Información técnica sobre xeo

Nombre del programa

xeo (version 2.0)

Autor

Daniel González Trabada: dgtrabada@yahoo.com

Breve descripción del programa

El programa es un gestor de proyectos que ofrece al usuario una interfaz intuitiva para modelizar, representar y transformar estructuras atómicas.

El programa es capaz de leer estructuras atómicas escritas en diversos formatos, como por ejemplo en Fireball[1], CASTEP[2], VASP[3],... , en estos casos también puede obtener información diversa. Como los vectores de red (con los que podemos representar mas átomos a partir de la celda unidad). Además, se pueden ver los átomos que están fijos (son los círculos negros).

También podemos cargar películas escritas en un formato *.xyz, el mismo que leen muchos programas como JMOL[4]. Una vez abierta, podremos interpretar la dinamia y representar las distancias o coordenadas de los átomos que queramos utilizando para ello una herramienta de cálculo simbólico.

Para algunos de estos programas, además de representar y cambiar las estructuras atómicas, podemos representar las estructuras de bandas y las densidades de estados, como es en el caso de Fireball. Podemos así mismo, representar imágenes obtenidas con un microscopio de efecto túnel (STM) o un microscopio de fuerzas atómicas (AFM).

El programa es muy flexible. Es capaz de obtener la entrada de datos de diversas formas. Es un programa versátil, es decir, pueda ser usado para representar y transformar puntos, flechas y superficies en tres dimensiones de una forma general.

Fireball[1] is a suite of programs designed to perform calculations using ab initio tight-binding molecular dynamics.
CASTEP[2], is a commercial (and academic) software package which uses density functional theory with a plane wave basis set to calculate electronic properties of solids from first principles.
VASP[3] is a package for performing ab initio quantum mechanical molecular dynamics (MD) using either Vanderbilt pseudopotentials.
JMOL[4] is a Java molecular viewer for three-dimensional chemical structures

Lenguaje de programación y dependencias

El programa está escrito en java, lenguaje interpretado y multiplataforma. El código java es compilado en bytecode de java, y compactado en un archivo jar que contiene todo el código del programa. El programa depende únicamente de las librerías estándar de java y utiliza características de la versión: **java version "1.5.0_04"**
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_04-b05)
Java HotSpot(TM) Client VM (build 1.5.0_04-b05, mixed mode, sharin

Entorno operativo

Puede ejecutarse en cualquier sistema que disponga de una máquina virtual de java completa. La lista de plataformas posibles incluye PC's compatibles con un sistema operativo POSIX basado en Linux, BSD, Solaris o Mac OS X, o de la familia Windows, pero también puede eventualmente incluir PDAs y dispositivos de otro tipo.
El programa se puede ejecutar escribiendo en línea de comandos "java -jar xeo.jar" de forma general. Para el caso de utilizar GNU-Linux podemos también ejecutar el script "./install" que generará un archivo xeo el cual es ejecutable y podemos introducir en el PATH de nuestro sistema, de tal forma que todos los usuarios tengan el programa instalado. En el caso de utilizar Windows, con hacer doble-click en xeo.jar se ejecutará automáticamente.

Listado de Ficheros

Dentro de la carpeta código fuente, nos encontramos dos carpetas:

1. “codigo” : En esta carpeta estaría todo el código java
2. “codigo_paquete” : En esta carpeta nos encontramos el mismo código que en la carpeta “codigo”, pero esta vez estaría organizado por carpetas dependiendo del paquete al que corresponda.

