Integración de JFlex y Cup en Eclipse

CONTENIDO

FIGURAS

CONSIDERACIONES PREVIAS

Esta guía se ha desarrollado bajo las siguientes condiciones de trabajo:

1. workspace = C:\workspace
2. directorio de JFlex = C:\JFlex
3. directorio de Cup = C:\Cup
4. java.exe utilizado = C:\j2sdk1.4.2_05\bin

Si estas condiciones varían, deberán adaptarse las rutas (tal y como se recordará más adelante) al poner en práctica los apartados de esta guía relativos a la configuración de las herramientas.

Se ha colocado, en la web de la asignatura, un workspace de eclipse con las herramientas ya configuradas. Este workspace es específico para las aulas de laboratorio de LFA para el presente curso. El alumno podrá, en las clases de prácticas, copiar y descomprimir el workspace en el disco duro. Se generará el subdirectorio “LFA\workspace” (borrarlo si existe previamente) que deberá seleccionarse como workspace al iniciar Eclipse (podemos seleccionarlo también una vez iniciado Eclipse con la opción ‘Switch workspace’).

Asimismo, al finalizar la práctica, el alumno puede guardarse dicha carpeta y seleccionarla como workspace en la próxima sesión para mantener el trabajo realizado.

Integración de JFlex

Vamos a crear un proyecto Java donde se hace uso de la herramienta JFlex. Se trata de un ejemplo muy sencillo, donde incluiremos un analizador léxico representado por una clase llamada 'Lexer'.

1. Creación de un proyecto nuevo

En primer lugar arrancamos el entorno eclipse y nos aseguramos de que esté seleccionada la perspectiva Java (menú Window→Open Perspective→Java).

A continuación seleccionamos la opción de crear un nuevo proyecto (menú File→New→Project…), con lo que nos aparece la siguiente ventana:

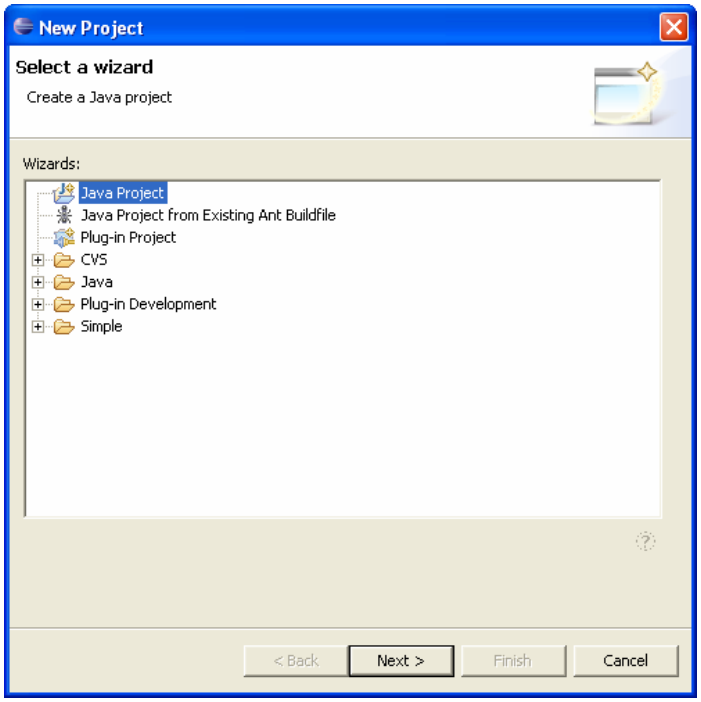

figura 1. New Project

Tras seleccionar 'Java Project' pulsamos el botón 'Next' y nos aparece una ventana (figura 2), donde escribiremos el nombre del proyecto que vamos a crear (en este caso lo llamaremos 'proyecto_jflex') y opcionalmente, el directorio donde queremos crearlo (por defecto se crea en el workspace especificado al iniciar eclipse). Finalmente podemos pulsar 'Finish', lo que nos llevará al entorno principal.

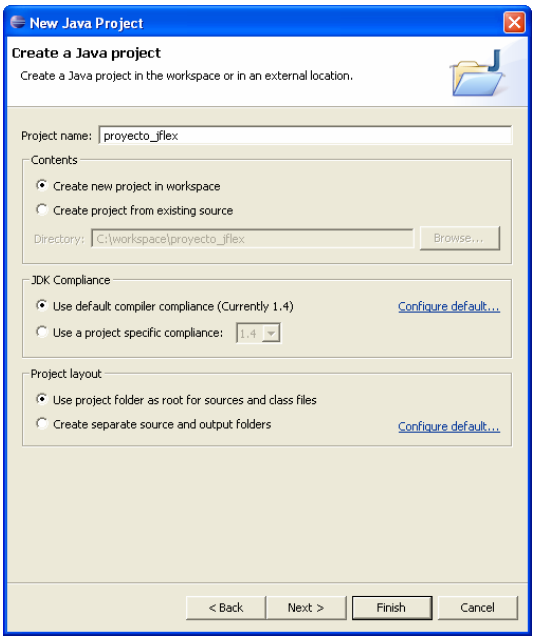

figura 2. Datos del proyecto nuevo

2. Creación de 'Lexer.flex'

A continuación creamos el fichero 'Lexer.flex' (nuestro analizador léxico en JFlex). Para ello pulsamos el botón derecho del ratón sobre el nombre del proyecto y seleccionamos la opción New→File (figura 3).

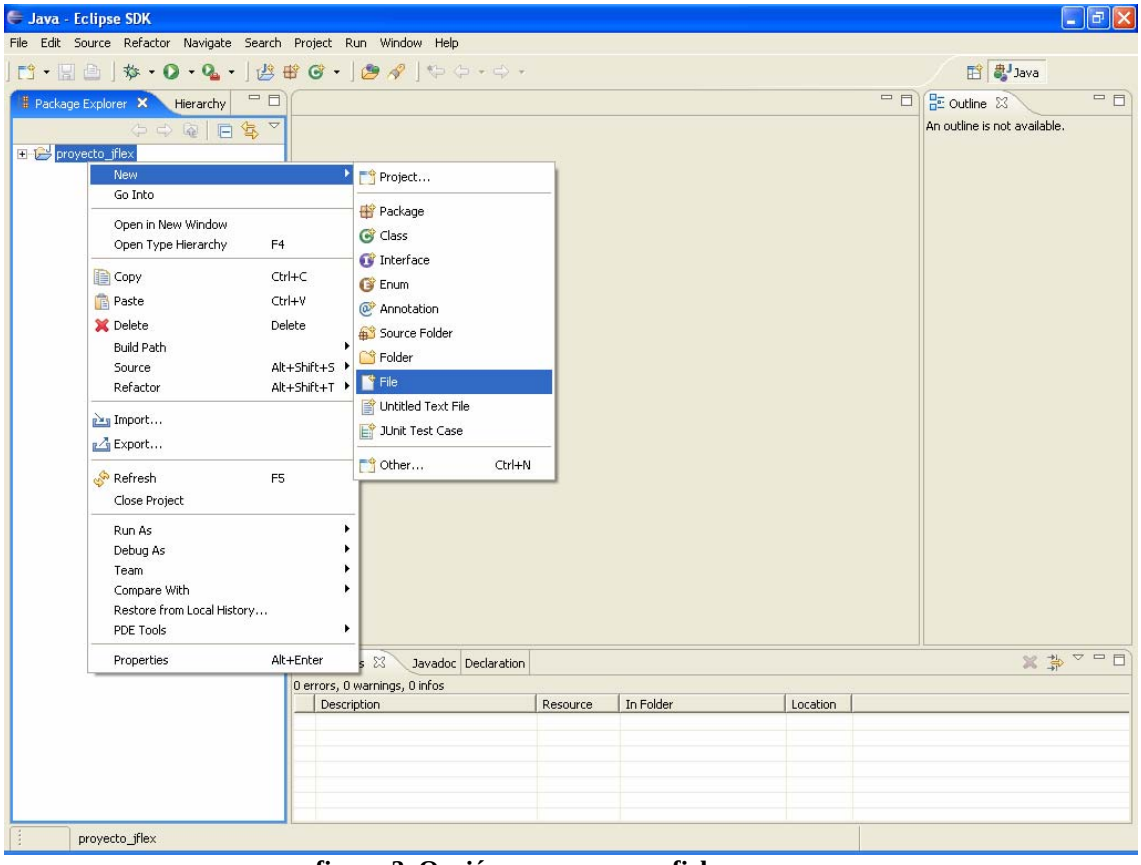

figura 3. Opción para crear un fichero nuevo

Se nos abre una ventana (figura 4) donde introducimos el nombre del fichero nuevo que queremos crear y pulsamos el botón 'Finish'.

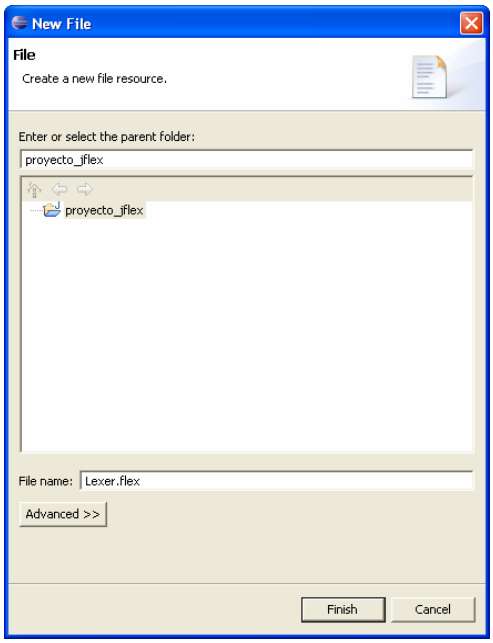

figura 4. Especificando el nombre del fichero nuevo

Aparece ya en nuestro entorno el fichero 'Lexer.flex' vacío (figura 5) para ser editado (más tarde veremos como procesarlo con JFlex).

