ningún argumento.

Por ejemplo, si compilamos el programa `ejem1.prg` que se puede encontrar en los ejemplos proporcionados más adelante, obligatoriamente se tiene que producir una clase de nombre `Ejem1` que debe contener un método `main` que se utilizaría para ejecutar el código generado. El resultado de ejecutar `Ejem1` será equivalente a la ejecución de las instrucciones contenidas en `ejem1.prg`, según lo indicado más arriba y en la práctica 1.

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>
```

Donde `<nombre_fichero>` es el nombre del fichero (con el path si es necesario) del programa fuente a traducir.

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 `resul.txt` que contiene el resultado que debería

imprimirse por pantalla al ejecutarse 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.

Se proporciona además una clase abstracta IntSetA que implementa algunas de las operaciones con conjuntos de las que dispone el lenguaje PF2024. El uso de esta clase es opcional. Esta clase utiliza la excepción EmptySetException, que también se proporciona.

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`).