RAPPELS JAVA

Caractéristique du langage

- Indépendance de la plate-forme
- Fait pour le réseau: applets, servlets, libraries réseau , sgbd, etc
- Langage simple basé sur le C++ sans les pointeurs.
- beaucoup de librairies (packages)
- Strictement typé ("typed")
- Fait par Oracle(Sun il fut un temps !) (mais plusieurs autres implémentations existent)
- De plus en plus utilisé, surtout dans les systèmes embarqués ARM, Android, etc...

Vérifier la présence et la version du SDK java :

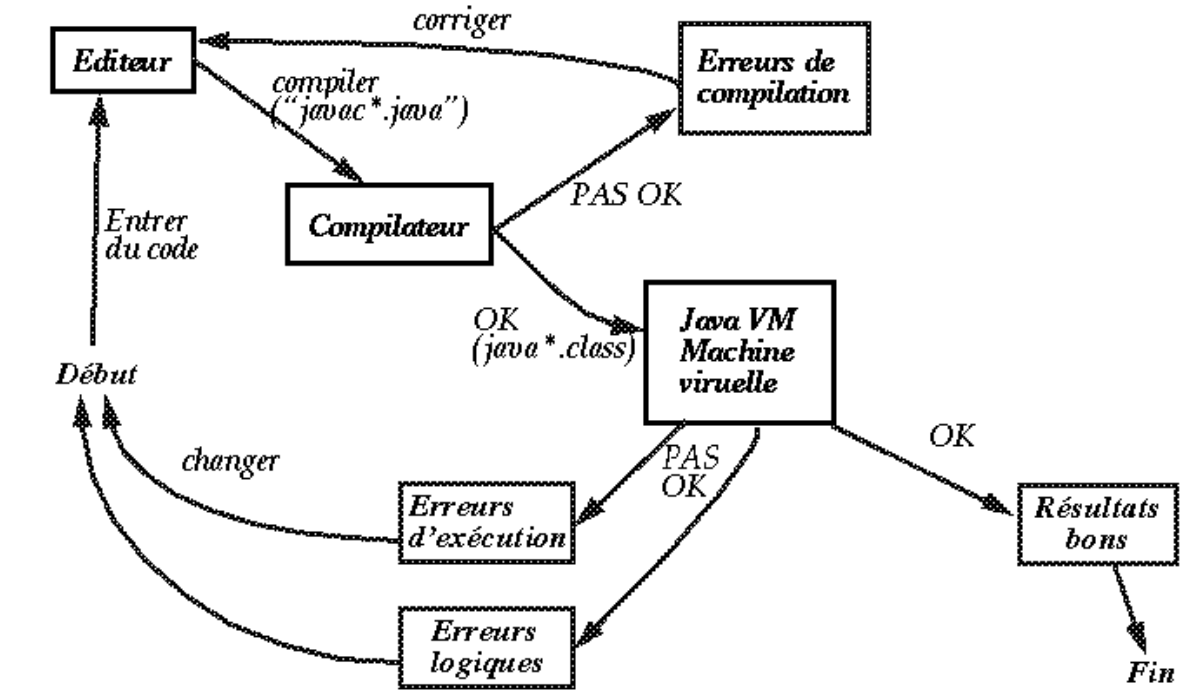
javac pour vérifier l'installation du SDK java, on obtient le texte suivant…

Usage: javac <options> <source files>
where possible options include:
-g Generate all debugging info
-g:none Generate no debugging info
-g:{lines,vars,source} Generate only some debugging info
…..

java -version pour verifier la version de la JVM… (Attention la JVM peut être indépendante du compilateur).

java version "1.7.0_40"
Java(TM) SE Runtime Environment (build 1.7.0_40-b43)
Java HotSpot(TM) 64-Bit Server VM (build 24.0-b56, mixed mode)