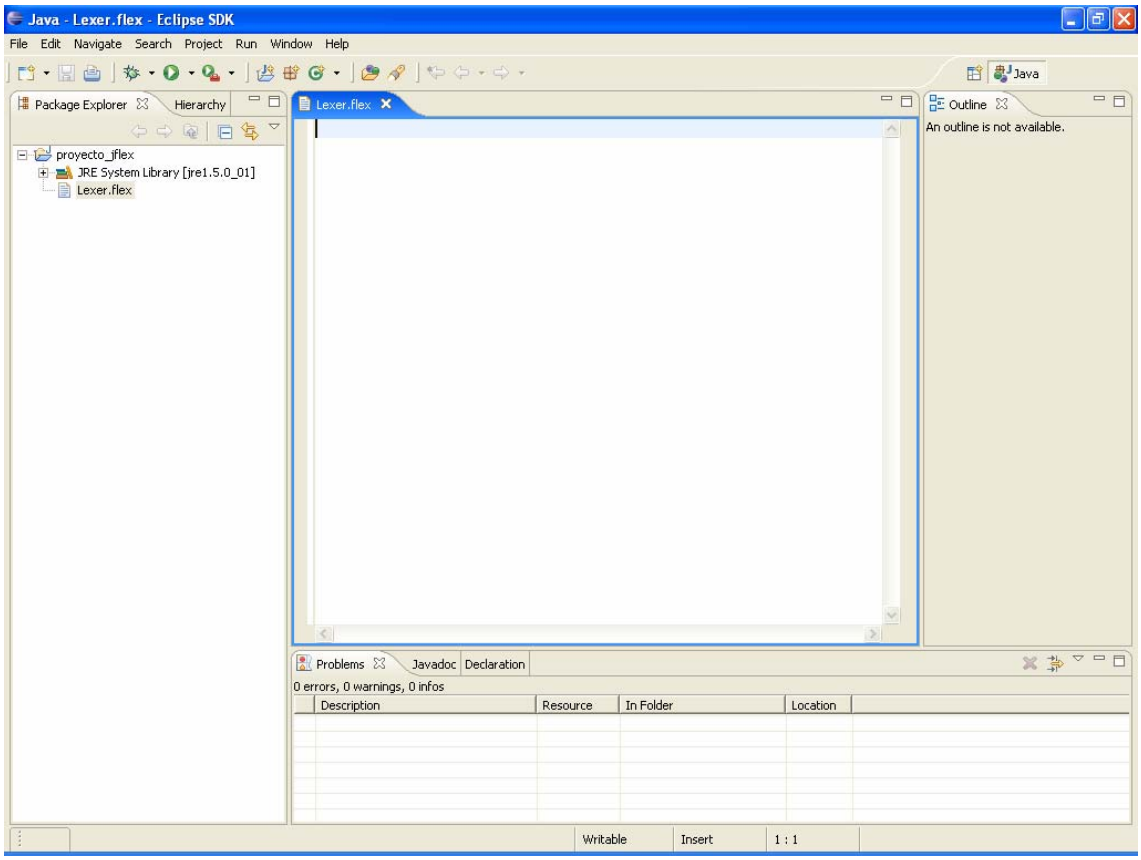

figura 5. Fichero 'Lexer.flex' recién creado

3. Incorporación del fichero de entrada al proyecto

Para incorporar a nuestro proyecto el fichero de entrada que vamos a pasarle a nuestro analizador léxico, de manera que nos sea más fácil editarlo, podemos proceder de la misma manera que con el fichero 'Lexer.flex'. En la ventana de la figura 4 introduciremos 'entrada.txt' en lugar de 'Lexer.flex' y listo.

4. Compilación con JFlex

Ya podemos pasar a configurar cómo compilar los ficheros para JFlex (Lexer.flex) en eclipse. Utilizaremos la opción para especificar herramientas externas al entorno (en nuestro caso, la herramienta JFlex) y para ello seleccionamos el menú Run→External Tools→External Tools… (figura 6).

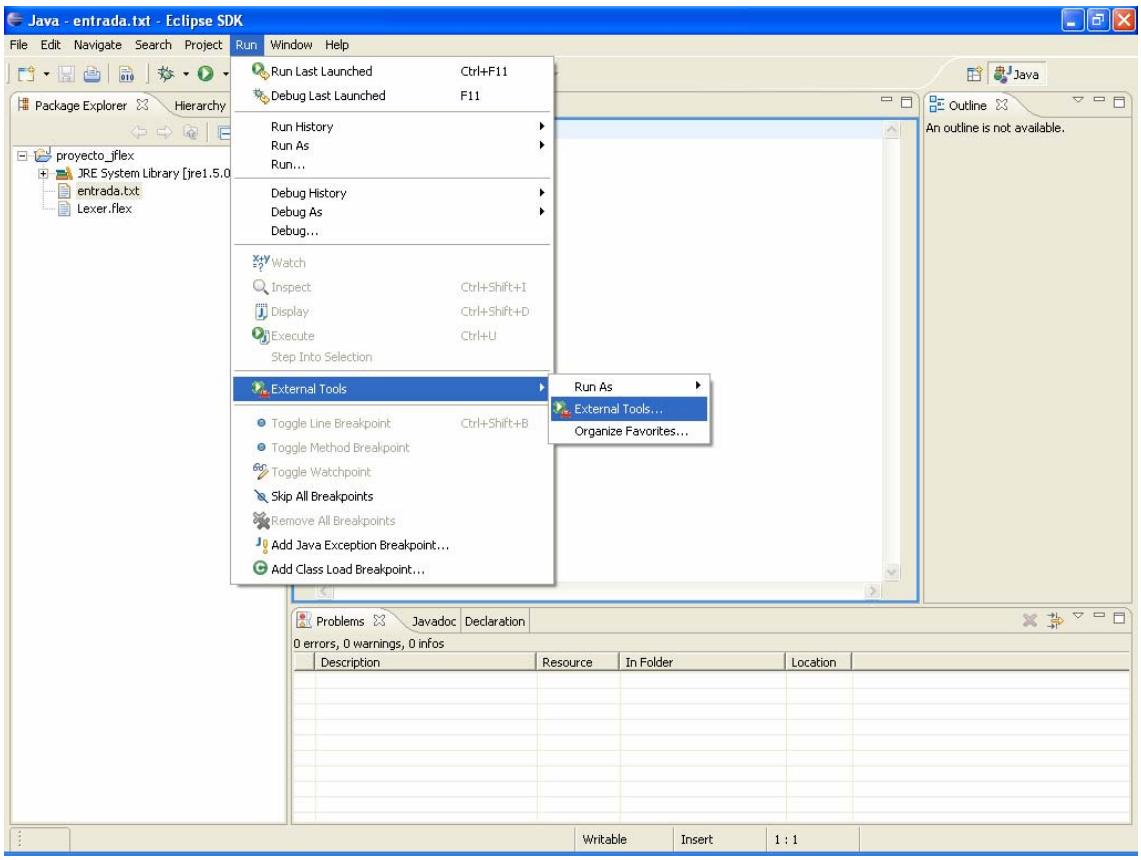

figura 6. Menú de herramientas externas

A continuación seleccionaremos la opción 'Program' para luego pulsar el botón 'New'.

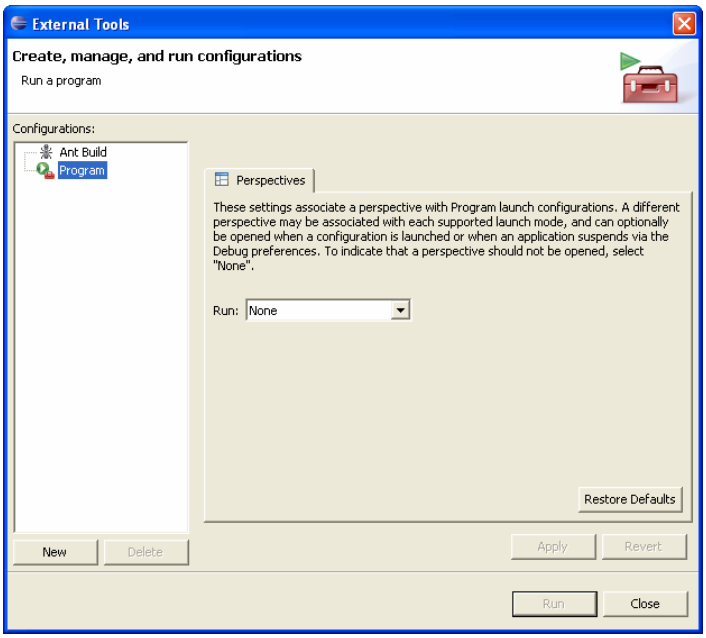

figura 7. Ventana inicial de herramientas externas

Ahora especificaremos la herramienta externa que queremos ejecutar para compilar con JFlex. Para ello introducimos los siguientes datos (tomar como ejemplo la figura 8 adaptando las rutas a los ficheros):

- ✓ en la casilla 'Name': el nombre que queremos darle a esta configuración para ejecutar JFlex (p. ej. 'JFlex').
- ✓ en la casilla 'Location': la ruta hasta el archivo 'java.exe', que será el encargado de ejecutar JFlex.
- ✓ en la casilla 'Working Directory': la variable ${project_loc} que hace referencia al directorio donde está nuestro proyecto.
- ✓ en la casilla 'Arguments': los parámetros de JFlex (ver figura 8)

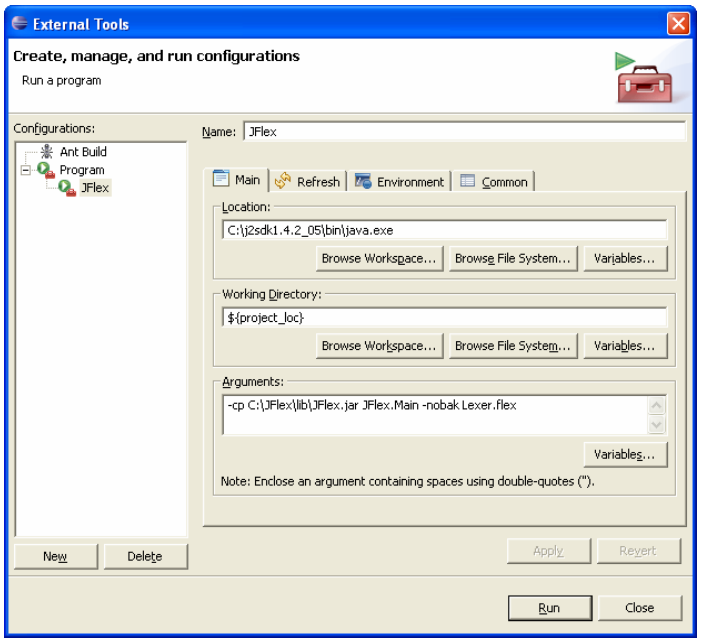

figura 8. Configuración para ejecutar JFlex

En la pestaña ‘Refresh’ de la misma ventana (figura 9) debemos indicarle a eclipse que actualice el proyecto una vez ejecutada esta herramienta (para que se tengan en cuenta los cambios del fichero Lexer.java creado o modificado por JFlex).

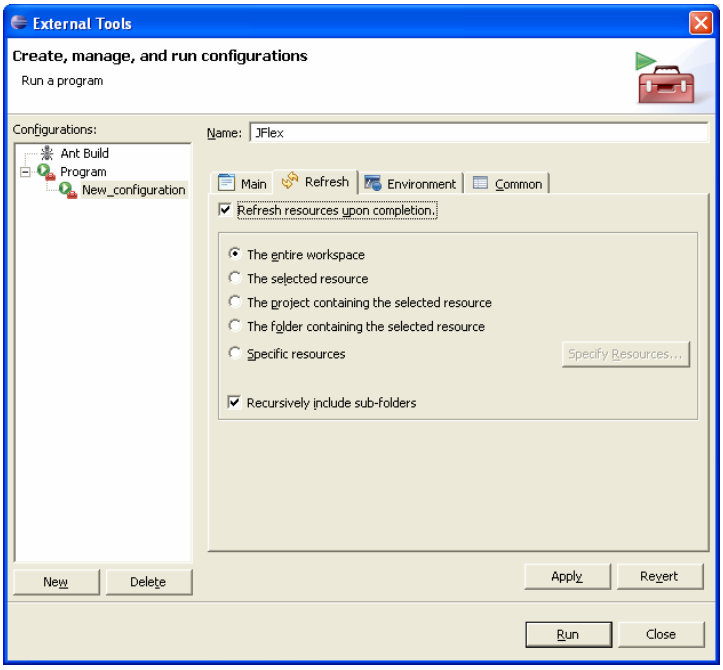

figura 9. Opciones para actualizar el proyecto

Para terminar pulsamos el botón ‘Apply’ y luego el botón ‘Run’ para ejecutarlo por primera vez. A partir de este momento, para volver a compilar con JFlex, podremos recurrir al botón de acceso rápido que tenemos en el entorno principal (pulsando en la flecha del botón se despliegan las *external tools* que tengamos configuradas).

