Manual de desenvolupador

Java-KLASS versió 6

5 de novembre de 2009

Índex

Capítol 1

Eines

Per a poder crear noves funcionalitats o millorar les que ja existeixen dins del **Java-KLASS** s'han de tenir instalades unes eines bàsiques depenent de la plataforma utilitzada:

1.1 Windows

En primer lloc, necesitem un entorn de desenvolupament Java i un editor per poder escriure els nostres programes

- Java: Utilitzarem la versió de desenvolupador j2sdk 1.4.1. o superior
  Si es tenen dubtes sobre la seva instal·lació mirar:
  `http://java.sun.com/products/archive/j2se/1.4.1_07/jsdk/install.html`.

- Editor: Podem utilitzar qualsevol editor per crear els nostres .java pero recomanen *Crimson* que, a més de ser gratuit, ofereix molt recursos interesants per un desenvolupador, es pot baixar de `http://www.crimsoneditor.com/`.
  Una alternativa al *simple* editor, és instal·lar un entorn JBuilder.

- Latex: Una distribució de LaTeX amb un compilador i un visualitzador, com per ejemple MiKTeX. La distribució MiKTeX es pot descarregar i instal·lar fent servir l'executable que es troba a:
  `http://www.tex.ac.uk/tex-archive/systems/win32/miktex/setup/setup.exe`
  o a:

`http://www.miktex.org/setup.html` en l'apartat DOWNLOADS. Per poder fer servir, la sortida latex del **Java-KLASS** hem de configurar les opcions del menu *Fitxer*, dintre l'apartat *Configuració* indicant d'intre els *Executable* el *path* on es troba l'arxiu de Latex, Dvi...etc, com es mostra en la figura 1.1.

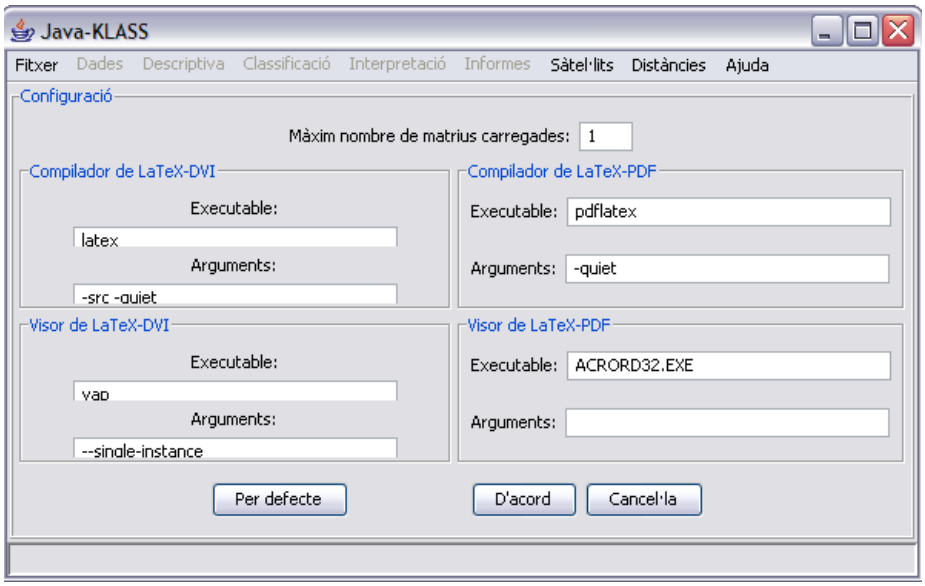


Figura 1.1: Configuració

Capítol 2

Diagrama de Clases

En aquest capítol es mostra el diagrama de clases actual, per fer al usuari més comprensible l'herència i dependència entre les classes de **Java-KLASS**. Primer es presenta el diagrama de la Capa de presentació i a continuacio el diagrama de la Capa nucli.