Le cycle de développement

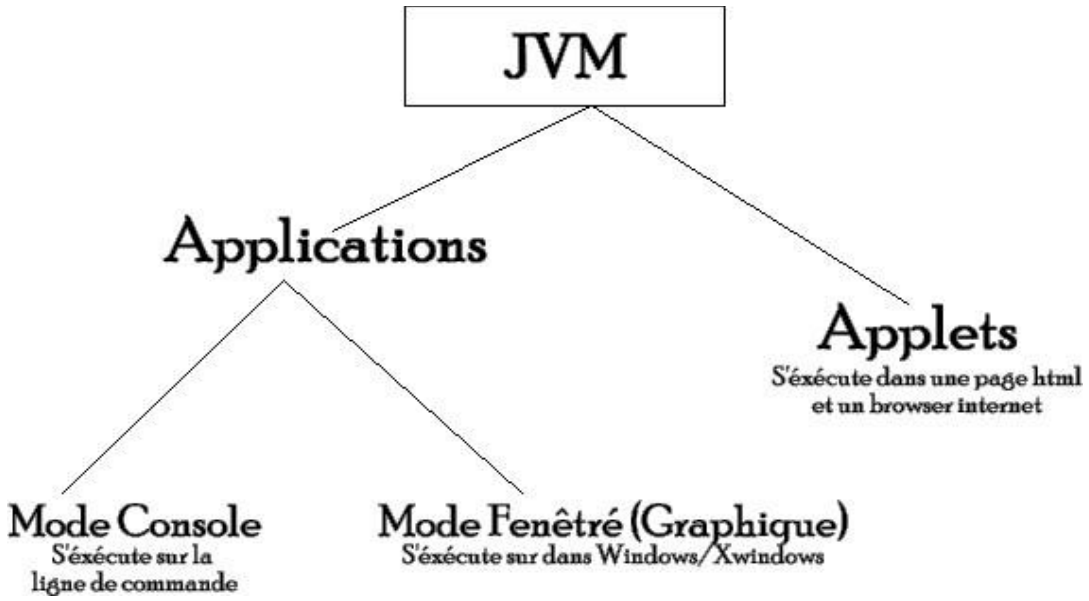


Compilation : javac nom_fichier_source.java

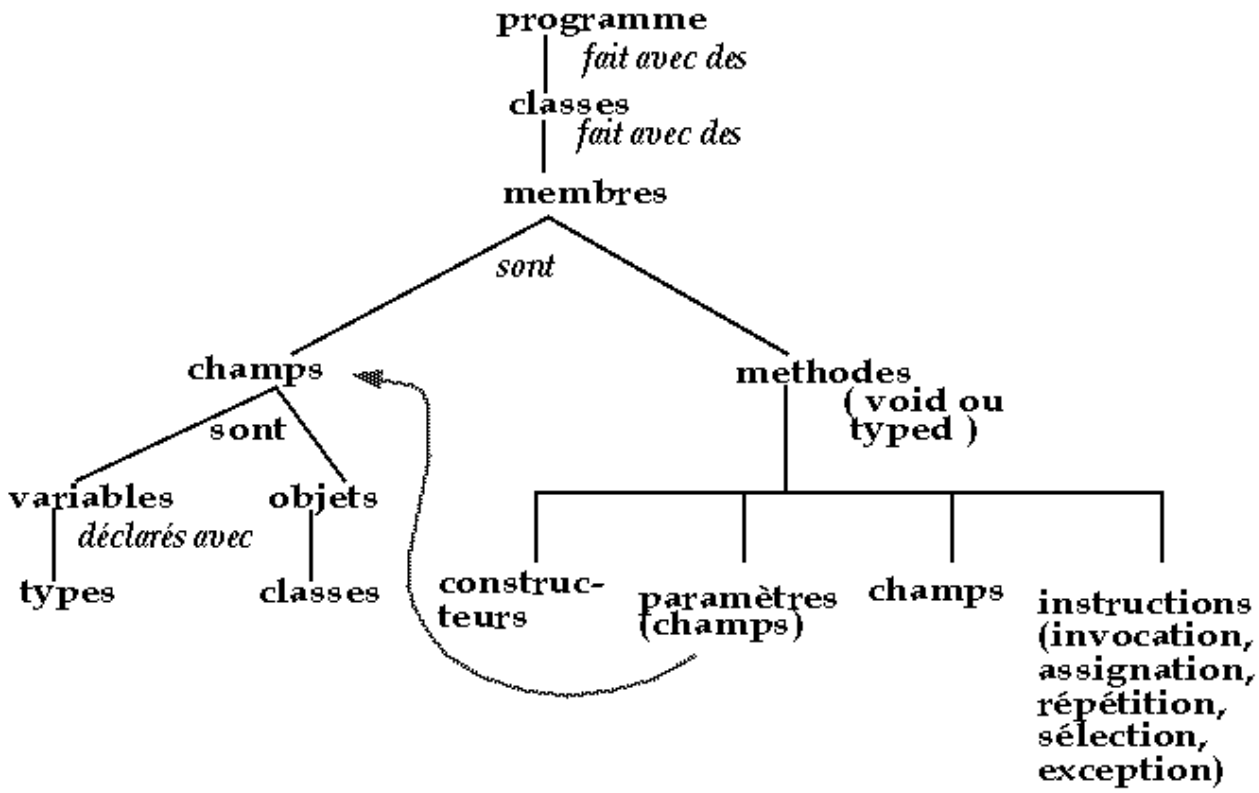
Exécution : java nom_fichier_**classe (attention ne pas mettre l'extension .class)**