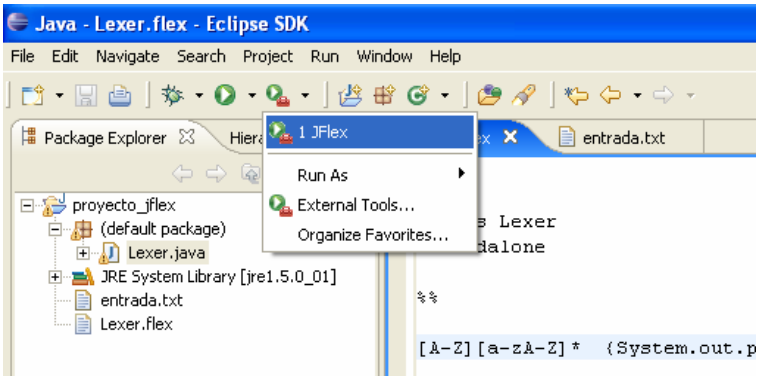

figura 10. Acceso rápido a las herramientas externas

En el caso de ejecutarlo sin haber introducido la especificación del analizador en el fichero ‘Lexer.flex’, es decir, con el fichero vacío, la consola mostrará el mensaje de error que aparece en la parte inferior de la figura 11.

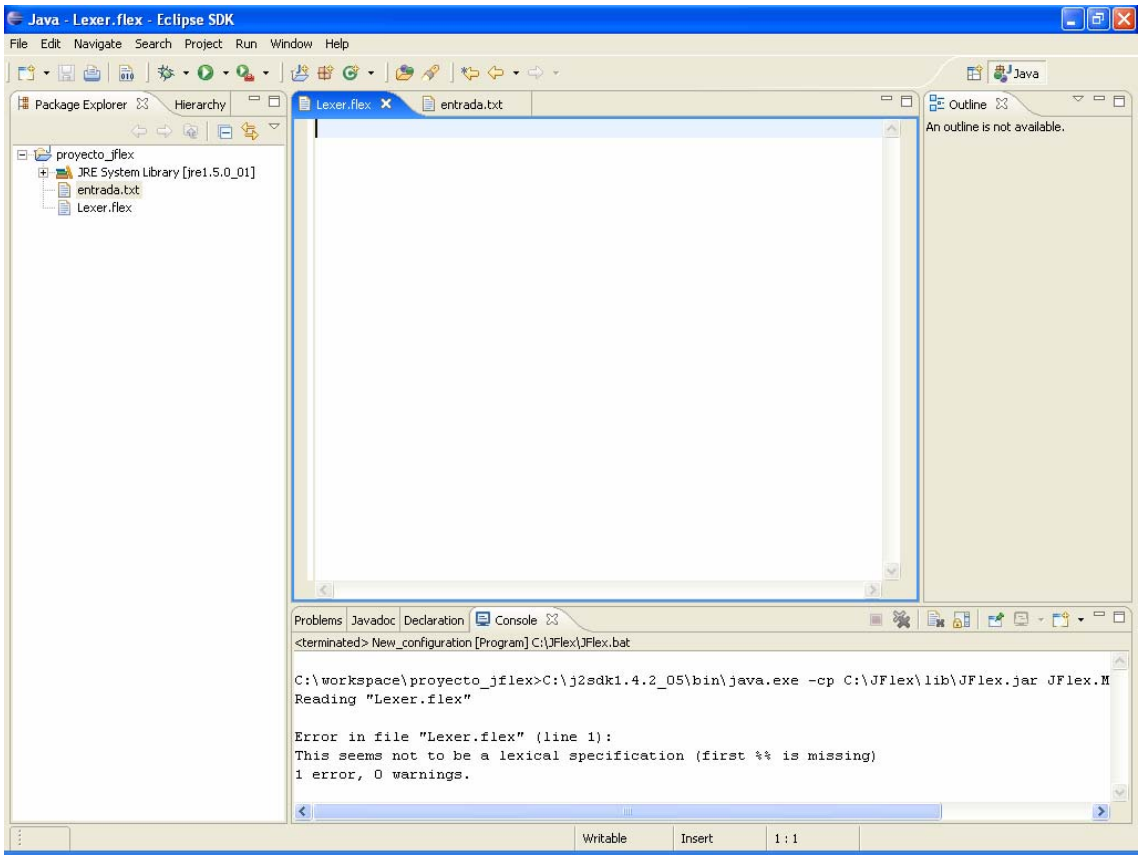

figura 11. Error al estar 'Lexer.flex' vacío

Opcionalmente, podemos sustituir 'Lexer.flex' en los argumentos de la herramienta JFlex (ver figura 8) por la variable ${resource_name}. Esto nos permite ejecutar JFlex sobre el fichero que queramos sin necesidad de que se llame 'Lexer.flex'. Para hacerlo seleccionaríamos el nombre del fichero que contiene nuestro analizador léxico[1] en el 'Package Explorer' (o la pestaña del propio fichero si lo tenemos abierto) y ejecutaríamos la opción de menú Run→External Tools→JFlex. De todas formas, hay que tener en cuenta la incompatibilidad de esta configuración con el Apéndice 2 y la necesidad de que esté seleccionado el fichero adecuado cada vez que ejecutemos una herramienta externa.

5. Ejecución del proyecto

Una vez configurada la herramienta JFlex y habiendo comprobado que se ha generado el fichero 'Lexer.java'[2] (vemos que se agrega al 'default package' en el panel que aparece a la izquierda de la pantalla), accedemos al menú Run→Run… para indicar la forma de ejecutar nuestro proyecto (figura 12).

[1] Si ejecutamos JFlex teniendo otro fichero seleccionado que no sea el '.flex', nos saldrá un error de especificación no válida, ya que se está intentando compilar con JFlex dicho fichero incompatible.
[2] Suponemos que se ha incluido la opción '%standalone' en el fichero 'Lexer.flex' para que la clase 'Lexer' generada tenga el método 'main' (si no fuese así habría que implementarla en otra clase aparte).

Nos aparece la ventana principal donde crear y gestionar la forma de ejecutar nuestro proyecto (figura 13), en la que seleccionaremos la opción 'Java Application' pulsando el botón 'New' para crear una nueva configuración (podremos crear tantas configuraciones como queramos para ejecutar el proyecto de varias formas diferentes).

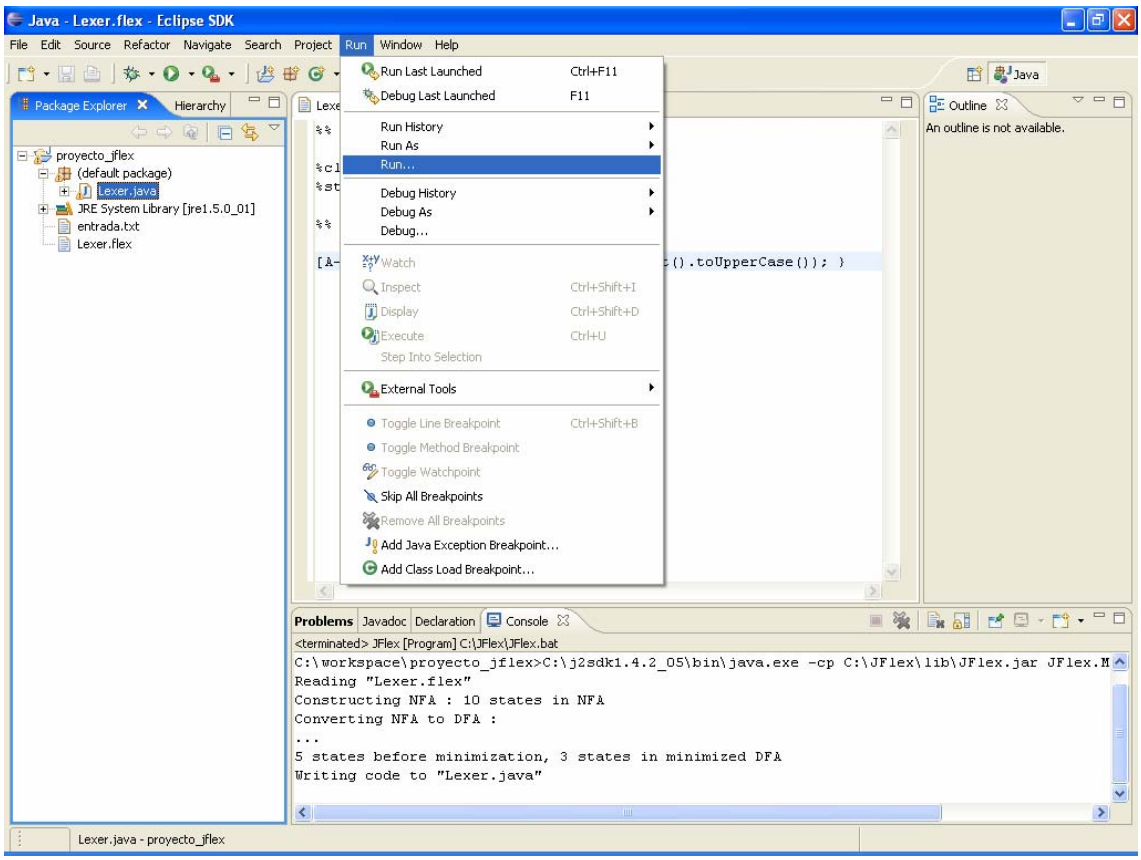

figura 12. Acceso a las opciones de ejecución del proyecto

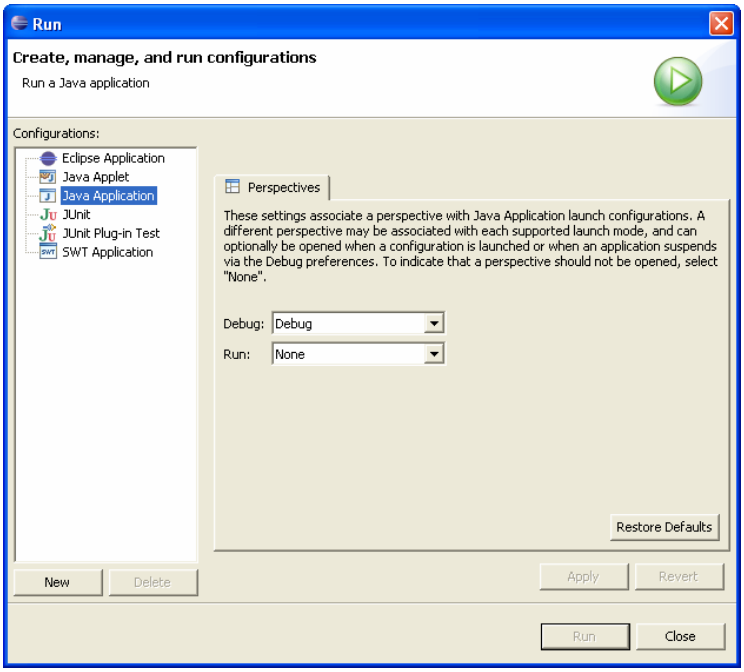

figura 13. Ventana inicial de opciones de ejecución

En esta nueva configuración que estamos creando, indicaremos en primer lugar el nombre de la configuración, el proyecto sobre el que se aplicará y la clase que debe utilizarse como clase principal.

