Portée des variables :

Si une variable est déclarée en dehors de toute fonction, alors elle est globale.

Si une variable est déclarée à l'intérieur d'une fonction, y compris le programme principal main, alors sa portée est limitée à cette fonction.

Si une variable fait partie des arguments d'une fonction, alors sa portée est limitée à cette fonction.

Si deux variables de même nom sont déclarées dans deux fonctions différentes, alors elles représentent en réalité deux boîtes distinctes, et la portée de chacune est limitée à la fonction correspondante. Dans ce cas, elles peuvent même avoir des types différents.

Il faut éviter d'utiliser le même nom pour une variable globale et une variable locale. Cependant, si cela se produit, à l'intérieur de la portée de la variable locale, c'est celle-ci qui est visible et non plus la variable globale, qui ne sera à nouveau accessible que quand on sera sorti de la portée de la variable locale.

Par exemple, dans le programme suivant

 static int a;

 static void f (int x) {

 System.out.println(2 \* x);

 a = 2 \* x;}

 public static void main (String [] args) {

 int n;

 a = 3;

 n = 4;

 f(a + n);}

on exécute le programme principal dans un état qui contient la variable globale a



au cours de l'exécution du programme principal, on déclare une autre variable n, on affecte la variable a avec la valeur 3 et la variable n avec la valeur 4



au moment de l'appel f(5); de la fonction f, on supprime la boîte de nom n de l'état, car n est une variable locale au programme principal, mais on garde la boîte de nom a car a est une variable globale, et on ajoute une boîte de nom x, qui contient la valeur 7 de l'expression a + n



On exécute alors le corps de la fonction ce qui a pour effet d'afficher 14 et d'affecter la variable a avec la valeur 14.



puis, en quittant la fonction pour revenir au programme principal, on supprime la boîte de nom x et on remet la boîte de nom n avec le contenu qu'elle possédait avant l'appel de la fonction



PASSAGE DES ARGUMENTS

En java le passage des arguments lors de l’appel à une fonction se fait par valeur (pas pour tous les langages)

(Attention : dans le cas d’un objet comme un tableau ou une chaîne de caractères la valeur est une référence c’est à dire l’adresse du premier élément de l’objet : on peut parler de passage par référence mais c’est une question de vocabulaire)

Dans le programme suivant

 static int a,b;

 public static void main (String [] args) {

 int c;

 a = 4;

 b = 7;

 c = a;

 a = b;

 b = c;

 System.out.print(a);

 System.out.print(" ");

 System.out.println(b);}

l'exécution de l'instruction c = a; a = b; b = c; a pour effet d'échanger le contenu de la boîte a et de la boîte b. Le contenu de la boîte a étant 4 et celui de la boîte b 7, avant l'exécution de cette instruction, après l'exécution de cette instruction, le contenu de la boîte a est 7 et celui de la boîte b 4. Ainsi le programme affiche

7 4

Cette opération d'échange du contenu de deux boîtes étant souvent utilisée, on peut vouloir l'isoler dans une fonction.

 static int a,b;

 static void echange (int x, int y) {

 int z;

 z = x;

 x = y;

 y = z;}

 public static void main (String [] args) {

 int c;

 a = 4;

 b = 7;

 echange(a,b);

 System.out.print(a);

 System.out.print(" ");

 System.out.println(b);}}

Mais, ce programme, contrairement au précédent, affiche

4 7

et non

7 4

En effet, l'appel de la fonction echange(a,b); dans l'état



crée l'état



échange le contenu des boîtes x et y



puis les supprime



et le contenu des boîtes a et b n'a donc pas changé.

Il faut donc se souvenir que, dans un appel echange(e,e');, on ne passe jamais les expressions e et e' à la fonction echange, on ne passe que la valeur de ces expressions. Ces valeurs sont mises dans de nouvelles boîtes de nom x et y, qui n'existent que le temps de l'exécution du corps de la fonction. Quand les expressions e et e' sont des variables a et b, cela revient à recopier le contenu des boîtes de nom a et b dans les boîtes de nom x et y.