Débuter par une installation Minimale

Ce polycopié est pour ceux qui ne sont pas habitués à installer des logiciels

Il est épuré un maximum pour vous aider à effectuer une installation minimale. Le but principal est que vous puissiez commencer et vous familiariser avec le fonctionnement de jfMindMap (jfMM), ses fichiers et ses dossiers.

Seulement 4 fichiers à 'downloader' pour commencer :

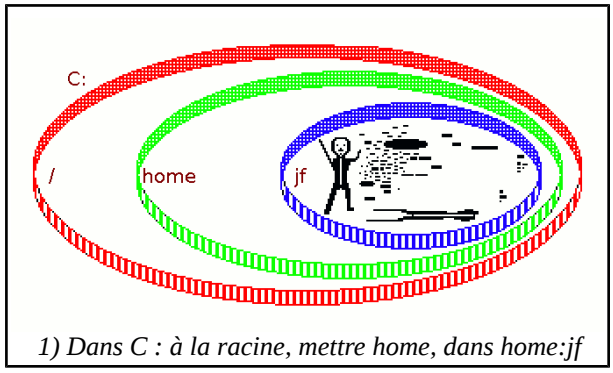
2 dossiers zippés : avant de les installer, il faudra les dézipper, décompresser

DossierDesClass.zip

Ce fichier est obtenu en zippant un dossier de 44 fichiers *.class

C'est-à-dire dont l'extension est '.class' (il finissent par les caractères '.class').

Ce sont des fichiers exécutables Java correspondant au code, au programme, au logiciel jfMindMap. Leur travail est de faire tourner sa machinerie (de jfMM).



1) Dans C : à la racine, mettre home, dans home:jf

Installation

Une fois dézippés, les 44 fichiers devront être 'installés'. Facile, il suffit de les copier dans C:/home/jf

Plus précisément : à la racine du disque dur, qui souvent est C : et parfois D :, vous devez créer un dossier nommé 'home', dans lequel vous créez un nouveau dossier nommé 'jf'. C'est dans ce dernier (jf), que vous copiez les 44 *.class.

Lancement :

Nous verrons ça plus tard, quand tout sera installé

§.zip : paragraphe.zip, il contient à la fois des images et du texte

Il contient une bonne soixantaine de fichiers : des fichiers image et des fichiers texte

Les fichiers images, surtout des *.png, servent à illustrer les cours. Donc ils ne sont pas exécutés, mais seulement affichés.

Les fichiers *.txt correspondants à chaque illustration, sert à la commenter, l'expliquer. Donc ils ne sont pas exécutés, mais seulement affichés.

Ces 2 types de fichiers, sont classés : il sont rangés dans une arborescence. Ils étaient organisés quand on les a zippés.

Installation

Il faut les dézipper pour les mettre au même niveau que dossierDesClass, c'est-à-dire dans C:/home/jf

Une fois dézippés, les fichiers retrouvent leur organisation initiale. Ils doivent être installés, c'est-à-dire dézippés dans C:/home/jf. Càd dans le dossier 'jf' qui se trouve dans home qui est à la racine.

Lancement :

Ces images et textes servent à illustrer le fichiers du cours 1. Ils sont affichés en cliquant dans le cours. Nous verrons ça plus tard, quand tout sera installé.

2 fichiers non zippés : donc inutile de les dézipper avant de les installer

Le fichier pdf : DébutAvecUneMiniInstallation_4.pdf,

C'est simplement celui que vous lisez actuellement. Vous le mettez où vous voulez. Le plus pratique serait de le copier dans C:/home/jf, mais ça n'a pas d'importance !

Le fichier du cours 1 que nous devons étudier :i1introBasicFm1PtDVuTrèGlobl_079.lsp

C'est simplement le fichier à consulter pour étudier le cours. Vous le mettez où vous voulez à condition de savoir où le retrouver, car vous devrez le charger dans jfMM pour l'étudier.