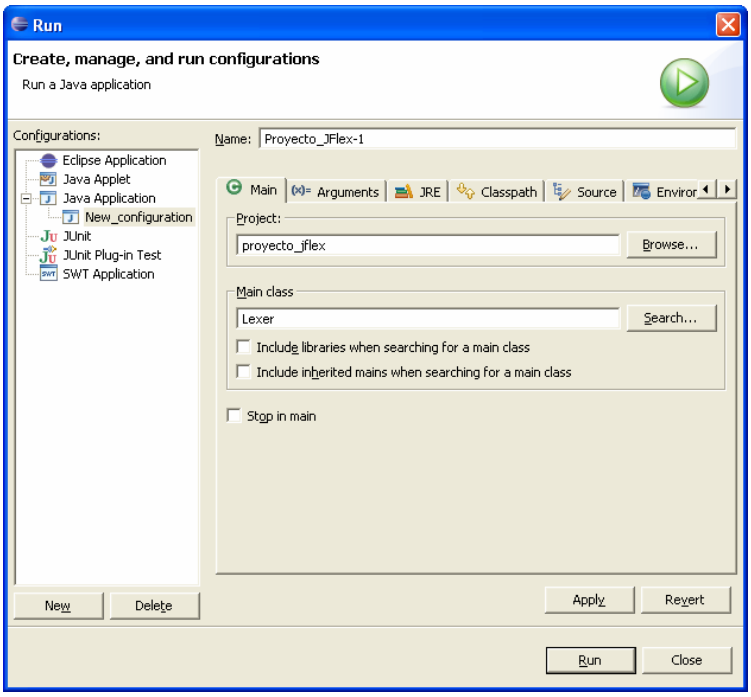

figura 14. Creando una configuración de ejecución

Además de esto accederemos a la pestaña 'Arguments' para especificar la lista de argumentos que se le pasarán al programa (en nuestro ejemplo será el fichero sobre el que ejecutar nuestro analizador léxico).

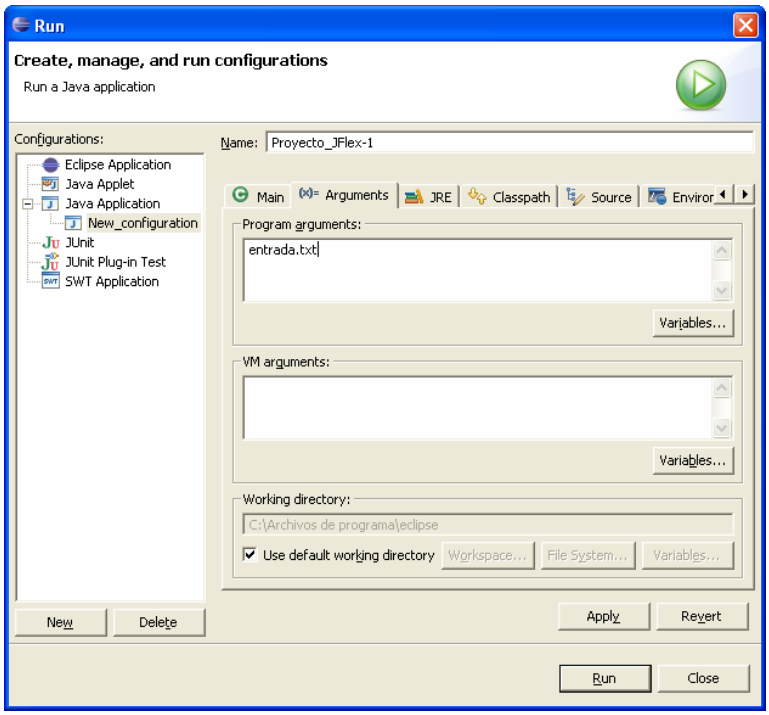

figura 15. Especificando los argumentos del programa

Para terminar pulsamos el botón 'Apply' y luego el botón 'Run' para ejecutarlo por primera vez. A partir de este momento, para volver a ejecutar el proyecto según esta configuración, podremos recurrir al botón de acceso rápido que tenemos en el entorno principal (figura 12).

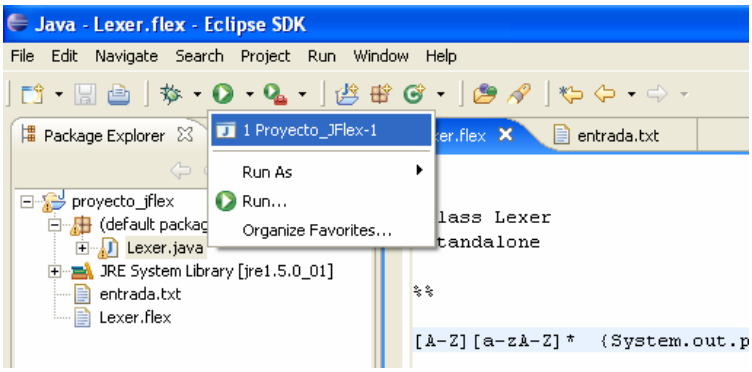

figura 16. Acceso rápido a las configuraciones de ejecución

6. Desactivación de la compilación automática

Por último y de forma opcional, podemos desactivar la opción de compilar de forma automática, para mantener el control sobre el proceso de compilación y ejecución.

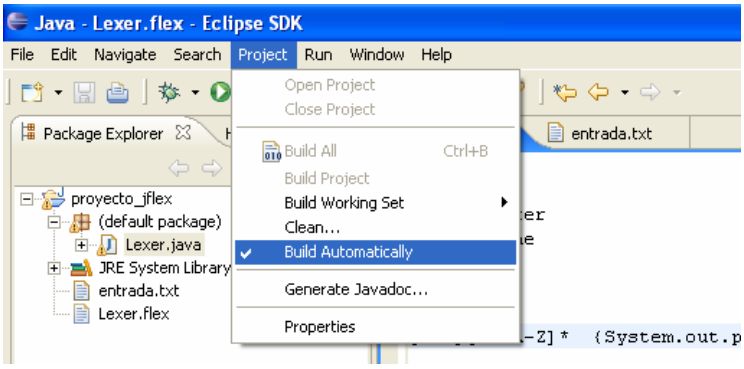

figura 17. Opción de compilación automática

También podemos acceder a las opciones de configuración de Eclipse (menú Window→Preferences…) y seleccionar el grupo de opciones 'General – Workspace' en el árbol de la izquierda. Aparece la opción 'Build automatically' que podemos desactivar y también la opción 'Save automatically befote build' que conviene activar para que nuestros ficheros se guarden antes de compilar.

Integración de Cup

Suponemos que el proyecto ya incluye el fichero 'Lexer.flex'[3], el de entrada, y que está creada la configuración para ejecutar JFlex.

7. Creación de 'Parser.cup'

Los pasos que hay que seguir para crear el fichero 'Parser.cup' son los mismos que para el fichero 'Lexer.flex' cambiando el nombre en la ventana de la figura 4.

8. Compilando con Cup

Al igual que hicimos en el apartado 4 para JFlex, creamos una nueva configuración para compilar la especificación de Cup introduciendo (tomar como ejemplo la figura 18 adaptando las rutas a los ficheros):

- ✓ en la casilla 'Name': el nombre que queremos darle a esta configuración para ejecutar Cup (p. ej. 'Cup').
- ✓ en la casilla 'Location': la ruta hasta el archivo 'java.exe', que será el encargado de ejecutar Cup.
- ✓ en la casilla 'Working Directory': la variable ${project_loc} que hace referencia al directorio donde está nuestro proyecto.
- ✓ en la casilla 'Arguments': los parámetros de Cup (ver figura 18)

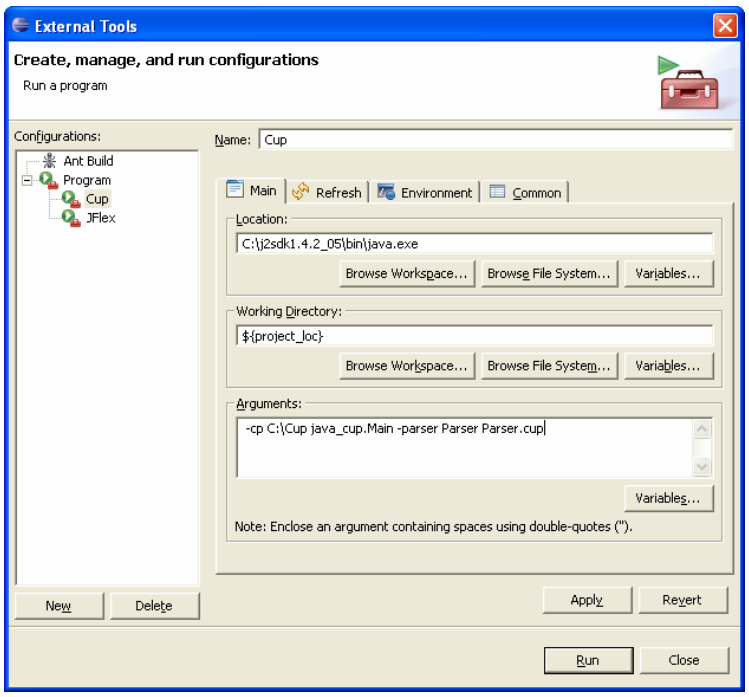

figura 18. Configuración para ejecutar Cup

[3] Se supone que no está siendo utilizada la directiva '%standalone' (incompatible con el uso de Cup)

En la pestaña ‘Refresh’ (al igual que hicimos en la figura 9 con JFlex) debemos indicarle a eclipse que actualice el proyecto una vez ejecutada esta herramienta (para que se tengan en cuenta los cambios de los ficheros Parser.java y sym.java creados o modificados por Cup).

Para terminar pulsamos el botón ‘Apply’ y luego el botón ‘Run’ para ejecutarlo por primera vez. A partir de este momento, para volver a compilar con Cup, podremos recurrir al botón de acceso rápido que tenemos en el entorno principal (figura 10).

Opcionalmente, podemos sustituir ‘Parser.cup’ en los argumentos de la herramienta Cup (ver figura 18) por la variable ${resource_name}. Esto nos permite ejecutar Cup sobre el fichero que queramos sin necesidad de que se llame ‘Parser.cup’. Para hacerlo seleccionaríamos el nombre del fichero que contiene nuestro analizador sintáctico[4] en el ‘Package Explorer’ (o la pestaña del propio fichero si lo tenemos abierto) y ejecutaríamos la opción de menú Run→External Tools→Cup. De todas formas, hay que tener en cuenta la incompatibilidad de esta configuración con el Apéndice 2 y la necesidad de que esté seleccionado el fichero adecuado cada vez que ejecutemos una herramienta externa.