- **xeo**
  - **xeo.java** : Es la parte principal del programa, desde aquí se gestionan los proyectos, los hilos que leen los archivos *xyz .... En esta clase está incluida la interfaz gráfica de la mayoría de las clases, así como la manipulación de sus variables.
  - **C3D.java** : Esta clase se ocupa de transformar el espacio tridimensional (coordenadas de los átomos) en un espacio bidimensional (coordenadas en la pantalla)
  - **Main.java** : Nos da la opción de ejecutar el programa en línea de comandos o visualizarlo en modo gráfico. Para obtener la ayuda en modo texto escribir “xeo -help”.
  - **atom.java** : En este archivo está definida la clase atom, contiene variables y métodos que nos dan información acerca de la posición del átomo, sus primeros vecinos, distancias, posición en la pantalla, etc...
  - **bulk.java** : Ajusta a la ecuación de Birch “Eleni Ziambaras and ElsebethSchroder, Phys. Rev B. 68, 064112 (2003)” la curva de la energía frente al parámetro de red del volumen de una superficie, y obtiene la Energía mínima (eV), el parámetro de red (Å) a dicha energía y el Bulk Modulus (Gpa).
  - **calcStructure.jav**a : Esta clase se encarga de re-escalar, rotar respecto a un eje, rotar respecto a una dirección y rotar respecto a las posiciones de los átomos una determinada estructura cargada. Así como, la posibilidad de hacer diferentes operaciones sobre la estructura.
  - **info_bas.java** : Esta clase se encarga de organizar, ordenar, copiar, pegar, establecer los enlaces y los ángulos entre los objetos atoms. Así como la repetición de los objetos atoms teniendo en

cuenta la periodicidad. Esta clase es también la encargada de cargar los archivos con un formato *xyz.

- **mol.java** : Esta clase se encarga de pintar las coordenadas de los átomos, sus enlaces, las flechas, los ejes, y las superficies, utiliza info_bas.java, de donde lee la estructura.
- **plugBabel.ja**va : Esta es la clase responsable de la comunicación con xeoBabel. Obtiene de xeoBabel la estructura en un formato de xeo por medio del método read en xeoBabel. Para guardar la estructura utiliza el método write de xeoBabel.
- **povray.java** : Esta clase se encarga de obtener la entrada para el programa Povray y obtener la imagen.

- **calc_xyz**
  - **Calc_bas.java** : Utiliza la calculadora hecha en calculadora.java para hacer cálculos entre una o mas estructuras, es decir, es una calculadora donde ahora las variables corresponden a los ficheros de las estructuras.
  - **Calc_diff_xyz.java** : Es la interfaz gráfica de Calc_bas.java
  - **JCalc_xyz.java** : Podemos representar las variables para dinámica molecular, es decir, la coordenada X del átomo primero, o la distancia entre los átomos 4 y 6, o los ángulos que formar durante la dinámica. Así como sus promedios, para ello lo único que tenemos que hacer es escribir las variables simbólicas con mayúsculas, ejemplo: d[3][7] = representa la distancia entre el átomo 3 y 7, mientras que D[3][7] representa el promedio.
  - **Main.java** : clase principal de calc_xyz, gracias a ella podemos utilizar calc_xyz en linea de comandos
  - **SustCalculator.java** : Esta clase se encarga de comunicar Calc_xyz.java con Calc.java.

- **calculadora**
  - **calculadora.java** : calculamos numéricamente una expresión escrita en cálculo simbólico.

- **fireball**
  - **Dos_bands.java** : La utilizamos para pintar las densidades de estados y la estructura de bandas obtenidas con Fireball

- **Jbegin.java** : Interfaz gráfica de begin.java
- **Jhopping.java** : Interfaz gráfica de hopping.java.
- **begin.java** : Lo utilizamos para visualizar los pseudopotenciales *.pp , los archivos con los átomos neutros *.na0, las funciones de onda *.wf y los potenciales *.na creados con el programa begin perteneciente a Fireball.
- **fireball.java** : Clase responsable de la comunicación con xeo.
- **hopping.java** : Con esta clase podemos ver los hopping obtenidos con Fireball, y ajustar el decaimiento.