Documentation : a posséder sous le coude...
http://java.sun.com/j2se/1.4.2/docs/api/
http://java.sun.com/j2se/1.5.0/docs/api/
http://java.sun.com/javase/6/docs/api/
http://docs.oracle.com/javase/7/docs/api/

Les types de programmes Java



La structure d'un programme JAVA



A retenir :

Nom de la classe principale = nom du fichier java
La classe principale doit contenir le point d'entrée du programme.

Point d'entrée en java :

public static void main (String argv[]){…..}

nb : toutes variables introduites dans une méthode static doit être static, sous peine d'un erreur de compilation.

Exemple de progamme simple :

```
public static void main(String[] args) {

System.out.println("Hello World!");

}//fin de la methode main
} //fin de la classe
```

Les types de bases :

Type	Désignation	Longueur	Valeurs
boolean	valeur logique : true ou false	1 bit	true ou false
byte	octet signé	8 bits	-128 à 127
short	entier court signé	16 bits	-32768 à 32767
char	caractère Unicode	16 bits	\u0000 à \uFFFF
int	entier signé	32 bits	-2147483648 à 2147483647

float	virgule flottante simple précision (IEEE754)	32 bits	1.401e-045 à 3.40282e+038
double	virgule flottante double précision (IEEE754)	64 bits	2.22507e-308 à 1.79769e+308
long	entier long	64 bits	-9223372036854775808 à 9223372036854775807

Cas particulier des chaînes de caractères :

Le type String n'est pas un type de base, mais une classe, de même on trouve sous la forme de classes les types de base (int => Integer, float => Float, etc, ils commencent tous par une majuscule.

Utilisation de tableaux :

Int T[] ;

T= new int [10] ; // dimensionne le tableau
T[0]=5
T[1]=10
[2]=3

Taille = T.lenght ; // donne la taille d'un tableau

Nb : Il est recommandé d'utiliser les classes conteneur pour gérer des suites d'éléments.

Les constructeurs :

Le contructeur d'une classe porte le même nom que la classe auquel il appartient, et ne possède pas de type de retour.

Exemple :
Class Animal {

Animal(){….}
Animal (String nom){…..}
}

Il peut y avoir plusieurs constructeurs pour une classe donnée.

Création d'un objet :

Animal unAnimal = new Animal (« chien ») ;

Nom_classe nom_objet=new nom_classe(paramètres) ;

Les méthodes :

Les méthodes possèdent toujours un type de retour .

Type_retour Nom_méthode (paramètres) {implémentation…}

Exemple : Void ManipulerA(int a,int b){…..}

Interface et classe abstraite :

Attention : Une Interface oblige à redéfinir toutes les méthodes.

Exemple la classe de gestion des événement de la souris :

Re-déclarer les 5 méthodes abstraites (obligatoirement) :

Public void mousePressed (MouseEvent e)
Public void mouseClicked (MouseEvent e)
Public void mouseReleased (MouseEvent e)
Public void mouseExited (MouseEvent e)
Public void mouseEntered (MouseEvent e)

Attention : Une classe abstraite doit être instanciée avant d'être utilisé.