9. Incorporando java_cup.runtime al proyecto

Debemos incorporar ‘java_cup.runtime.*’ a nuestro proyecto ya que contiene código fuente utilizado por la clase ‘Parser’ y puede que otras (en el Apéndice 3 se detalla un método alternativo para incluir estas clases, más eficiente aunque también más engorroso). Pulsamos para ello el botón derecho del ratón sobre el nombre del proyecto y seleccionando la opción ‘Import…’. Seleccionamos la opción ‘File system’ en la ventana que aparece y pulsamos el botón ‘Next’.

Buscamos ahora la carpeta donde está instalada la herramienta Cup y seleccionamos el subdirectorio ‘java_cup\runtime’ tal y como se muestra en la figura 19.

[4] Si ejecutamos Cup teniendo otro fichero seleccionado que no sea el ‘.cup’, nos saldrá un error, ya que se está intentando compilar con Cup dicho fichero incompatible.

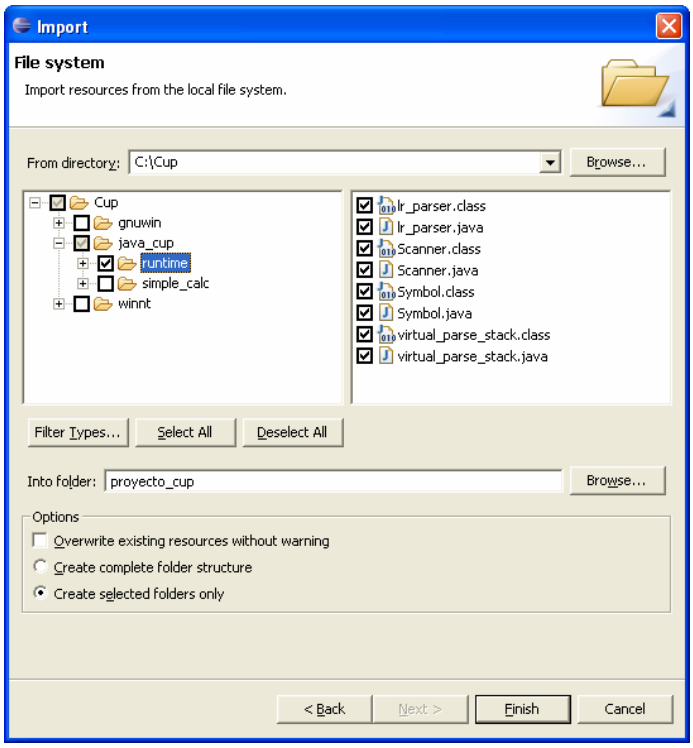

figura 19. Importando 'java_cup.runtime'

Pulsamos el botón 'Finish' y comprobamos que se ha añadido el paquete 'java_cup.runtime' a nuestro proyecto (mirando en el 'Package Explorer').

10. Creación de la clase 'Inicio'

Ahora crearemos la clase Inicio (fichero 'Inicio.java') que será la encargada de ejecutar el analizador sintáctico, conteniendo el método 'main'. Pulsamos el botón derecho del ratón sobre el nombre del proyecto que aparece en el 'Package Explorer' y seleccionamos la opción New→Class (también se tiene acceso por el menú File→New→Class).

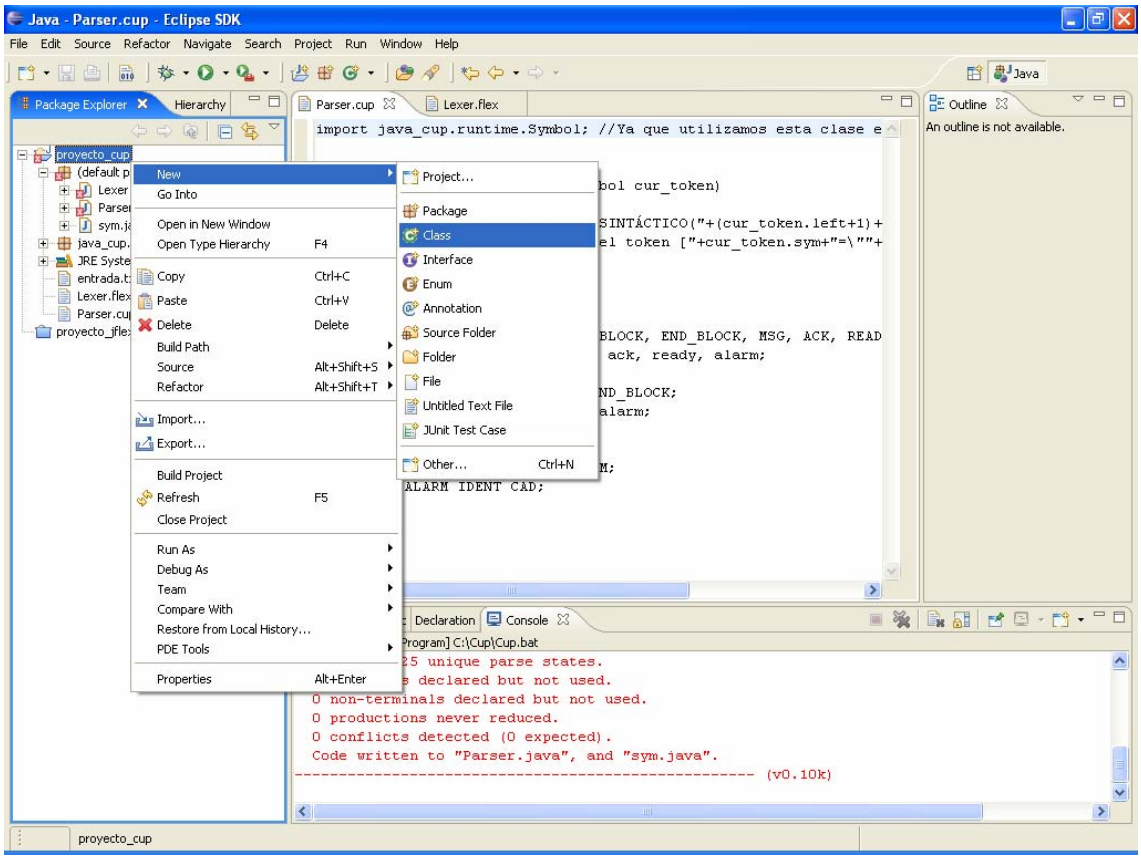

figura 20. Opción para crear una clase nueva

Se nos abre una ventana (figura 21) donde especificar el nombre de la nueva clase que vamos a crear. Si queremos podemos marcar la opción de incorporar el método 'main' para que eclipse escriba su prototipo por nosotros.

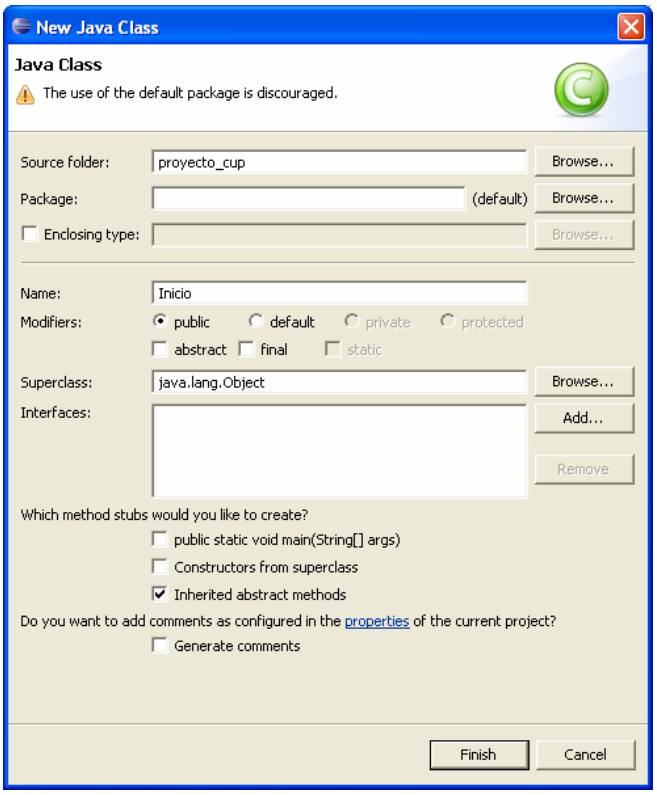

figura 21. Creación de una clase nueva

Una vez introducidos los datos, pulsamos el botón 'Finish' y aparece en el entorno la nueva clase lista para completar su definición.

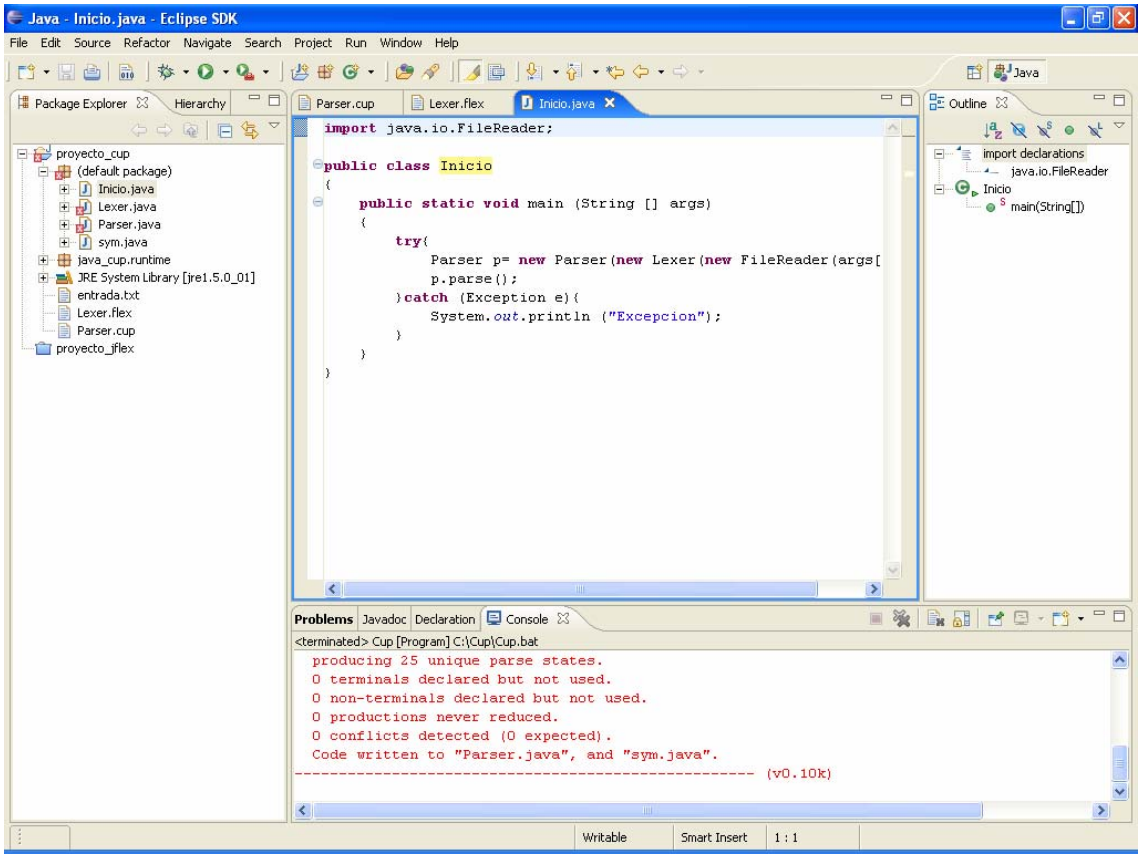

figura 22. Nueva clase 'Inicio' con el método 'main'

11. Ejecución del proyecto (con Cup)

Al igual que hicimos en el apartado 5 creamos una configuración de ejecución pero introduciendo la clase 'Inicio' como clase principal (figura 23).