Le plus pratique serait de le copier dans C:/home/jf, mais c'est à vous de voir !

```
DébutAvecUneMiniInstallation_6.pdf
§
dossierDesClass
i1introBasicFm1PtDVuTrèGlobl_079.lsp
```

2)Pour une installation minimale, il vous suffit de 2 dossiers et de 2 fichiers

Voilà, tout est installé !

La solution la plus simple est de tout mettre dans C:/home/jf !

Finalement, dans C:/home/jf, on trouve tous les composants nécessaires : il sont seulement 4, et l'un d'eux est ce fichier pdf que vous lisez. Ce dernier explique comment installer les 3 premiers. Il n'y a que 3 composants à installer : un fichier et deux dossiers.

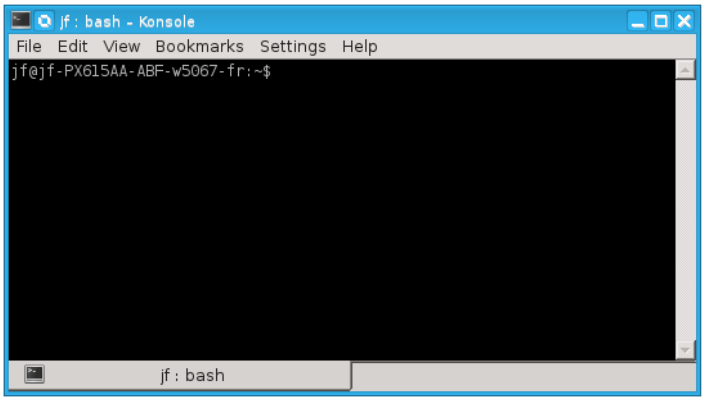
Lancement, il nous faut seulement lancer notre seul programme : jfMM

Le lancement se fait depuis une console, une invite de commande.

Décrire ce qu'on appelle 'console' (sous Linux), 'invite de commande' (sous Windows)

Le lancement de ce programme Java se fait à un niveau très profond, celui de l'operating system, c'est-à-dire au niveau du cœur, du noyau de l'ordinateur : on se trouve dessous l'interface graphique du système.

Donc une des caractéristiques de ce niveau est de ne pas utiliser le graphisme. Pas de bouton, menu, souris, d'environnement multi-fenêtré ; seulement une fenêtre blanche, le clavier pour saisir du texte : seulement des lettres, des chiffres, des mots. C'est ainsi qu'on écrit des commandes à l'intention de l'operating system.



3) la console de texte présente la référence de l'ordi et l'invite : '$'

Cet environnement austère comprend seulement une 'console' nommée "fenêtre en mode texte". Très épurée, elle affiche seulement les nom et référence du système et une invite (prompt en anglais) c'est-à-dire un caractère d'invite par lequel le système enjoint l'utilisateur à taper sa commande. Ici, le pilotage, la gestion du système, tout se fait au moyen de commandes uniquement rédigées en mode texte.

Comment lancer jfMM pour étudier le cours

Pour Linux, en mode console, c'est-à-dire en ligne de commande, il se lance :

```
jf@jf-PX615AA-ABF-w5067-fr:~$ pwd
/home/jf
jf@jf-PX615AA-ABF-w5067-fr:~$ ls
§145de0aCwinBad   coordonnées       dossierDesClass
§145de0àCwin.zip  cpyUrlUséBuf2.sh  Downloads
jf@jf-PX615AA-ABF-w5067-fr:~$ java dossierDesClass.JFMindMap
```

4) 'pwd' pour montrer que je suis dans '/home/jf' ; 'ls' montre les dossiers

Pour cela, dans le dossier où vous avez copié le dossier 'dossierDesClass' (on le voit en bleu, en haut à droite), après le caractère d'invite (ici, le dollar), il faut taper la commande :

java dossierDesClass.JFMindMap

Puis, en faisant 'entrée' on obtient un écran d'accueil qui ressemble à ceci :

Voici quelques explications qui vous aideront à mémoriser cette commande :

'java'