Figura 2.1: Diagrama de classes general de la capa de presentació

Figura 2.2: Diagrama de classes general de la capa de presentació

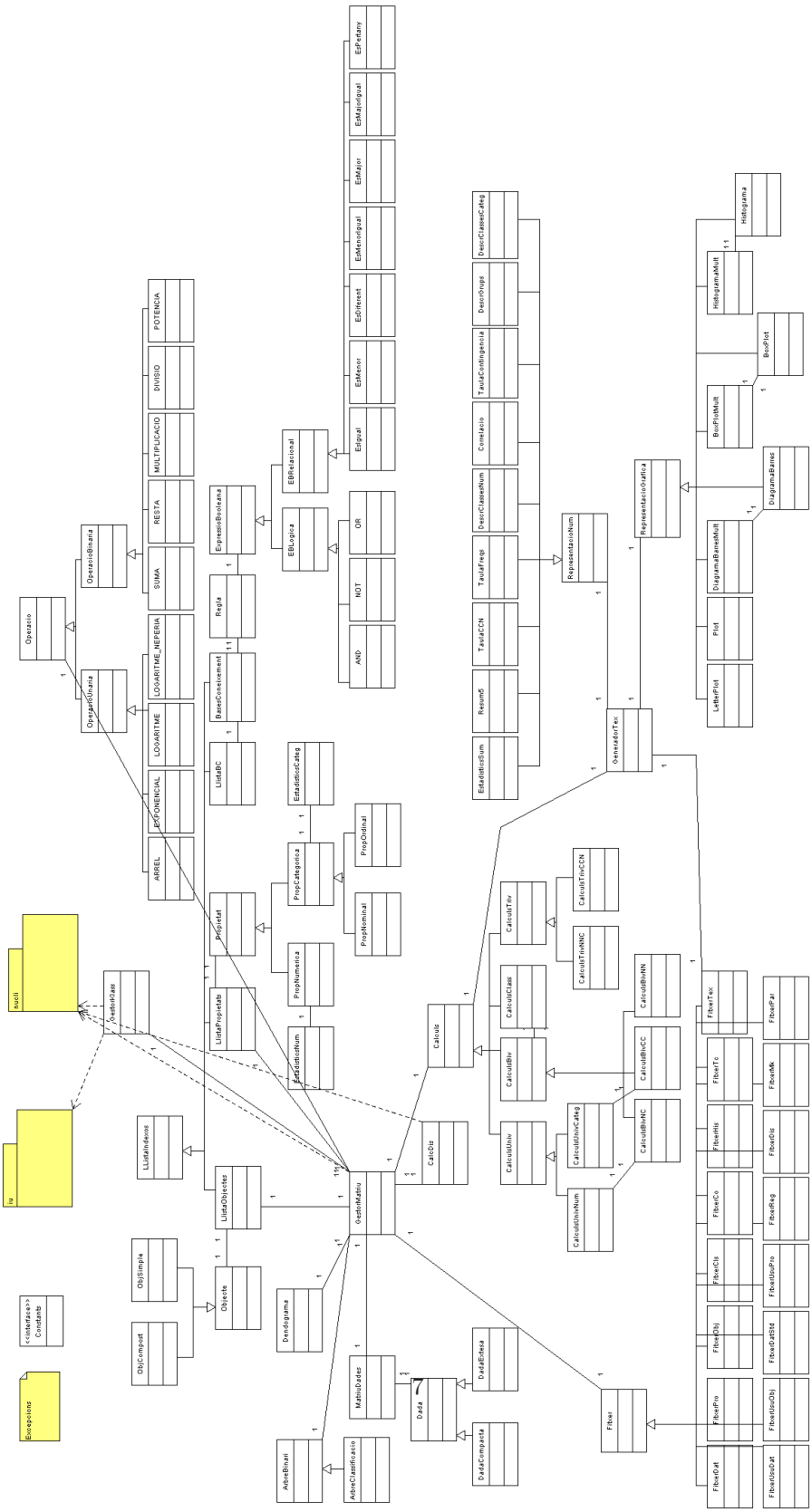


Figura 2.3: Diagrama de classes general de la capa de nucli

Capítol 3

Crear una nova funcionalitat

Per crear una nova funcionalitat dins del **Java-KLASS** s'han de fer tres passos:

- el primer és insertar la nova funcionalitat dins el menu principal que implica manipulacions a nivell d'interfície,
- després començar a implementar la propia funció a nivell de nucli
- i per últim utilitzar els *imports* de les classes existents per poder utilitzar funcionalitats ja creades per altres desenvolupadors.

3.1 Insertar una nova funcionalitat en el menu principal

La classe del menu principal es la *FrPrincipal.java*. Per crear una nova funcionalitat s'ha de possicionar en el lloc de les variables globals i crear:

- un *JMenu*, si la nova funcionalitat és un nou menú.
- un *JMenuItem* si aquesta nova funcionalitat pertany a un menú ja existent.

