Paso 0. Creación de la estructura inicial

Resumen

En este primer capítulo se creará el proyecto en Eclipse con el código inicial del proyecto. El alumno no necesitará hacer ninguna tarea; simplemente copiará el código ya hecho y comprobará que funciona. De esta manera se estará preparado para poder ya empezar a trabajar a partir del siguiente capítulo. Los capítulos posteriores consistirán en modificar y/o añadir fuentes a este proyecto.

Esqueleto del Traductor

La carpeta "Esqueleto Traductor" incluye clases Java que facilitan el arranque de la implementación de un traductor. Los fuentes del esqueleto son de *dos tipos*:

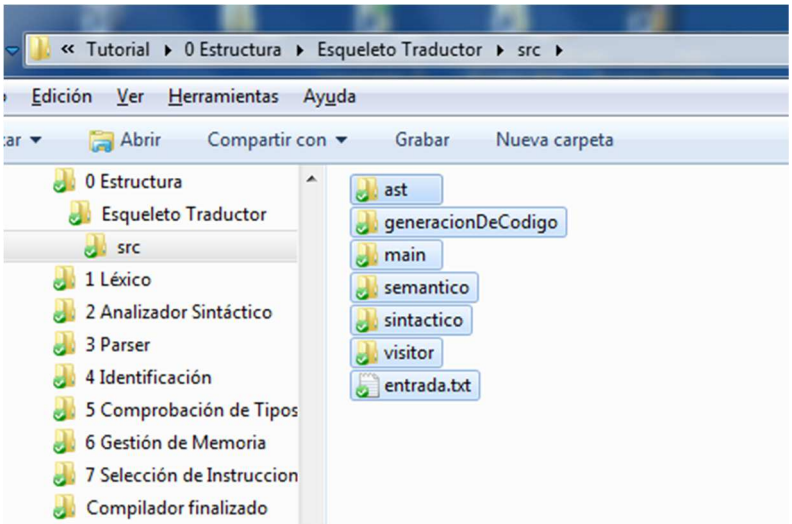
- Aquellos que incluyen código rutinario que es siempre igual en todos los compiladores. Estos no serán modificados.
- Aquellos que están vacíos y su única función es indicar dónde se deberían ubicar dichos ficheros cuando se creen en la fase apropiada. Se pretende ayudar así a mantener una estructura adecuada en el compilador.

Este esqueleto no ha sido hecho expresamente para este tutorial, sino que puede ser utilizado independientemente de él. De hecho, se recomienda también su uso como punto de partida para la práctica del alumno.

Creación del Proyecto

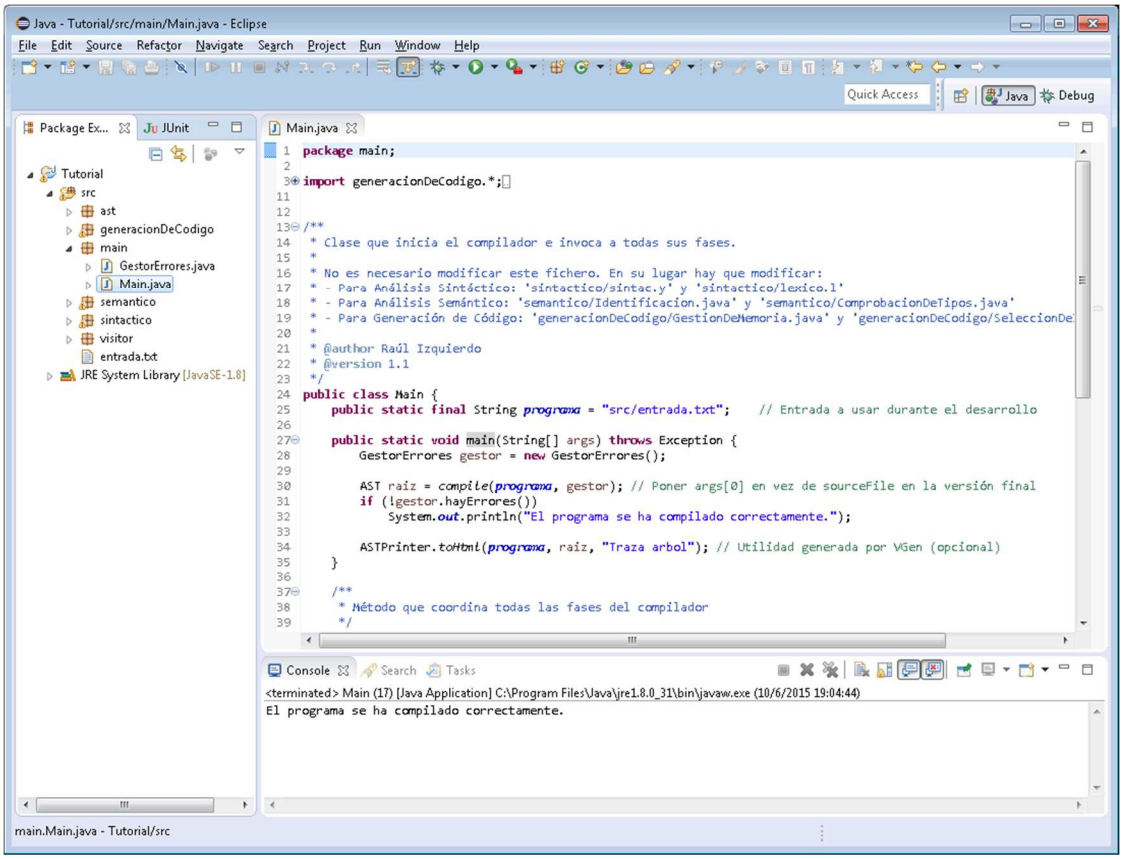
El primer paso es crear un proyecto llamado *Tutorial* en Eclipse. Con esto aparecerá dicho proyecto con una carpeta *src* vacía.