Exemple la classe Calendar, nécessite d'être instancié avant utilisation avec une méthode spécifique :

Calendar c= Calendar.getInstance() ;

Clauses d'insertions :

On utilise le mots clé « import » équivalent du #include en C ou C++

Exemple : Import java.util.* ;

Héritage simple :

Class chien **extends** Animal {….}

Héritage multiple :

Uniquement avec une classe et plusieurs Interfaces

Exemple : class Chien **extends** Animal **implements** Mammiféres{…}

Récupération des arguments de la ligne de commande :

```
Public static void main (String argv[]){

If (argv.length>=1) {
        For(int i=0 ;i<argv.length ;i++) {
                System.out.println(« argument  » + i + « :  » +arv[i]) ;
        }
        }
else {System.out.println (« Pas d'arguments…. » ) ;}

}//fin du main
```

Introduction aux Threads :

Exemple de programme :

```
Public class test_thread implements Runnable {

Thread t ;

public static void main(String argv[]){
t=new Thread(this) ;
t.start() ;
}

public void run(){
```

```
while (t.isAlive()){
        try{t.sleep(1000) ;} catch(Exception e){}
        repaint() ;}
}
}//fin du run
```

Gestion du clavier :

Déclarer :
BufferedReader br ;

Créer l'objet :
Br=new BufferedReader(new InputStreamReader(System.in)) ;

Utilisation :
Try{nom=br.readLine() ;}catch(Exception e){}

```
import java.io.*;
class Greetings {

  public static void main (String [] args) throws IOException {
    BufferedReader in = new BufferedReader
        (new InputStreamReader(System.in));
    System.out.println("Quel est votre nom?");
    String name = in.readLine();
    System.out.println("Bonjour " + name);
  }
}
```

La classe scanner depuis la V1.5

```
Scanner sc = new Scanner(System.in);
 int i = sc.nextInt();
String str = sc.nextLine() ;
```

LES APPLETS

Pour définir une applet, il faut ajouter la close « import javax.swing.JApplet; ».

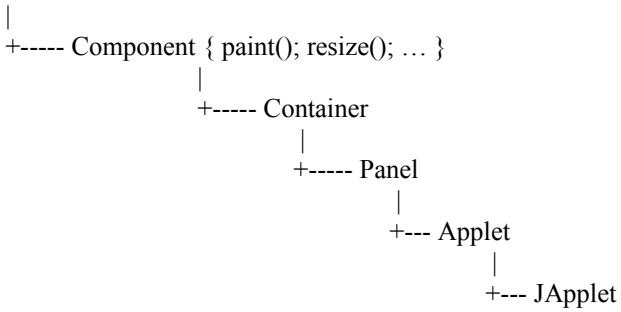
Puis étendre la classe JApplet :

Public class MonApplet extends JApplet implements MouseListener {.....}

Dans une applet, il existe ce que l'on appel le « le cycle de vie d'une applet », ce cycle correspond à un certains nombres de méthodes que l'on peut surcharger.

Ce cycle de vis correspond à un certains nombres d'états de notre applet.
• Les applets sont sous le contrôle du navigateur WWW (Firefox, IE, Hot Java, etc.)
• L'interface décide quand charger les applets d'une page HTML (=> état inactif)
• L'interface (re)démarre une applet quand sa fenêtre est visible sur l'écran (=> état actif)
• L'interface arrête l'applet quand elle disparaît de l'écran (=> état inactif)
• L'interface efface l'applet quand elle n'en a plus besoin.

La classe Applet



• Une applet est donc un *objet graphique*.
• Mais c'est aussi un *objet actif* créé et contrôlé par le navigateur Web

La classe Applet possède deux catégories de méthodes:

• les méthodes d'interface graphique (héritées):
public void paint(Graphics g);
public boolean mouseDown(Event evt, int x, int y);
public boolean action(Event evt, Object what);
etc.
etc.

• les méthodes de contrôle d'exécution
public void init();
public void start();
public void stop();
public void destroy();

Ces méthodes correspondent aux différents états de notre applet, et sont appelées lorsque nécessaire par notre JVM.