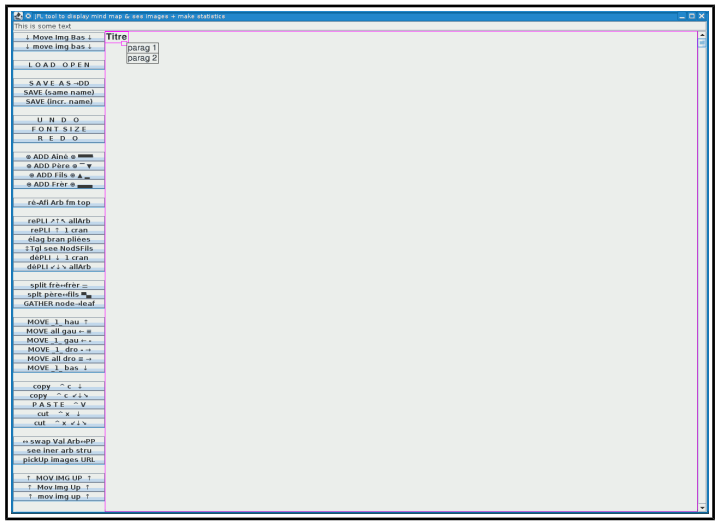
Pour lancer la JVM (Java Virtual Machine) : la machine virtuelle Java qui exécute le code compilé afin d'effectuer le travail.

"dossierDesClass.JFMindMap"

C'est le paramètre qu'on passe à Java, à la JVM. Il se décompose en deux parties :

'JFMindMap'

Car c'est le nom du programme principal du logiciel 'JFMindMap', qu'on demande à la JVM d'exécuter. J'ai obtenu le code compilé 'JFMindMap.class' en compilant le programme principal nommé "JFMindMap.java".



5) page d'accueil de jfMM, après le lancement

'dossierDesClass'

Car c'est le nom du dossier contenant JFMainPr.class et les autres *.class, c'est à dire les 40 autres programmes exécutables qui constituent le programme JFMindMap tournant sur la JVM (machine virtuelle Java).

Le point '.' qui sépare ces deux paramètres ?

En syntaxe Java il signifie que le fichier 'JFMindMap' est dans le dossier 'dossierDesClass'. C'est un peu l'équivalent du slash en syntaxe système.

Sous Window, en mode console, c'est-à-dire en ligne de commande, comment lancer jfMM ?

À la commande classique utilisée pour Linux : java dossierDesClass.JFMindMap il faut ajouter "*-Dfile.encoding=UTF-8*" l'instruction dédiée à Java pour forcer la console en mode 'UTF-8'. Au total la commande devient :

java "-Dfile.encoding=UTF-8" dossierDesClass.JFMindMap

à laquelle il faut ajouter 'entrée'.

C'est un peu compliqué à mémoriser par cœur mais on peut utiliser le copier-coller !

Comment utiliser jfMM pour étudier les *.lsp du cours

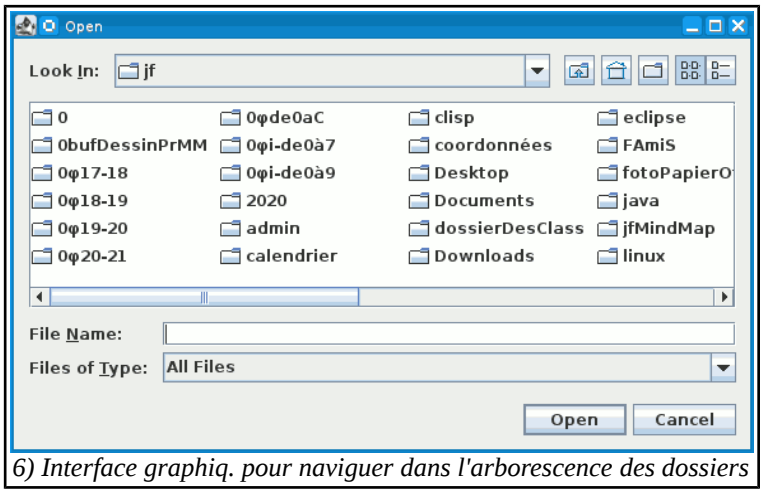
Utiliser jfMM pour afficher à l'écran le texte arborescent correspondant à un fichier *.lsp du cours

Quand on a lancé Java et qu'on obtient la page principale de jfMM (illustration 5) il nous faut maintenant aller chercher un fichier *.lsp, c'est-à-dire un des cours de 'φde1aN' / 'phiDe1aN' donc c'est à vous de vous souvenir où vous l'avez copié sur le disque dur pour aller le chercher.