A continuación, seleccione todo el contenido de la carpeta *"Esqueleto Traductor/src"* y cópielo a Eclipse a la carpeta *src* del proyecto *Tutorial*[1].



[1] Esto puede hacerse simplemente arrastrando los fuentes seleccionados en la imagen a la carpeta *src* del proyecto en la ventana "*Package Explorer*" de Eclipse.

El proyecto quedará con la estructura que se muestra en la siguiente imagen. Para comprobar que todo funciona correctamente, ejecute la clase *main/Main*. Debería aparecer el mensaje "*El programa se ha compilado correctamente*".



Descripción del Código

Funcionalidad

El código copiado implementa un traductor que solo admite como entrada válida un fichero de texto que tengan únicamente el carácter *punto y coma*. En el fichero *"sintactico/sintac.y"* se encuentra la siguiente gramática:

```
program: ';'
```

En el fichero *entrada.txt* se tiene el único carácter válido:

```
;
```

La única razón de esta funcionalidad es tener un código inicial que compile sin errores y se pueda ejecutar, aunque no haga nada útil aún. Es más sencillo ir modificando un programa que ya compila sin errores que empezar con un proyecto completamente vacío.

Clase Main

La clase Main dirige todo el proceso de compilación.

```
public class Main {
   public static final String programa = "src/entrada.txt";// A usar durante el desarrollo

   public static void main(String[] args) throws Exception {
      GestorErrores gestor = new GestorErrores();

      AST raiz = compile(programa, gestor); // Poner args[0] en la versión final
      if (!gestor.hayErrores())
         System.out.println("El programa se ha compilado correctamente.");

     ASTPrinter.toHtml(programa, raiz, "Traza arbol"); // Generada por VGen (opcional)
   }

   /**
    * Método que coordina todas las fases del compilador
    */
   public static AST compile(String sourceName, GestorErrores gestor) throws Exception {

      // 1. Fases de Análisis Léxico y Sintáctico
      Yylex lexico = new Yylex(new FileReader(sourceName), gestor);
      Parser sintáctico = new Parser(lexico, gestor, false);
      sintáctico.parse();

      AST raiz = sintáctico.getAST();
      if (raiz == null) // Hay errores o el AST no se ha implementado aún
         return null;

      // 2. Fase de Análisis Semántico
      AnalisisSemantico semántico = new AnalisisSemantico(gestor);
      semántico.analiza(raiz);
      if (gestor.hayErrores())
         return raiz;

      // 3. Fase de Generación de Código
      File sourceFile = new File(sourceName);
      Writer out = new FileWriter(new File(sourceFile.getParent(), "salida.txt"));

      GeneracionDeCodigo generador = new GeneracionDeCodigo();
      generador.genera(sourceFile.getName(), raiz, out);
      out.close();

      return raiz;
   }
}
```

La clase *Main* no habrá que modificarla en todo el tutorial. Lo que harán los siguientes capítulos del tutorial es implementar algunas de las clases *que son invocadas desde aquí* y que actualmente están vacías. Dichas clases, aunque no hagan nada, se incluyen para indicar dónde irá cada funcionalidad en el futuro. Por ejemplo, a la hora de implementar la Fase de Identificación del analizador semántico, habrá que completar los métodos de la clase *"semantico/Identificacion.java"* (que ahora se encuentra vacía).

Se recomienda ahora mirar cada una de las clases del esqueleto (son bastante pequeñas y con comentarios que explican su función). En cualquier caso, se irá viendo su función en capítulos posteriores a medida que se vayan necesitando en cada fase.