- **XeoBabel**
  - **xeoBabel.java** : clase principal, gracias a ella podemos utilizar xeoBabel en linea de comandos.
  - **plugin.java** : Esta clase es la responsable de la comunicación con xeo, o cualquier otro programa. Utiliza dos métodos read y write, read lee los diferentes formatos y devuelve los archivos leídos en un formato de "xeo". El otro método importante es write, este método a partir del formato xeo escribirá los archivos en el formato que corresponda.
  - **format.java** : en esta clase separamos los Strings por columnas, por líneas, por muchos tipos de clases primitivas, como por ejemplo (double. String, int). Emulamos la salida en formato de Fortran ...
  - **periodicTable.java** : Tabla periódica de los elementos, aquí encontramos los valores para el radio covalente, el color, ..., hemos tomado los mismos valores de JMOL.

    Las siguientes clases se utilizan para leer y escribir los archivos en diferentes formatos:
    - **abinit.java** : //http://www.abinit.org/
    - **bas.java** : //http://www.fireball-dft.org/web/fireballHome; http://www.efireball.
    - **castep.java** : //http://en.wikipedia.org/wiki/CASTEP
    - **fireball.java** : //http://www.fireball-dft.org/web/fireballHome; http://www.efireball.cz/
    - **fireball_TG.java** : //http://www.fireball-dft.org/web/fireballHome; http://www.efireball.cz/

        - **vasp.java** //http://cms.mpi.univie.ac.at/vasp
        - **xeo.java** //https://sourceforge.net/projects/xeo/
        - **xyz.java** : http://en.wikipedia.org/wiki/XYZ_file_format

- **java_STM_AFM**
    - **atomo.java** : Guarda la información de la estructura atómica, es decir, la posiciones, el numero atómico y la periodicidad.
    - **java_STM_AFM.java** : Es la parte gráfica de stm.java, donde se gestiona toda la información, se cargan los archivos, vemos la imagen de salida, las direcciones sobre la superficie, etc ..
    - **periodicTable.ja**va : Tabla periódica de los elementos, aquí encontramos los valores para el radio covalente, el color, ..., hemos tomado los mismos valores de JMOL.
    - **stm.java** : Esta clase es la encargada de realizar todos los cálculos para interpretar los datos, así como los cambios de color, pintar direcciones, y dibujar los átomos.

- **dialogo**
    - **chooser.java** : Diálogo para seleccionar los ficheros.
    - **help.java** : Muestra en forma de web un archivo escrito en formato html.
    - **helpURL.java** : Muestra en forma de web una URL escrita en formato html.
    - **show_picture.jav**a : Muestra una imagen cargada:

- **editor**
    - **editor.java** :Es un simple editor de texto al que llama el programa cada vez que queremos visualizar datos en la pantalla.

- **jcalc**
    - **jCalc.jav**a : Es la interfaz gráfica de la clase calculadora.java, en ésta podremos probar si la función que vamos a utilizar existe, o si la expresión puede tener algún error de escritura. También podemos utilizarla como una simple calculadora.
    - **statistic.java** : Analiza estadísticamente la columna X de un archivo, mostrándonos cuánto vale la media y la variación típica.