Dans la colonne de gauche de la page principale, cliquer le bouton 'LOAD OPEN' affiche une interface graphique qui permet de naviguer sur le disque dur jusqu'à un fichier 'lsp', afin de l'ouvrir à l'écran.

Les conventions définissant les gestes de navigation sont classiques et je vous laisse les découvrir vous-même.

Après une navigation au travers des dossiers, vous arrivez en bout de branche sur 1 ou plusieurs fichiers : sélectionnez-en un pour que son nom apparaisse dans 'File Name'. Un clic sur le bouton 'Open' l'ouvre et l'affiche à l'écran : on obtient la structure d'un texte indenté, Voir en page 1 l'illustration 1.



6) Interface graphiq. pour naviguer dans l'arborescence des dossiers

Dans cet exemple, presque chaque ligne contient un pointeur vers une illustration. Pour afficher à l'écran le dessin pointé par un lien, il suffit de cliquer sur lui et jfMM escamote le texte pour le remplacer par l'illustration sélectionnée. Ensuite pour quitter l'illustration et retourner au texte, il suffit de cliquer n'importe où sur l'écran.

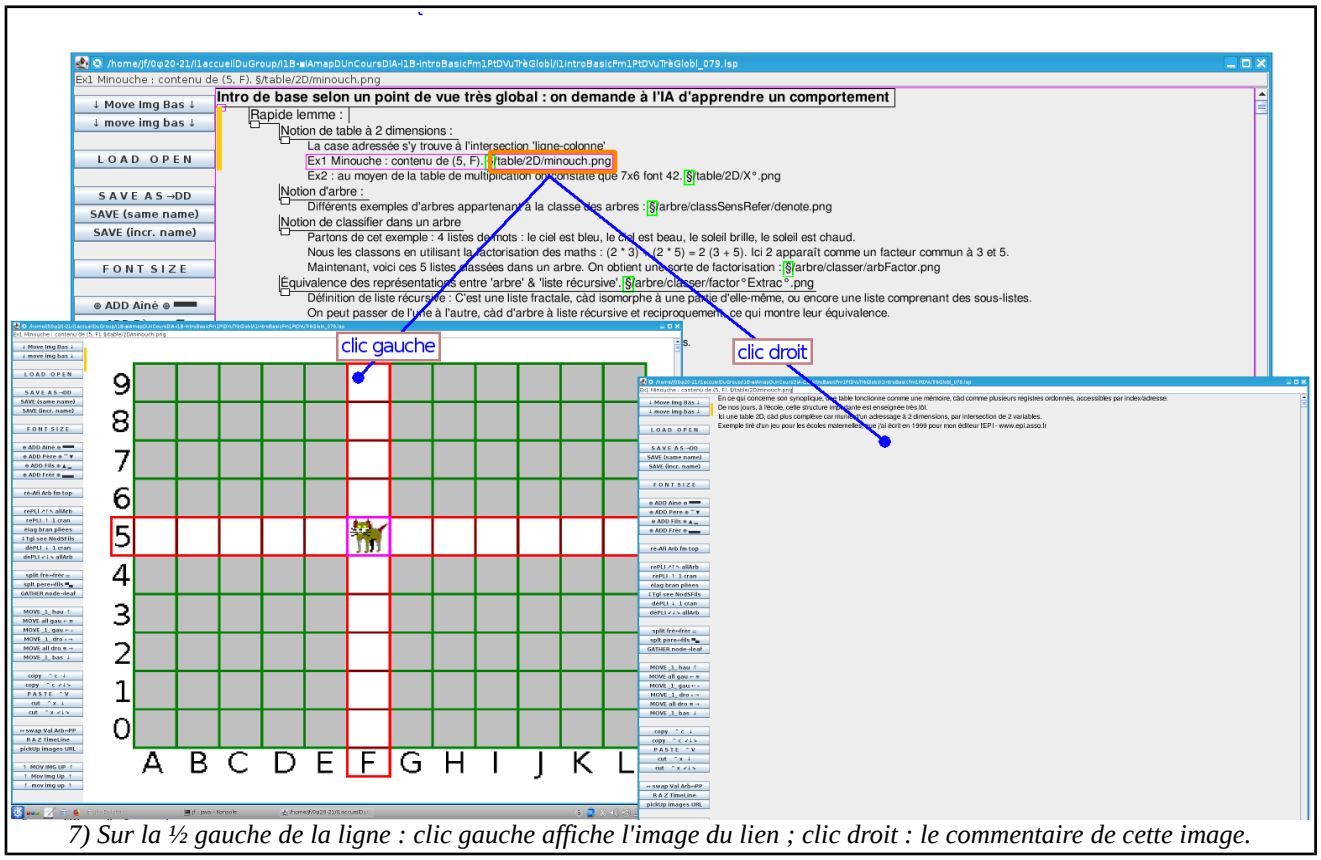
Plus précisément, l'affichage en fonction du lieu d'un clic sur la ligne agit ainsi (illustration 7) :