Una vegada s'ha creat la variable del menu s'ha d'instanciar dins la funció *jbInit()* com en l'exemple següent:

```
jMenuSelec.setEnabled(false);
jMenuSelec.setMnemonic('S');
jMenuSelec.setText("Selecciona submatriu");
jMenuSelec.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
    jMenuSelec_actionPerformed(e);
   }
});
```

Després simplement s'ha d'insertar dins el menu al que pertany:

- dins del menu *jMenuBar1* si es un nou menu, com en l'exemple:

```
jMenuBar1.add(jMenuDistancies);
jMenuDistancies.add(jMenuDdirecte);
jMenuDistancies.add(jMenuDmatriu);
```

- dins del menu *JMenu* al que pertany.

La majoria de les funcionalitats necessiten una matriu carregada, sense una matriu carregada no té sentit utilitzarles, fins i tot es podrien donar errors, per això en un principi estan deshabilitades fins que es carrega una matriu de dades, i després es posen en actiu dins la funció *habilitarOpcionsMenu*; és aquí on també s'ha d'habilitar la nova funcionalitat, l'exemple mostra com es fa.

```
public void habilitarOpcionsMenu(boolean habilita, boolean arbres){
  jMenuSelec.setEnabled(habilita);
```

Per últim només falta construïr la funció que crearà el nou menú al posar-lo dins el menú principal, és a dir dins del *FrPrincipal.java*; l'estructura és molt simple i continuant amb l'exemple de la selecció de submatriu es mostra el codi necessari:

```
void jMenuSelec_actionPerformed(ActionEvent e) {
  actualitzarBarraEstat(" ", false);
  contentPane.remove(panelCentral);
  panelCentral = new PanelSelec(this,gestor);
  contentPane.add(panelCentral,  BorderLayout.CENTER);
  validate();
  repaint();
}
```

3.2 Primers passos per implementar la nova funcionalitat

Una vegada s'han fet els passos de la secció anterior, és el moment de començar a crear la nova funcionalitat. Per continuar amb el disseny de tots els desenvolupadors dividirem la nova funcionalitat en dos parts, l'esquelet i la vista.

3.2.1 Creació de l'esquelet

En la primera línea de codi indicarem a quina de les carpetes de clases va la nova classe. Hi ha tres carpetes ben diferenciades:

- IU: Dintre d'aquesta carpeta es troben les classes on l'usuari interacciona amb el sistema, selecciona dades, introdueix valors....etc, és a dir són les classes de la Interficie d'Usuari.
- NUCLI: Dintre d'aquesta carpeta es troben les classes on l'usuari no interacciona amb el sistema, és a dir són les classes que processen, fan càlculs, obren i guarden arxius...etc.
- UTIL: Dintre d'aquesta carpeta es troben les classes que tenen variables estàtiques o configuracions segons el sistema operatiu. Son les classes que per dir-ho d'alguna manera no fan res directament sobre el **Java-KLASS** sino que serveixen per preparar-lo per fer-lo servir.

Quan ja s'ha decidit on anirà la nova classe depenen de les seves caracteristiques, es crea el constructor que crida a la funcio *jbInit()*, aquesta funció dibuixa el formulari de la nostra funcionalitat, és a dir la *vista*, això es mostra a la secció 3.2.3. Un exemple de com crear aquest constructor és el següent:

```
public PanelDmatriu(FrPrincipal fr,GestorKlass gk) {

  frPare = fr;
  gestor = gk;
  pDistan=new PanelD(frPare,opc,gestor,true);
  try {
    jbInit();
  }
  catch(Exception e) {
    e.printStackTrace();
  }
}
```

3.2.2 Creació de la Vista

Com s'ha esmentat en l'apartat anterior, la creació de la vista es fa en la funció *jbInit()*. La nova classe és una extensió d'un *JPanel* i s'ha d'anar amb compte amb aquesta classe, ja que en el moment que es fa un *this.add(....)* depenent de la versió de *JAVA*, pot ser que no es pugui veure el que s'ha construït, per això

es recomana possar *this.getContentPane().add(....)* que soluciona aquest petit problema.

3.2.3 Vistas multilenguaje

Utilizamos dos clases: ResourceBundle y Locale. La primera clase representa un recurso externo, es decir, un archivo de propiedades. La segunda, identifica un lenguaje y un país en particular, su constructor recibe 2 parámetros: el primero es un String que indica el lenguaje, y el segundo, es otro string que indica el país. Creamos un objeto del tipo Locale para representar el país y lengua a utilizar, y con este objeto, instanciamos un ResourceBundle, que nos servirá para obtener del archivo de propiedades, cada una de las etiquetas respectivas.