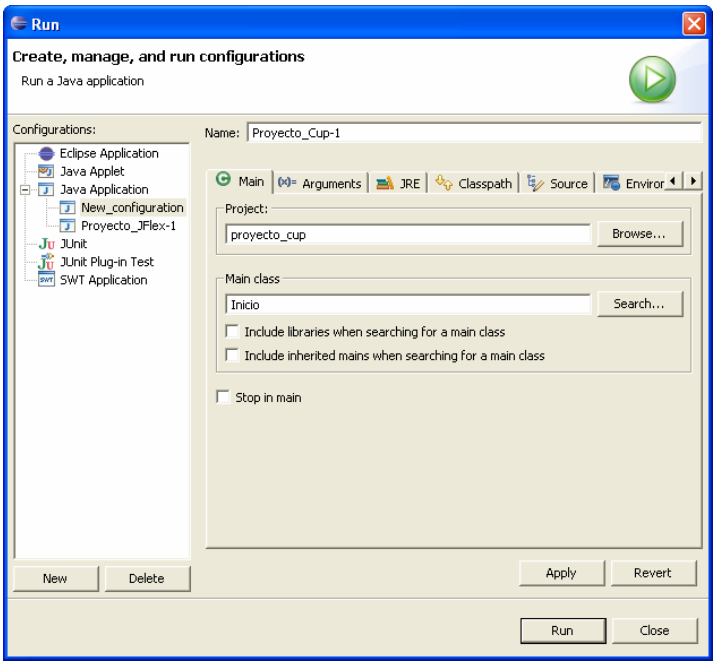

figura 23. Configuración de ejecución con Cup

Además de esto accedemos a la pestaña 'Arguments' y especificamos el fichero de entrada que vamos a pasarle a nuestro analizador.

Para terminar pulsamos el botón 'Apply' y luego el botón 'Run' para ejecutarlo por primera vez. A partir de este momento, para volver a ejecutar el proyecto según esta configuración, podremos recurrir al botón de acceso rápido que tenemos en el entorno principal (figura 16).

Apéndice 1 – Línea de órdenes

Para ejecutar nuestro proyecto desde una ventana de MSDOS o línea de órdenes, solo tenemos que posicionarnos en la carpeta de nuestro proyecto y ejecutar la clase principal con el comando: (‘Inicio’)

“java –cp . *ClasePrincipal fichero_de_entrada*”

La clase principal, siguiendo el contenido de esta guía, será ‘Lexer’ si estamos trabajando sólo con JFlex (con la directiva ‘%standalone’), o ‘Inicio’ si estamos combinando JFlex (sin la directiva ‘%standalone’) y Cup.

Apéndice 2 – Automatizando la compilación

Podemos incluir las tareas de compilación de JFlex y Cup en las tareas que debe realizar el entorno para compilar nuestro proyecto por completo. De esta manera, podremos modificar el fichero 'Lexer.flex' y/o 'Parser.cup' y darle a ejecutar todo el proyecto, mostrándose los efectos de la modificación en los resultados, ya que estos ficheros se compilarán junto al resto de ficheros de Java.

Comenzamos accediendo a las propiedades de nuestro proyecto, para lo que pulsamos con el botón derecho del ratón sobre el nombre del proyecto en el 'Package Explorer', seleccionando la opción 'Properties'. Aquí seleccionamos la opción 'Builders' donde aparecerán los constructores que participan en la compilación del proyecto completo. Inicialmente sólo se encuentra el 'Java Builder' (figura 24), que es el encargado de compilar los ficheros de Java.

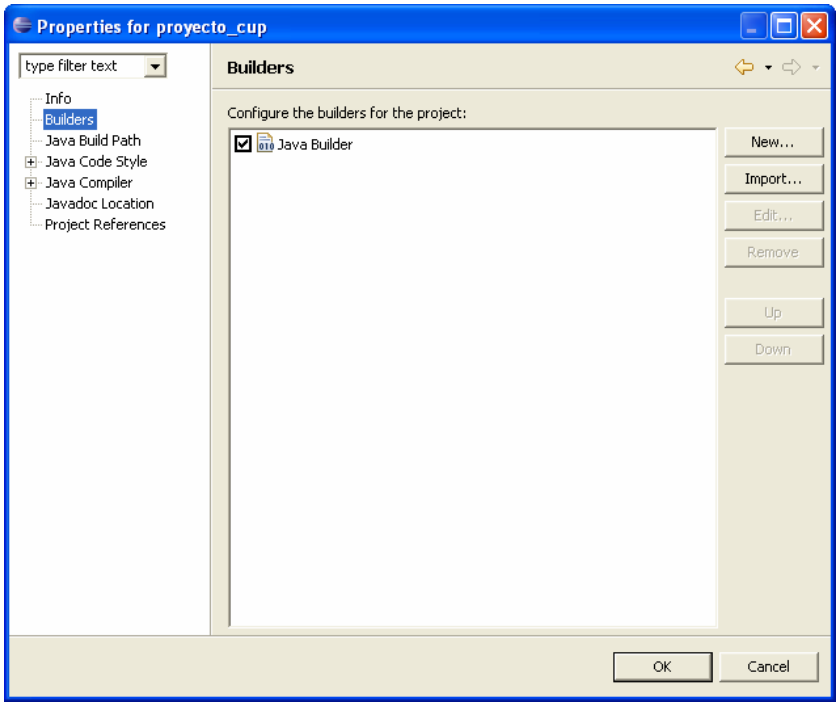

figura 24. Lista de constructores del proyecto

Como queremos que se compilen también los ficheros de JFlex y Cup, añadiremos los constructores que se encarguen de ello pulsando el botón 'Import…'. De ahí seleccionamos los constructores que hemos creado anteriormente para ejecutar las herramientas JFlex y Cup y pulsamos 'OK' para añadirlos a la lista de constructores de nuestro proyecto.

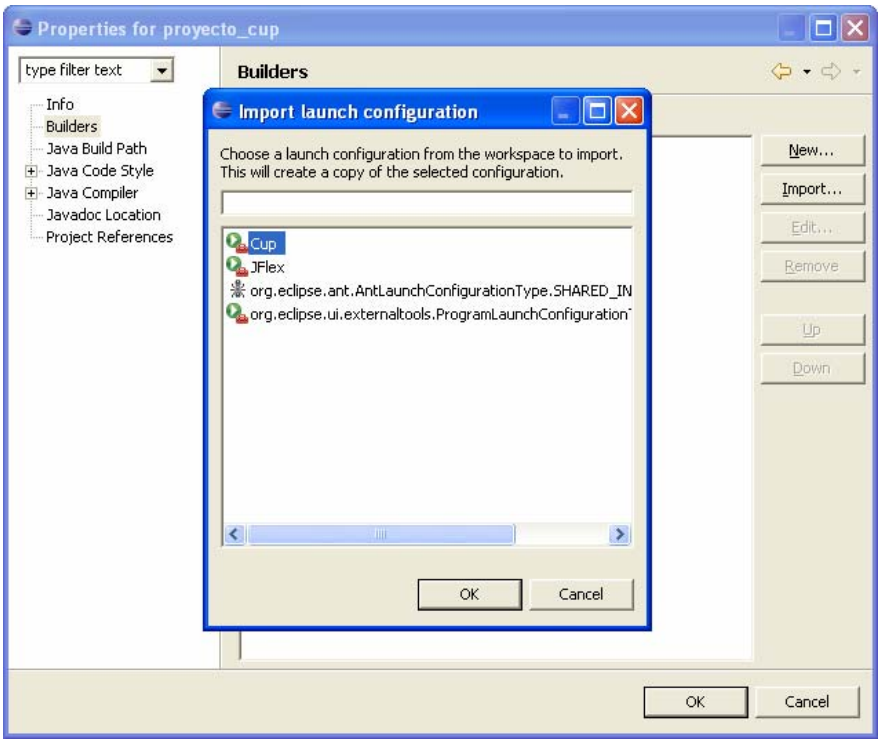

figura 25. Seleccionando los constructores de Cup y JFlex

Una vez incluidos ambos constructores a la lista los ordenamos mediante los botones 'Up' y 'Down' de modo que queden en el orden que aparece en la figura 26.

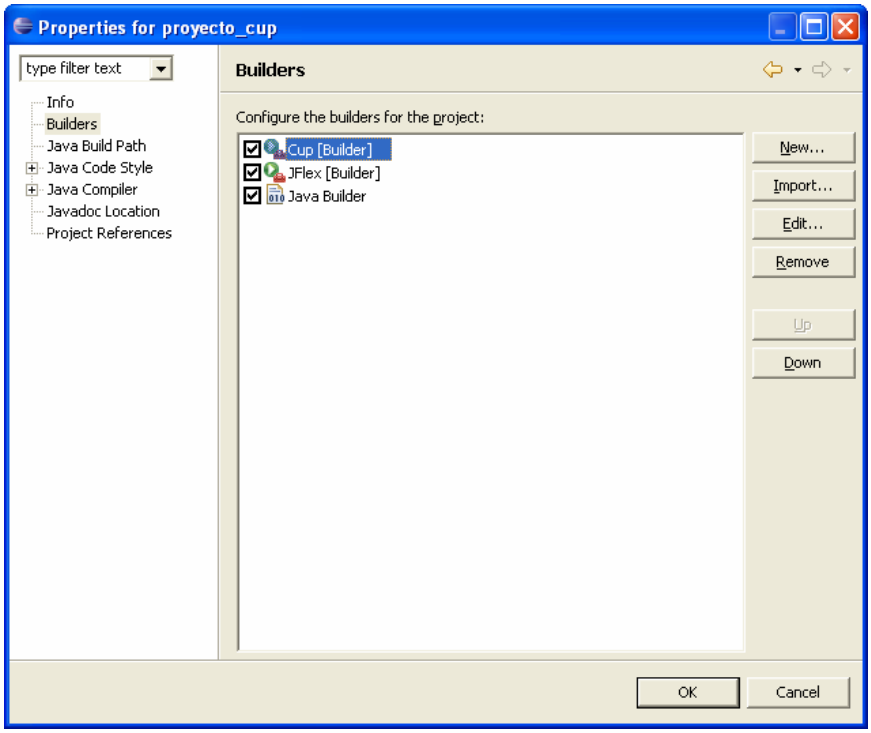

figura 26. Constructores que intervienen en el proyecto

Apéndice 3 – Incluir clases auxiliares (Cup)

En el capítulo 9 veíamos como incluir algunas clases de Cup que son necesarias para que no se generen errores al compilar. Aquí presentamos una forma alternativa más eficiente que enlaza dichas clases con nuestro proyecto en lugar de copiar los ficheros directamente. Partimos por tanto de la suposición de que no se ha llevado a cabo el método descrito anteriormente.