- Un clic sur sa moitié gauche la sélectionne : un cadre magenta est dessiné autour d'elle et son texte est recopié dans le cartouche d'édition, c'est-à-dire l'éditeur de ligne situé tout en haut de l'écran.

- Un clic sur sa moitié droite d'une ligne, si elle contient un lien vers un fichier du disque dur, produit l'affichage correspondante selon les conventions suivantes :

* un clic avec le bouton gauche de la souris, affiche l'image référencée par le lien.

* un clic avec le bouton droit de la souris, affiche le texte commentant cette image.



7) Sur la ½ gauche de la ligne : clic gauche affiche l'image du lien ; clic droit : le commentaire de cette image.

Attention : si une ligne est très courte et ne contient que quelques caractères en plus d'un nœud très long, cliquer sur le lien, par exemple sur son caractère '§' n'est pas suffisant et dysfonctionne. Il faut bien cliquer sur la partie droite de la ligne (c'est-à-dire à la droite de son milieu) !

Les boutons pour monter ou descendre le texte arborescent à l'écran

"↓ Move Img Bas ↓" bouge l'image vers le bas à grands pas
"↓ move img bas ↓" bouge l'image vers le bas à moyens pas
"↑ MOV IMG UP ↑" bouge l'image vers le haut à grands pas
"↑ Mov Img Up ↑" bouge l'image vers le haut à moyens pas
"↑ mov img up ↑" bouge l'image vers le haut à petits pas

Note :

Le tamponnage de l'image de l'arbre indenté

Sur jfMM dessiner la structure d'un texte indenté est lent. Donc pour accélérer cet affichage l'ordinateur dessine la structure dans une mémoire tampon qu'il utilise et ainsi affiche plus rapidement (tant que l'arbre affiché demeure inchangé).

Le tamponnage progressif de l'image

Alors, quand un texte est long, pour accélérer cette mémorisation, on génère seulement ses premières lignes et les lignes suivantes sont tronquées. Dans ce cas, une ligne horizontale rouge apparaît au niveau de la dernière ligne afin d'indiquer cette troncature.

Ré-affichage de l'arbre à partir du nœud initial

Si par la suite l'utilisateur fait monter l'image vers le haut, vient un moment où jfMM génère la partie suivante du texte. De cette façon, à mesure qu'on affiche l'arbre à l'écran, l'image est progressivement générée.

Mais pour générer l'arbre dans sa totalité, on peut cliquer sur le bouton : "ré-Afi Arb fm top", mais, puisque ça génère la totalité de l'arbre d'un seul coup, c'est plus long : il faut attendre plus longtemps.

Je sais bien que dans jfMM subsistent des bogues...