Por tanto en la subcarpeta iu de la carpeta jklass existen tres ficheros de propiedades:

- Resources_ca
- Resources_en
- Resources_es

Debemos tener un fichero de recursos por cada uno de los idiomas que queremos soportar en nuestra aplicación, en nuestro caso catalán (el idioma por defecto) inglés y español respectivamente. Los ficheros de propiedades contienen parejas clave/valor. Los valores consisten en texto traducido que nuestra interface mostrará. Especificaremos las claves cuando querramos utilizar los mensajes traducidos del ResourceBundle con el método getString. Por ejemplo, para recuperar el mensaje identificado con la clave 'Arxiu' del primer item de la barra de menú llamamos a getSting de este modo:

jMenuFile.setText(ResourceBundle.getBundle("jklass.iu.Resources").getString("Arxiu"))

Capítol 4

Compilació i execució

Una vegada ja s'ha creat la nova classe, amb les seves vistes i insertada dins el menú principal, és l'hora de compilar-ho tot i executar-ho i per això *(en entorn Windows)* són necessaris els següents elements:

- Path: Per poder utilitzar el compilador i el linkador de JAVA s'ha de completar la variable d'entorn PATH amb la ruta on es te instalat JAVA, per defecte es mostra la forma seguent:

  `PATH=\j2sdk1.4.2_07`.

- Compilar: Per compilar la nova classe s'ha d'escriure en una finestra de MSDOS

  `javac -d novaclasse.java`.

- Ejecutar: Per ejecutar una vegada s'ha compilat la classe s'ha d'escriure en una finestra de MSDOS

  `java -Djava.util.logging.config.file=".\conf\logging.properties"`

  `jklass.iu.AplJavaKlass`.

Es clar, que es pot construir un arxiu amb les dues línees de compilació i ejecució, i *llançar-ho* tot amb un .bat.

Una vegada s'ejecuta l'aplicació a part del menú de **Java-KLASS** també hi ha una finestra d'informació que està en segon pla. Aquesta finestra pot indicar totes i cada una de les acciones que fa l'aplicació o només les més rellevants o cap. Per

controlar aquesta informacio es fa servir l'archiu *logging.properties* de la carpeta *conf*. La quantitat d'informació ve determinada per la variable *.level* de la següent manera:

- .level= INFO: Indica que només mostrar l'informació més rellevant de les accions de **Java-KLASS**, es a dir si s'ha realitzat l'operació o no.
- .level= ALL: Indica mostrar tota l'informació de les accions de **Java-KLASS**.
- .level= FINER: Indica mostrar absolutament tot, és a dir cada moviment de ratoli, cada cop que estem sobre un menú executable..etc.
- eliminant .level: No s'inidica cap acció.

Capítol 5

Generacio de documentació

La major part de la documentació, inclús aquest manual, s'ha creat utilitzant LaTeX ja que a més de ser un software d'edició i creació de documents científics que permet una facil inserció de formules i taules, és gratuït.

5.1 Inserció d'imatges

És important remarcar que encara que es molt util, LaTeX té algun error per compilar alguns tipus d'imatges *(.bmp, .jpg ...)*, per això en tot el manual s'han fet servir com a imatges arxius *.eps* que no donden cap problema a l'hora de generar documents *pdf*.

Per fer servir aquestes imatges *.eps*, el que es proposa es crear la imatge en *.bmp*, ja que és molt facil de construir i després transformar aquesta imatge amb algun conversor d'imatges a *.eps*. En aquesta memoria s'ha fet servir una distribució d'evaluació de *Advanced Batch Converter* que ens deixa 30 dies per fer les conversions de *.bmp* a *.eps*.

Una vegada creat l'arxiu *.eps*, la imatge ocupa un foli A4 exactament, per eliminar marges blancs i ajustar el tamany de la imatge, s'edita el fitxer *.eps* amb qualsevol edior de text i es canvien els paràmetres de la linea *%%BoundingBox:* per que s'ajustin a la imatge. Com ajut indiquem que els primers 2 números fan referència a les coordenades x,y on comença la imatge i els 2 números següents indiquen les coordenades x,y finals.

5.2 Bilbiografia i referències creuades

Una part molt important en la memòria del projecte és la bibliografia, ja que aquesta memòria preten ser un document científic, fa referència a molts articles, llibres o publicacions dels quals s'ha tret informació per crear i anar modificant **Java-KLASS**.