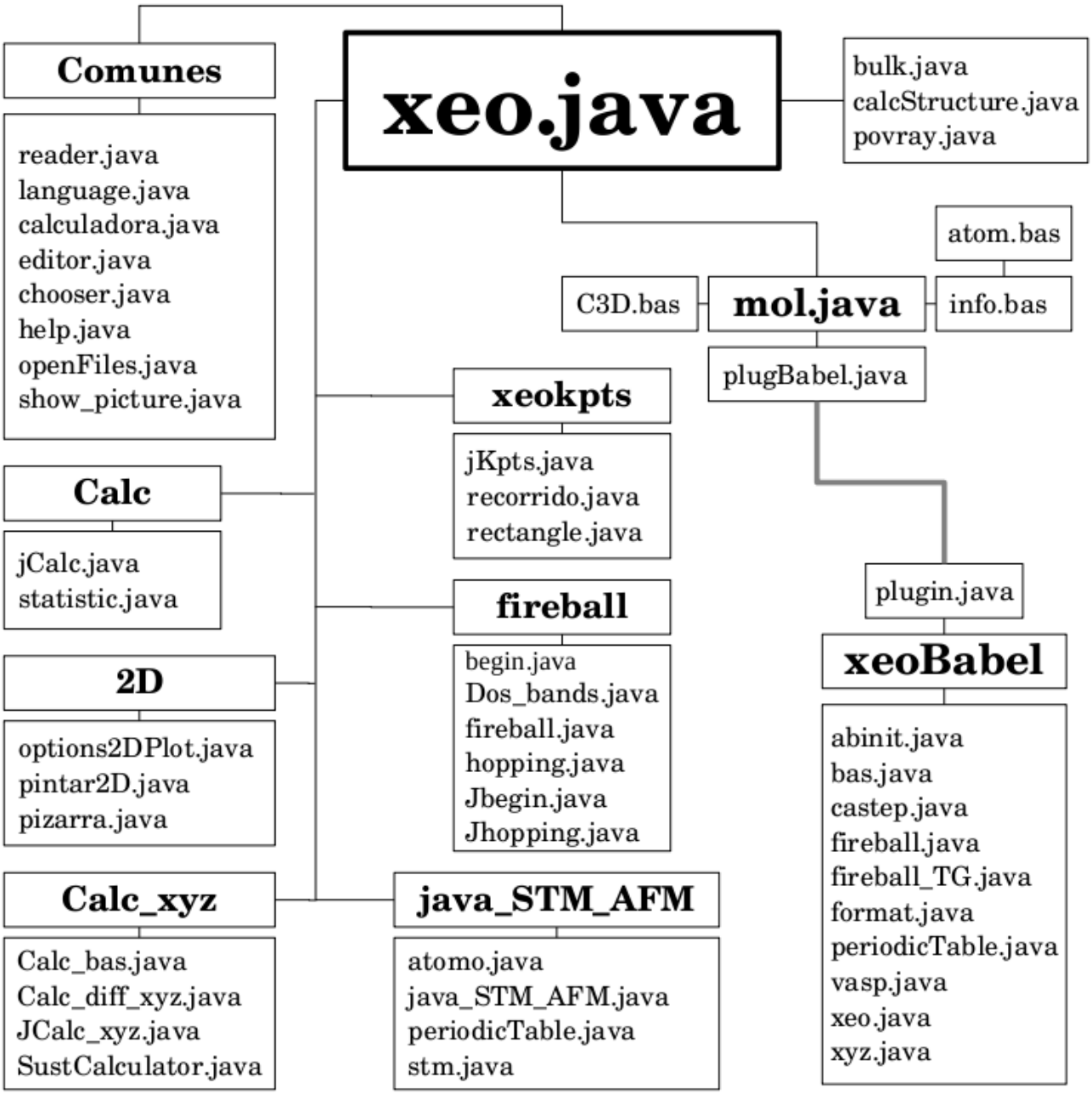
- **Language**
  - **language.java** : Aquí están definidos los posibles idiomas con los que podemos utilizar xeo.
  - **main.java** : clase principal de language, gracias a ella podemos utilizar language en línea de comandos