J'ai déjà réécrit jfMM trois fois. La dernière réécriture date du premier confinement et de l'été. Ce qui a induit beaucoup de bugs, que je corrige progressivement. En effet :

- Les bugs superficiels et bénins apparaissent rapidement et sont donc déjà corrigés.
- Puis apparaissent les bugs profonds et difficiles à corriger, mais ça se calme.
- Je n'ai pas daigné corriger certains bugs bénins. C'est ce à quoi je vais m'atteler maintenant... quand j'aurai le temps pour ça !

Au lieu d'afficher l'arbre entier, montrer à l'écran seulement les têtes de chapitres

Note : il faut d'abord cliquer sur un nœud pour le sélectionner. Quand c'est le cas :

Pour obtenir une approche plus synthétique d'un exposé, on peut souhaiter seulement voir les têtes de chapitres.

"rePLI ↗↑↖ allArb": replie toute la branche, c'est-à-dire toutes les branches du nœud sélectionné.
"rePLI ↑ 1 cran" : replie d'un cran les branches du nœud sélectionné. replie la dernière branche, la + fine
"dépLI ↓ 1 cran" : déplie d'1 cran les branches du nœud sélectionné. déplie la dernière branche, la + fine
"dépLI ↙↓↘ allArb": expanse toute la branche, c'est-à-dire toutes les branches du nœud sélectionné.

Note : Si on clique sur le petit rectangle juste au-dessous d'un nœud, ça inverse (toggle) son affichage : s'il était déplié ça le replie et réciproquement.

"↕Tgl see NodSFils" : inverse (toggle) l'affichage du nœud sélectionné : s'il était plié ça le déplie et réciproquement.

"élag bran pliées" : Attention Danger : réellement élague les branches pliées, c'est-à-dire les coupe définitivement. jfMM demande confirmation, mais ça reste dangereux.