Per crear la bibliografia s'ha optat per utilitzar els arxius *.bib* de LaTeX. Un arxiu .bib no es més que un arxiu en texte pla, que conté l'informació d'un article, llibre, publicació..etc al qual volem fer referència. En el següent exemple es mostra una de les entrades del *.bib* fet servir en el PFC.

@BOOK{abr,
author = {Abrams, R.},
title = {{Electroconvulsive Therapy}},
publisher = {Ed. Oxford University Press.},
year = {1997},
note = {Third Edition. NY.US.},
}

Per poder utilitzar l'arxiu *.bib* s'ha d'insertar al principi del document LaTeX la següent línea:

\bibliography{arxiubib1, arxiubib2, ...}

En LaTeX la bibliografia pot ser mostrada en molts estils diferents, en aquesta memoria i posteriorment en els següents PFC's s'utilitza el tipus *apalike* que mostra en cada referencia l'autor o autors i la data de publicació. Per fer que es mostri la bibliografia d'aquesta manera s'han d'insertar en el document principal de LaTeX les següents linees:

- \usepackage[apalike]{dk-bib}
- \bibliographystyle{apalike}

Per referenciar les cites bibliografiques en l'arxiu LaTeX només hem descriure \cite{abr} seguit l'exemple proposat avans.

Per referenciar imatges, taules, formules....etc només s'ha d'insertar a la taula o imatge o formula ...etc la linea \label{nom} on *nom* serà la referència que s'utilitzara dins del text posant \ref{nom}.

Per tal de que tant les referències creuades com la bibliografia surtin ben llistades, cal compilar el document LaTeX a *BibTex* i després en LaTeX un parell de cops.

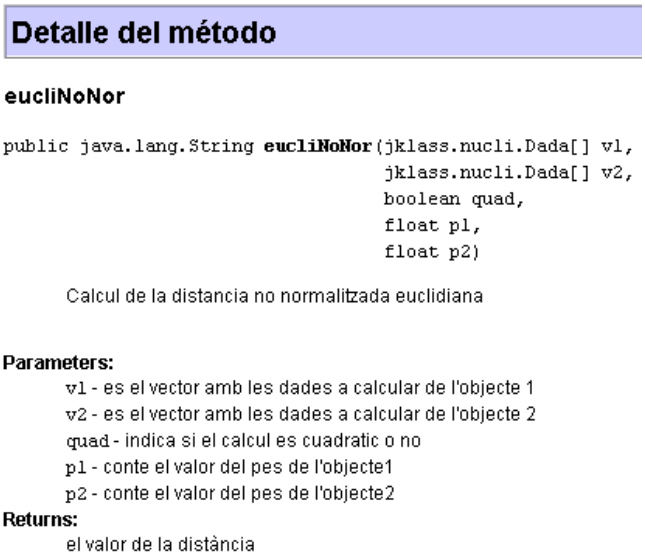
Capítol 6

Altre programari

L'altre petita part de documentació que no ha estat creada amb LaTeX s'ha construit amb el software següent:

- **Borland JBuilder 8:** Un entorn de desenvolpament que gracies a petites regles dins el codi, genera automaticament la documentació de les classes només *clicant* en l'apartat *doc* del programa. El següent ejemple mostra com aquests comentaris generen la documentació.

```
/**
* Calcul de la distancia no normalitzada euclidiana
*
* @param v1 es el vector amb les dades a calcular de l'objecte 1
* @param v2 es el vector amb les dades a calcular de l'objecte 2
* @param quad indica si el calcul es cuadratic o no
* @param p1 conte el valor del pes de l'objecte1
* @param p2 conte el valor del pes de l'objecte2
* @return el valor de la distància
*/
```



Detalle del método

eucliNoNor

```
public java.lang.String eucliNoNor(jklass.nucli.Dada[] v1,
                                   jklass.nucli.Dada[] v2,
                                   boolean quad,
                                   float p1,
                                   float p2)
```

Calcul de la distancia no normalitzada euclidiana

Parameters:

v1 - es el vector amb les dades a calcular de l'objecte 1
v2 - es el vector amb les dades a calcular de l'objecte 2
quad - indica si el calcul es cuadratic o no
p1 - conte el valor del pes de l'objecte1
p2 - conte el valor del pes de l'objecte2

Returns:

el valor de la distància

Aquesta informació proporcinada per **Borland JBuilder** és en format *html* i s'ha de transformar amb qualsevol dels programaris lliures a *.pdf* per poder-la insertar dins de la documentació.