Au premier lancement de l'applet :

Public void init (){…..}

Au démarrage de l'applet :

Public void start(){….}

A l'arrêt de l'applet :

Public void stop(){….}

A la destruction de l'applet :

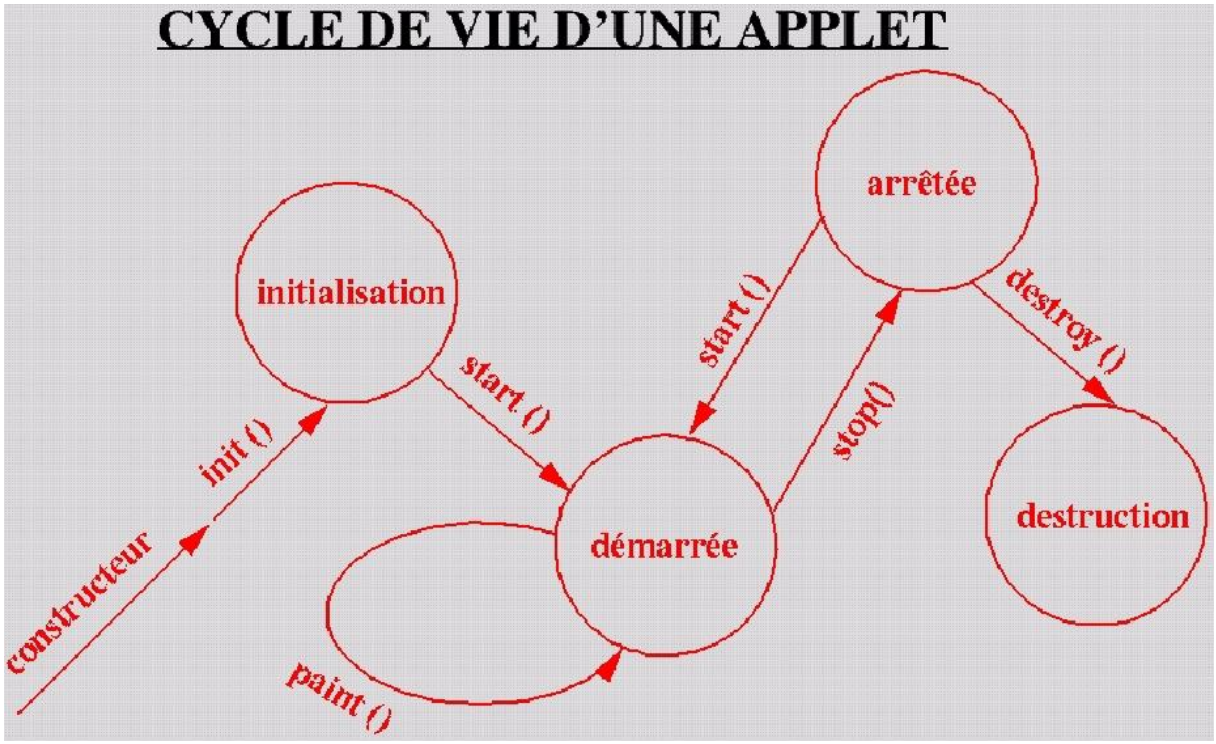
Public void destroy(){….}

A l'affichage de l'applet :

Public void paint(Graphics g){….}

La méthode repaint() permet de redessiner explicitement le « canvas » de l'applet, on appel pas directement la méthode paint().

Ce qui à été dit dans les pages précédentes est valable également avec les applets (gestion souris, etc…).



Nb : On remarquera que la méthode « paint » à pour paramètre un objet de la classe « Graphics », la classe « Graphics » apporte un ensemble de méthodes permettant de dessiner directement sur le « canvas de l'applet ».

En général, on utilise des méthodes de la classe Graphics en les appelant dans la méthode paint de l'applet.

- Dessiner et remplir : texte, lignes, rectangles, cercles, ellipses, polygones, arcs, images.
- Modifier : couleur, fonte, zone de dessin (« clipping area »)
- Mode de dessin : XOR avec une couleur donnée

Exemples de méthodes :

- g.setColor(bg);
- g.draw3DRect(0, 0, d.width - 1, d.height - 1, true);
- g.drawRoundRect(x, y, rectWidth, rectHeight, 10, 10);

Liste d'un certains nombres de ces méthodes :

void draw3DRect(int x, int y, int width, int height, boolean raised)

Draws a 3-D highlighted outline of the specified rectangle.

abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)

Draws the outline of a circular or elliptical arc covering the specified rectangle.

void drawBytes(byte[] data, int offset, int length, int x, int y)

Draws the text given by the specified byte array, using this graphics context's current font and color.

void drawChars(char[] data, int offset, int length, int x, int y)