Extrait du code source : le rôle de chaque bouton de jfMM

```
button20 = new JButton ("↓ Move Img Bas ↓");      button20.addActionListener(this);this.add(button20); // scroll down the image à grands pas
button17 = new JButton ("↓ move img bas ↓");      button17.addActionListener(this);this.add(button17); // scroll down the image à moyens pas

button28 = new JButton ("L O A D  O P E N");      button28.addActionListener(this);this.add(button28); // Ouvrir un file *.lsp et charger en mémoire

button13 = new JButton ("S A V E  A S →DD");      button13.addActionListener(this);this.add(button13); // Surf in le DD pr y sauv l'arb édité à 1 adresse existante
button32 = new JButton ("SAVE (same name)");      button32.addActionListener(this);this.add(button32); // sauve l'arbre que l'on a édité, sous le même nom
button33 = new JButton ("SAVE (incr. name)");     button33.addActionListener(this);this.add(button33); // sau l'arbr & incr son nom. si syntax= *_02.lsp → *_03.lsp

button34 = new JButton (" U  N  D  O ");          button34.addActionListener(this);this.add(button34); // ⮰ pour défaire la précédente instruct°, Mind : marche MAL
button39 = new JButton ("F O N T  S I Z E");      button39.addActionListener(this);this.add(button39); // set initFontSize vu val du Neu in PP
button35 = new JButton (" R  E  D  O ");          button35.addActionListener(this);this.add(button35); // ⮱ pour refaire la précédente undone ins, Mind : marche MAL

button03 = new JButton ("⊕ ADD Aîné ⊕ ▬▬");button03.addActionListener(this);this.add(button03); // ajouter un node frère au-dessus
button04 = new JButton ("⊕ ADD Père ⊕ ▯ ▼");      button04.addActionListener(this);this.add(button04); // ajouter un node fils au dessus
button02 = new JButton ("⊕ ADD Fils ⊕ ▲ ▯");      button02.addActionListener(this);this.add(button02); // ajouter un node fils au-dessous
button01 = new JButton ("⊕ ADD Frèr ⊕ ▬▬");button01.addActionListener(this);this.add(button01); // ajouter un node frère au-dessous

button24 = new JButton ("ré-Afi Arb fm top");     button24.addActionListener(this);this.add(button24); // ré-affiche l'arbre depuis la lig 0

button16 = new JButton ("rePLI ↗↑▯ allArb");      button16.addActionListener(this);this.add(button16); // fold tout et profond dans le sens de lourd
button23 = new JButton ("rePLI ↑ 1 cran");        button23.addActionListener(this);this.add(button23); // toggle pliage/affichage du sélected node
button27 = new JButton ("élag bran pliées");      button27.addActionListener(this);this.add(button27); // élague les branches qui sont pliées
button14 = new JButton ("↕Tgl see NodSFils");     button14.addActionListener(this);this.add(button14); // toggle pliage/affichage du sélected node
button22 = new JButton ("dépLI ↓ 1 cran");        button22.addActionListener(this);this.add(button22); // toggle pliage/affichage du sélected node
button15 = new JButton ("dépLI ▯↓↘ allArb");      button15.addActionListener(this);this.add(button15); // toggle pliage/affichage du sélected node

button25 = new JButton ("split frè↔frèr ▯");      button25.addActionListener(this);this.add(button25); // split ce Neu entre lui&1frèr direct à créer
button31 = new JButton ("splt père↔fils ▀▄");     button31.addActionListener(this);this.add(button31); // split ce Neu entre lui&1frèr direct à créer
button38 = new JButton ("GATHER node→leaf");button38.addActionListener(this);this.add(button38); // rassemble les branches de ce Neu→feuille

button07 = new JButton ("MOVE _1_ hau ↑ ");       button07.addActionListener(this);this.add(button07); // monter un node
```

```
button19 = new JButton ("MOVE all gau ← ≡");      button19.addActionListener(this);this.add(button19); // promo la fratry, càd lui et ts ls pti Frèr
button08 = new JButton ("MOVE _1_ gau ← -");button08.addActionListener(this);this.add(button08); // promouvoir only ce node
button06 = new JButton ("MOVE _1_ dro - →");button06.addActionListener(this);this.add(button06); // dégrader un node
button26 = new JButton ("MOVE all dro ≡ →");      button26.addActionListener(this);this.add(button26); // dégrader un node
button05 = new JButton ("MOVE _1_ bas ↓ ");       button05.addActionListener(this);this.add(button05); // descendre un node

button10 = new JButton ("copy   ^ c  ↓ ");        button10.addActionListener(this);this.add(button10); // copy → an inner PresPapier l'arb+fils but pas ses frères
button36 = new JButton ("copy   ^ c ▯↓↘");        button36.addActionListener(this);this.add(button36); // copy → an inner PresPapier l'arb+fils ET ses frères
button12 = new JButton ("P A S T E   ^ V");       button12.addActionListener(this);this.add(button12); // colle from inner PresPapier l'arbr qu'on vient d'y mettre
button11 = new JButton (" cut   ^ x  ↓ ");        button11.addActionListener(this);this.add(button11); // coupe / rub l'arb cliqué + fils but pas ses frères
button37 = new JButton (" cut   ^ x ▯↓↘");        button37.addActionListener(this);this.add(button37); // coupe / rub l'arb cliqué + ses fils ET ses frères

button29 = new JButton ("↔ swap Val Arb↔PP");button29.addActionListener(this);this.add(button29); // en intern. invert les val entre l'arb et le presse-papier
button30 = new JButton ("see iner arb stru");     button30.addActionListener(this);this.add(button30); // see inner mainNode struct sur la trace sur la console
button40 = new JButton ("pickUp images URL");button40.addActionListener(this);this.add(button40); // pickUp ls imag's URL used in la présenta°→cpyUrlUséBuf.sh

button09 = new JButton ("↑  MOV IMG UP  ↑");      button09.addActionListener(this);this.add(button09); // scroll up the image à grands pas
button21 = new JButton ("↑  Mov Img Up  ↑");      button21.addActionListener(this);this.add(button21); // scroll up the image à moyens pas
button18 = new JButton ("↑  mov img up  ↑");      button18.addActionListener(this);this.add(button18); // scroll up the image à petits pas
```