- **pdftk-1.12:** Per insertar els *.pdf* s'utilitza aquest software lliure que serveix per fusionar diferents *.pdf* en un de sol. un exemple d'execució d'aquest programa es:

  pdftk.exe archivo1.pdf archivo2.pdf archivo3.pdf cat output archivofinal.pdf

- **Rational Rose:** Un entorn de modelatge visual bassat en llenguatge UML, que s'ha fet servir per crear:

  - Model de Casos d'ús
  - Model Conceptual
  - Diagrama de clases
  - Diagrama de la capa de presentació i nucli

  La notació UML afovereix molt la comprensió per part d'un usuari,que no enten res d'informàtica, del funcionament del programa i per un desenvolupador deixa molt clar com son les *caixes negres* del programa, és a dir, mostra com son les coses pero no de quina manera es fan.

Capítol 7

Generació d'una distribució de Java-KLASS

Una vegada la nova funcionalitat s'ha provat i perfeccionat és l´hora de passar als usuaris la nova versió de **Java-KLASS**.

Per crear una nova distribució es fan els següents passos:

- Compilar totes les classes, per obtenir l'arbre de directoris jklass complert.
- Crear el *.jar* a partir de las classes compiladas. Per fer aixo s'ha d'escriure en una finestra de MSDOS:

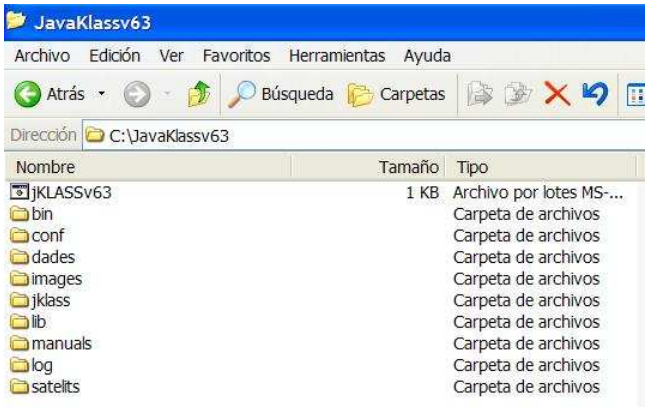
  `jar cvf .\lib\JavaKlass.jar -C .\ jklass`.

  i es crea el *fitxer.jar* en el directori *lib*.
- Ofuscar el *.jar*, es poden fer servir molts aplicatius, pero en aquesta versió s'ha utilitzat el *proguard3.8*. Per ofuscar el codi fent servir aquest programa s'han d'utilitzar els fitxers que es troben dins del subdirectori *lib*:
  - JavaKlass.pro.
  - proguard.jar.

  i s'ha d'escriure en una finestra de MSDOS:

  ```
  java -jar .\lib\proguard.jar @.\lib\JavaKlass.pro -verbose
  ```

- Preparar l'arbre de directoris i fitxers com en la figura seguent:



- Crear el *.zip* de l'arbre anterior amb nom *javaklassvX.jar*, on X es la versió de **Java-KLASS** i possar també un fitxer *README.txt*, que conté les modificacions més importants d'aquesta nova versió.

- La carpeta raíz de la nueva distribución de **Java-KLASS** de nombre *jklassvXgeneral* tendrá dos .zip: el de usuario y el de desarrollador que si se extraen crean dos subcarpetas: *jklassvX* la versión de usuario y *jklassvXdesenvolupador*con la estructura de carpetas presentada en el punto anterior. El ejecutable se llama *jklassvX.bat*.

- La carpeta Manuals en la versión de usuario tiene el pdf del manual de usuario y el pdf del manual de instalación, en la version de desenvolupador tiene tres subcarpetas: usuari, instalacio y desenvolupador. En cada subcarpeta fuentes latex, dvi y pdf del correspondiente manual.

- Tener la precaucion de asegurar que la distribucion de usuario no presenta trazas de la ejecucion en la ventana de background de java, lo que consume mucho tiempo de ejecucion innecesario. Para ello, asegurarse cada vez que se genera una version de usuario de que se ha modificado el nivel de loggin: ir a carpeta conf abrir el fichero logging.properties en ese archivo la segunda linea que no esta comentada es: .level = ALL cambiar ese ALL por SEVERE y se reducira el nivel de traza de la ejecucion.