Draws the text given by the specified character array, using this graphics context's current font and color.

abstract boolean drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer)

Draws as much of the specified image as is currently available.

abstract void drawLine(int x1, int y1, int x2, int y2)

Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.

abstract void drawOval(int x, int y, int width, int height)

Draws the outline of an oval.

abstract void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)

Draws a closed polygon defined by arrays of x and y coordinates.

void drawPolygon(Polygon p)

Draws the outline of a polygon defined by the specified Polygon object.

abstract void drawPolyline(int[] xPoints, int[] yPoints, int nPoints)

Draws a sequence of connected lines defined by arrays of x and y coordinates.

void drawRect(int x, int y, int width, int height)

Draws the outline of the specified rectangle.

abstract void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)

Draws an outlined round-cornered rectangle using this graphics context's current color.

abstract void drawString(AttributedCharacterIterator iterator, int x, int y)

Renders the text of the specified iterator applying its attributes in accordance with the specification of the TextAttribute class.

abstract void drawString(String str, int x, int y)

Draws the text given by the specified string, using this graphics context's current font and color.

void fill3DRect(int x, int y, int width, int height, boolean raised)

Paints a 3-D highlighted rectangle filled with the current color.

abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)

Fills a circular or elliptical arc covering the specified rectangle.

abstract void fillOval(int x, int y, int width, int height)

Fills an oval bounded by the specified rectangle with the current color.

abstract void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)

Fills a closed polygon defined by arrays of x and y coordinates.

void fillPolygon(Polygon p)

Fills the polygon defined by the specified Polygon object with the graphics context's current color.

abstract void fillRect(int x, int y, int width, int height)

Fills the specified rectangle.

abstract void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)

Fills the specified rounded corner rectangle with the current color.

abstract void setColor(Color c)

Sets this graphics context's current color to the specified color.

abstract void setFont(Font font)

Sets this graphics context's font to the specified font.

Une applet est un programme qui s'exécute sur dans page Web.

Nb : Commentaire à ajouter dans le fichier source java pour éviter de faire une page HTML, ce qui permet d'utiliser l'utilitaire appletviewer.exe pour exécuter notre applet.

Exemple : appletviewer monapplet.java

Pour cela il faut introduire en commentaire la ligne suivante :

//<applet code= »MonApplet.class » height=400 width=400></applet>

Il n'y a pas de fonction main dans une applet.

Le tag APPLET
But: inclure dans un document HTML un espace pour l'exécution d'une petite application.
<APPLET
[CODEBASE = *localisation_programme*]
CODE=*nom_fichier_programme*
WIDTH=*largeur_fenêtre*
HEIGHT=*hauteur_fenêtre*
autres>
<PARAM NAME=*nom1* VALUE=*valeur1*>
<PARAM NAME=*nom2* VALUE=*valeur2*>
...
</APPLET>
Le tag PARAM permet d'envoyer des paramètres (strings) à l'applet.

Transmettre un argument à une applet :

On transmet un argument à une applet par l'intermédiaire du code HTML :
< PARAM NAME = * VALUE = * >
< PARAM NAME = font VALUE = * >
Les * symbolisent les valeurs que vous devez spécifier, la première le nom du paramètre et la seconde sa valeur..

Récupérer un argument à partir d'une applet :

Pour récupérer à partir d'une applet les arguments transmis par le HTML, on utilise la méthode *getParameter()*.
TypeDeDonnée NomDeLaVariable = getParameter (" nomDuParamètre ") ;
String theFont = getParameter(" font ") ;

Gestion de la souris dans une applet

=> Implements MouseListener

Dans la méthode init pour une applet ou le constructeur sinon :
addMouseListener(this) ;

Re-déclarer les 5 méthodes abstraites (obligatoire) :

Public void mousePressed(MouseEvent e){….}
Public void mouseClicked(MouseEvent e){….}
Public void mouseReleased(MouseEvent e){….}
Public void mouseExited(MouseEvent e){….}
Public void mouseEntered(MouseEvent e){….}

AUTRES REMARQUES UTILES

Obtenir un nombre aléatoire :

Valeur = (int) (Math.random()* 900) ;

Convertir une chaîne de caractère en type entier :

String ValeurStr= « 10 » ;
Valeur = Integer.parseInt (ValeurStr) ;