Si editamos los ficheros Java que nos han generado las herramientas JFlex y Cup obtendremos algo como esto:

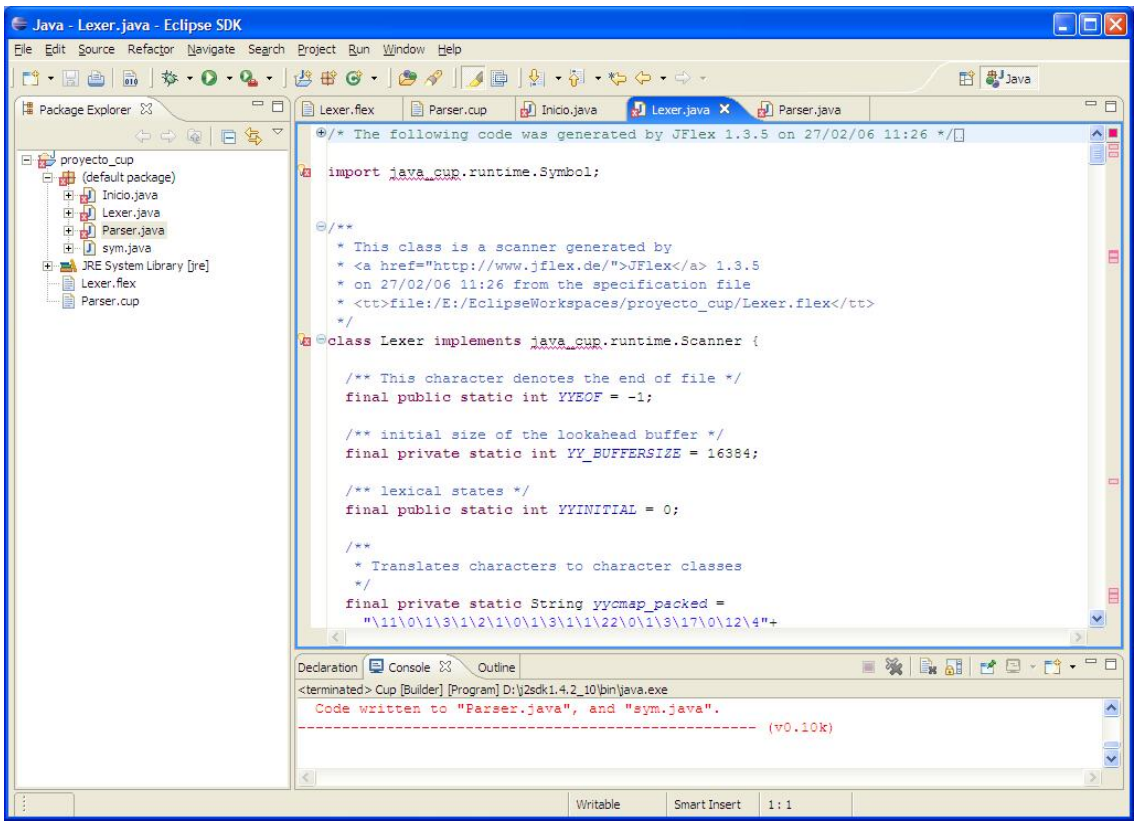

figura 27. Errores por falta de clases auxiliares de Cup

Como vemos, la compilación automática nos indica numerosos errores en el código. Vemos que todos son debidos a que no se encuentra el código de varias clases, como Symbol y Scanner, lo cual se debe a su vez a que el compilador no sabe cómo interpretar el símbolo java_cup. Podemos comprobar esto observando los subrayados en color rojo o acercando el puntero hacia los símbolos de la bombilla (parte izquierda de la ventana de edición).

Para solucionar esto hay que indicar al compilador dónde se encuentra el paquete java_cup, que contiene las clases anteriores.

Una manera de conseguir esto en Eclipse es añadiendo “class folders” al proyecto. Una “class folder” (carpeta de clases) es una indicación al compilador de que un conjunto de clases van a ser necesarias para el proceso de compilación, además de las que nosotros creemos en el proyecto.

Veamos cómo se indican estas carpetas.

Hacemos clic con el botón derecho sobre el icono del proyecto en el “Project explorer” y seleccionamos “Properties”. Seleccionamos la pestaña “Libraries”:

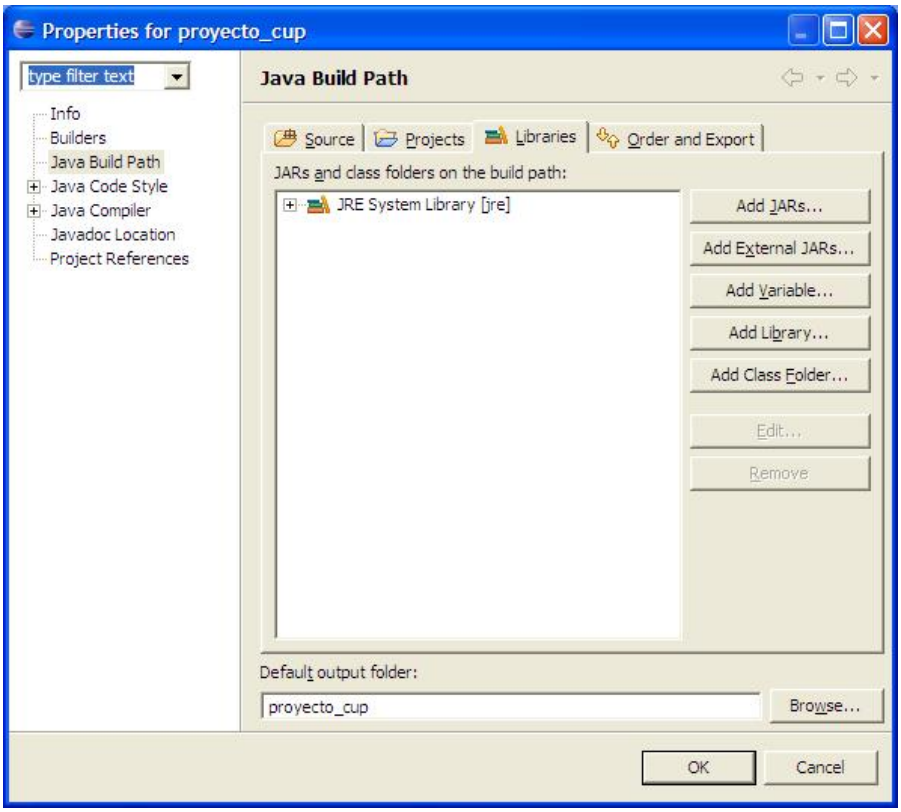

figura 28. Propiedades del proyecto - Libraries

Seleccionamos la opción “Add Class Folder” y a continuación “Create New Folder”. El nombre que le demos a la carpeta puede ser cualquiera, ya que simplemente es un identificador en el entorno de Eclipse. Por ejemplo podemos llamarla “Clases Cup”. Ahora hay que indicar en qué lugar del disco se encuentran las clases auxiliares ya compiladas. Para ello pulsamos “Advanced”, marcamos la casilla “Link to folder in the file system” y escribimos C:\Cup (o el lugar donde tengas instalado Cup). Por último pulsamos OK.

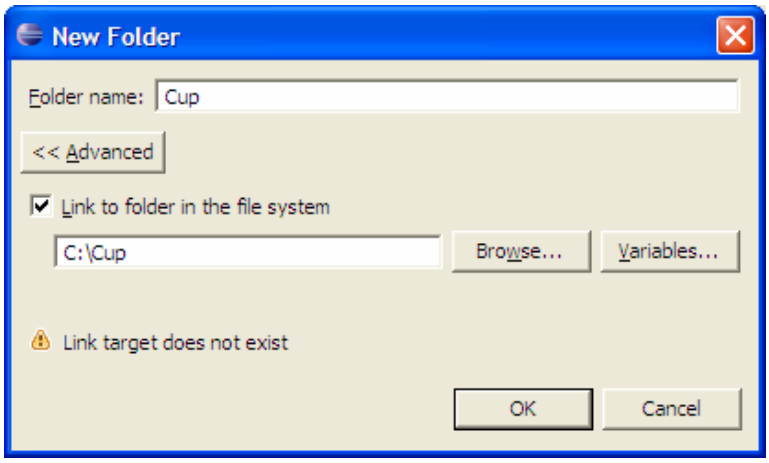

figura 29. Enlazando las clases de Cup al proyecto

Nótese que lo anterior no crea una carpeta nueva en el sistema de ficheros del disco duro. Simplemente asocia el identificador “Clases Cup” a un conjunto de clases de interés.

Hasta el momento tenemos lo siguiente:

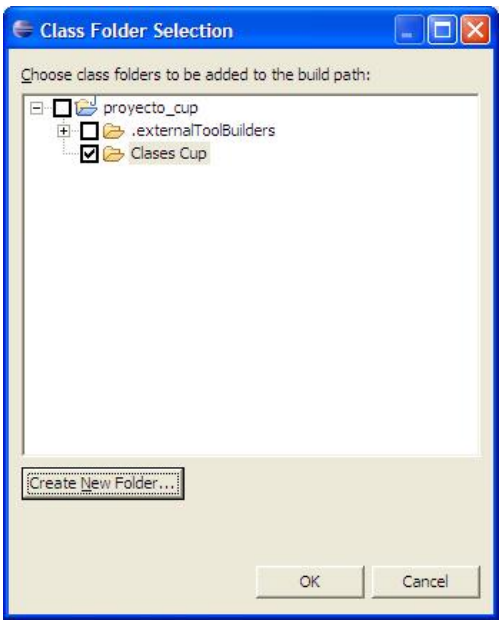

figura 30. Clases a añadir al proceso de compilación

Pulsamos de nuevo OK. Aparecerá el siguiente aviso:

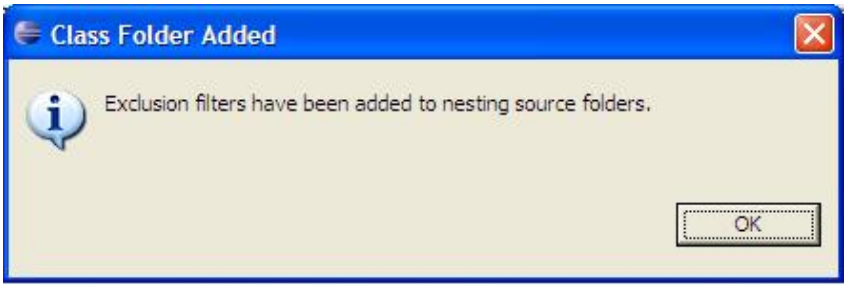

figura 31. Mensaje de aviso de filtros de exclusión

Esto simplemente indica que, de manera automática, Eclipse ha aplicado filtros de exclusión a las clases que le hemos indicado. Dichos filtros permiten que dichas clases no se compilen sino que simplemente se utilicen.