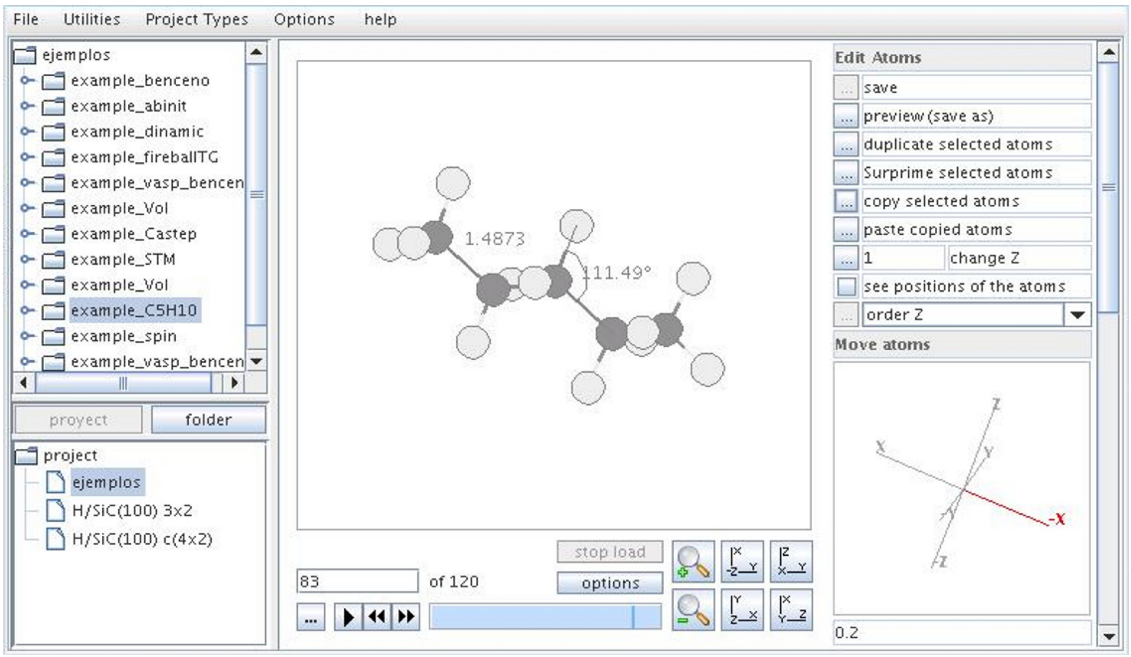
- **pintar2D**
  - **Options2DPlot.ja**va : Opciones de pintar2D.java, es decir, color de las gráficas, títulos, fuentes de los caracteres, etc ...
  - **pintar2D.java** : esta clase es la responsable de la representación de las gráficas en dos dimensiones. Tenemos múltiples posibilidades: un archivo XY, varios archivos XNY, tomar la X de un archivo y la Y de otro, graficar varias funciones tomando las X desde archivos y columnas diferentes y las Y de archivos y columnas también diferentes ...
  - **pizarra.java** : aplicación gráfica para poder utilizar pintan2D por separado.

- **reader**
  - **main.java** : clase principal, nos da la versión de la clase reader.java
  - **reader.java** : en esta clase separamos los Strings por columnas, por líneas, por muchos tipos de variables primitivas, como por ejemplo (doulbe. String, int). Emulamos la salida en formato de Fortran ...