La configuración hasta el momento es la siguiente:

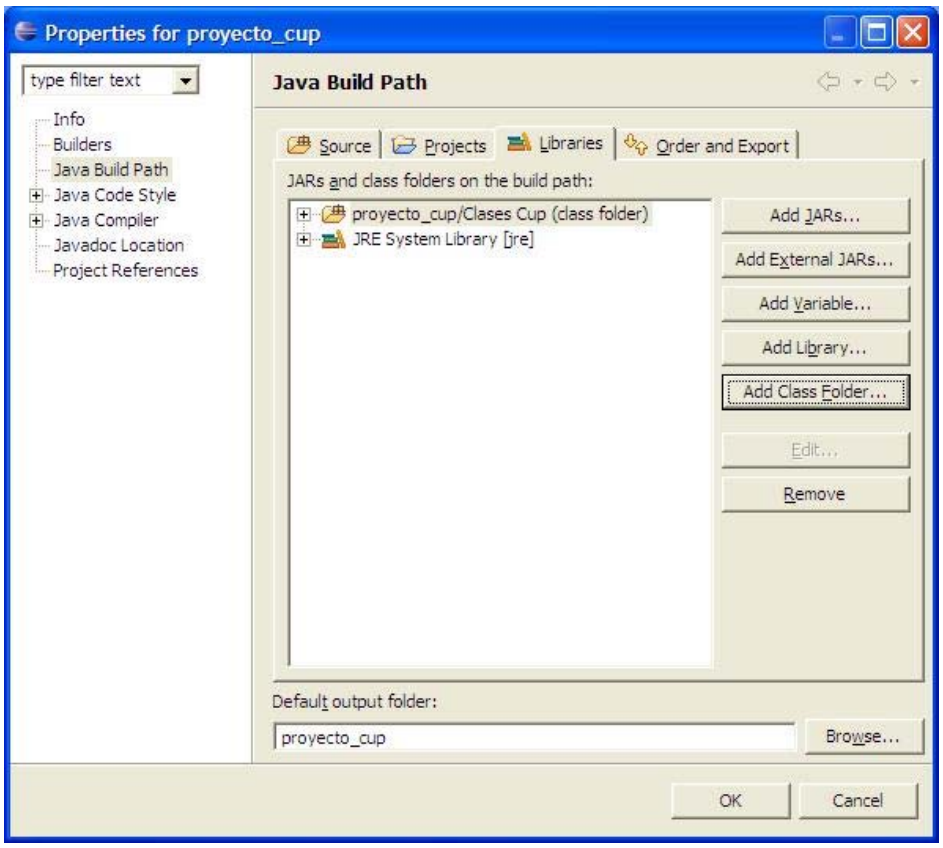

figura 32. Propiedades del proyecto con las clases auxiliares añadidas

Ahora simplemente queda seleccionar la pestaña “Order and Export”. Observamos que la carpeta que acabamos de crear aparece sin marcar. La marcamos, ya que así el compilador tendrá en cuenta dichas clases:

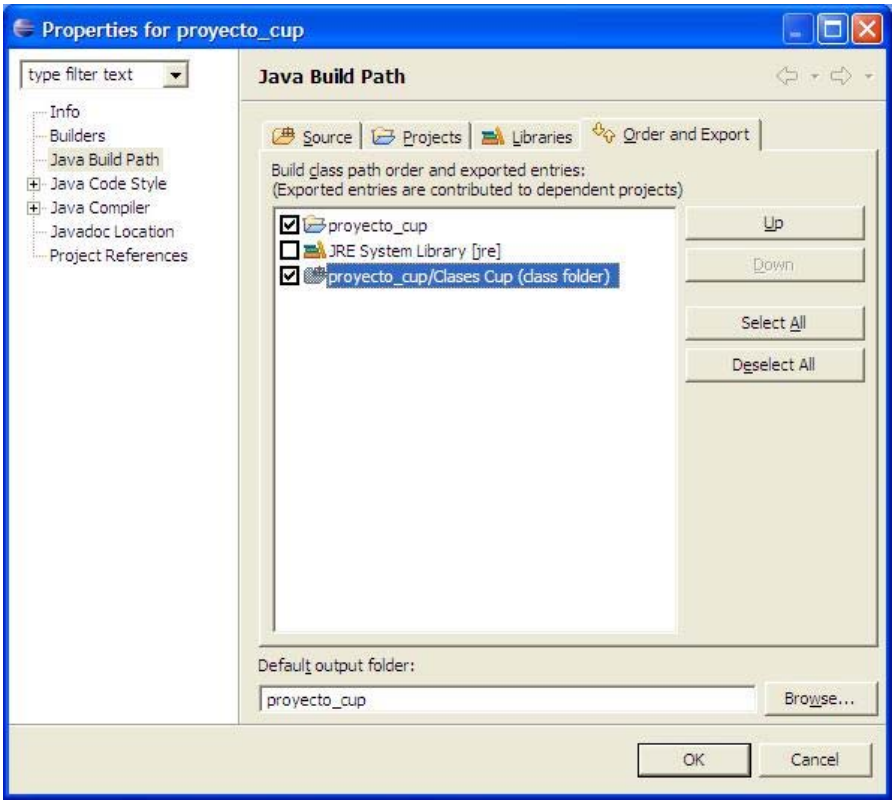

figura 33. Propiedades del proyecto - Order and Export

Por último pulsamos OK.

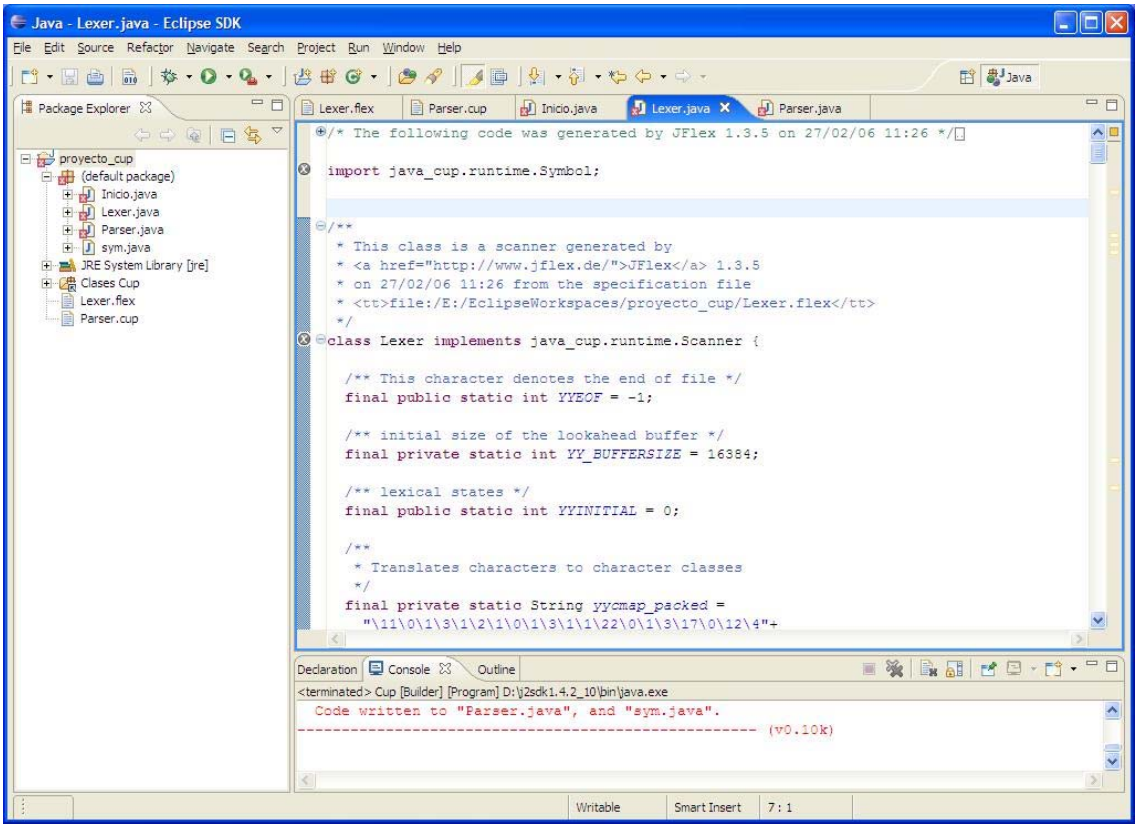

figura 34. Ficheros generados sin errores

Como vemos, los subrayados rojos que indicaban error han desaparecido de la ventana de edición. Aún quedan marcas en color azul apagado donde antes estaban los errores. Dichas marcas nos indican que ya no hay problemas pero que aún debemos hacer la compilación. Para ello seleccionamos Project y luego Build All, o simplemente pulsamos la tecla Control junto con la tecla B.

Estas “class folders” permiten incluso proteger los ficheros de clases originales de escrituras por error. Para comprobar esto, hacemos clic con el botón derecho sobre el icono de “Carpeta Cup” (en el Project Manager). Deberíamos de obtener algo así:

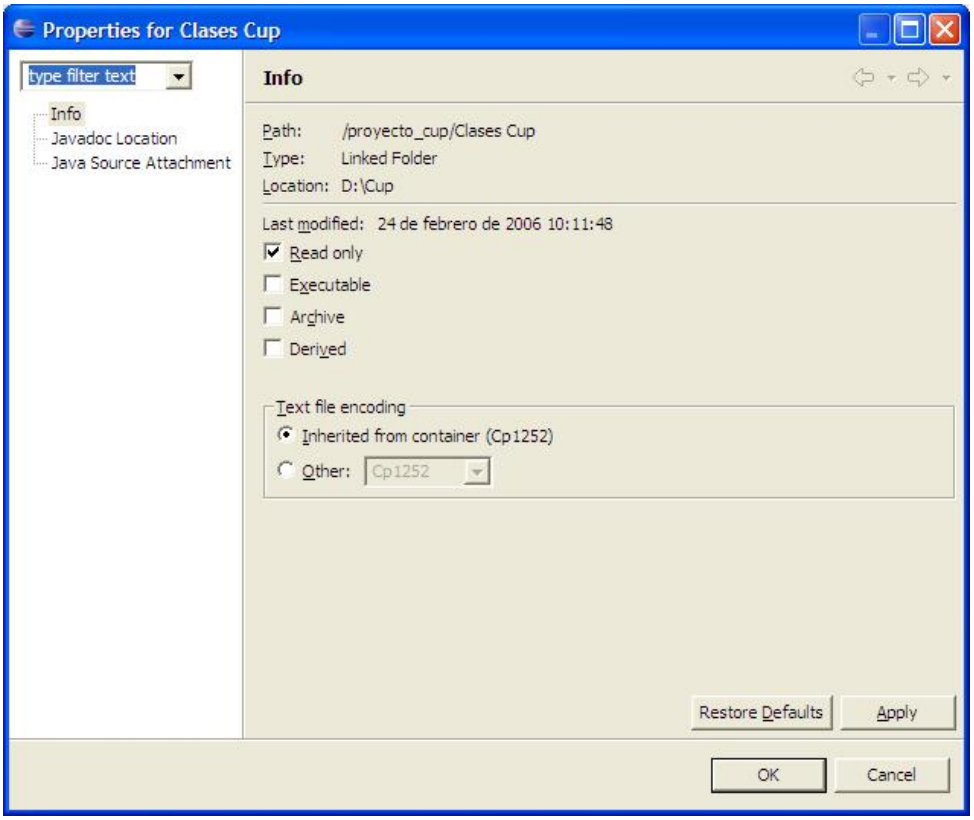

figura 35. Propiedad Read-only de las clases externas incluidas