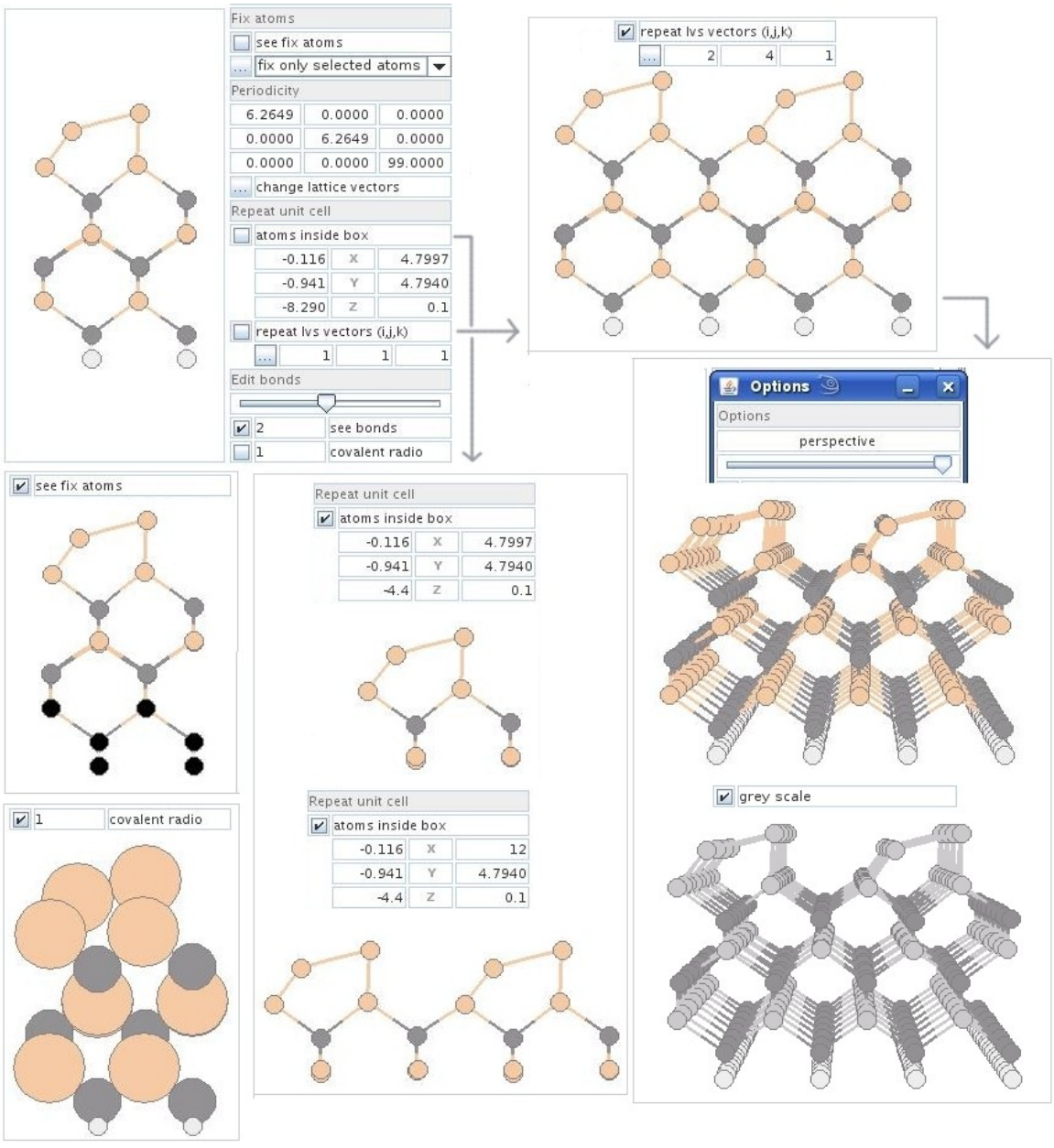
- **xeoKtps**
  - **jKpts.java** : Interfaz gráfica de xeoKpts. Aquí podemos obtener los puntos especiales K mediante el método Monkhorst-Path.
  - **recorrido.java** : Hacemos recorridos en tres dimensiones de los puntos especiales K, para luego representar las bandas de los sólidos.
  - **rectangle.java** : Hacemos recorridos en dos dimensiones de los puntos especiales K, para luego representar las bandas de las superficies

Diagrama de flujo





Formato interno de xeo:



```
10

Si     0.0000   -1.5419    7.9376   0 0 0
C      1.5733   -1.5706    6.8144   0 0 0
Si     1.5733    0.0038    5.6689   0 0 0
C      0.0000    0.0003    4.5251   0 0 0
Si     0.0000   -1.5730    3.3931   0 0 0
C      1.5733   -1.5732    2.2437   0 0 0
Si     1.5733    0.0000    1.1125   1 1 1
C      0.0000    0.0000    0.0000   1 1 1
H      0.0000   -0.8394   -0.6927   1 1 1
H      0.0000    0.8394   -0.6927   1 1 1
3.146  0.000     0.000
0.000  3.146     0.000
0.000  0.000    99.000
```