Développeur Web

Java

Date

SOMMAIRE

Présentation Générale

Historique

- La version 1.0 du langage Java apparaît en 1995 comme une solution aux limitations du html pour :
  - Créer des pages animées
  - Effectuer des controles de surfaces
- Initié par SUN dont la devise était « l'ordinateur c'est le réseau », Java en garde une inclinaison pour l'ouverture et les standards.
- Ses évolutions sont aujourd'hui pilotées par des processus communautaires ouverts.

But et positionnements

- À l’origine, concurrent de ActiveX et de Flash, Java a rapidement perdu la bataille du « client riche » (RIA) au profit de Flash

- Ce sont ses qualités intrasèques qui l’ont sauvé :
  - Un langage objet, fortement typé, simple d’accès,
  - Multi-plateformes,
  - Doté d’une API riche et également multi-plateformes

- C’est finalement coté serveur que Java a trouvé sa place avec des architectures élaborées pour la réalisation d’applications d’entreprise importantes.

Les différentes plateformes Java

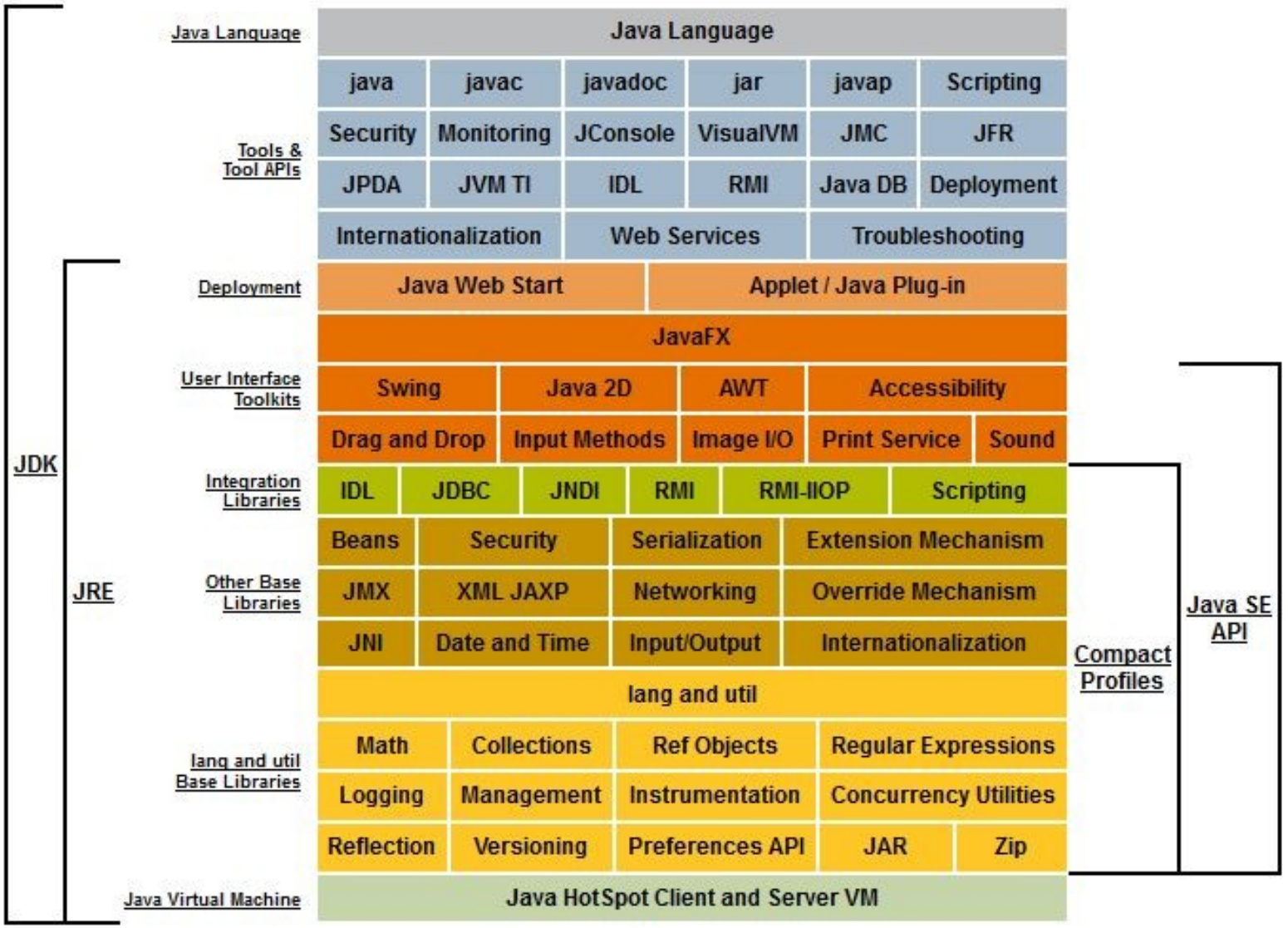
- Au delà du langage, le terme Java désigne l'ensemble des API qui constituent une plateforme de développement

- Il existe plusieurs plateformes ainsi définies, spécialisées dans le développement d'un type donné d'applications :
  - JSE (Java SE) : applications clients lourds, monopostes
  - JEE (Java EE) : applications serveurs ou réparties, clients légers ou riches
  - JME (Java ME) : applications pour l'informatique embarquée

JSE

- Il s’agit de la version « historique » de Java
- Elle comprend une riche bibliothèque d’utilitaires et d’algorithmes transversaux :
  - Structures de données telles que piles, files, tableaux dynamiques, …
  - Tris, manipulation de chaines de caractères, …
- Elle intègre également deux bibliothèques graphiques :
  - AWT : historique, liée à un rendu natif
  - Swing : rendu spécifique mais consistant entre plateformes, modèle MVC

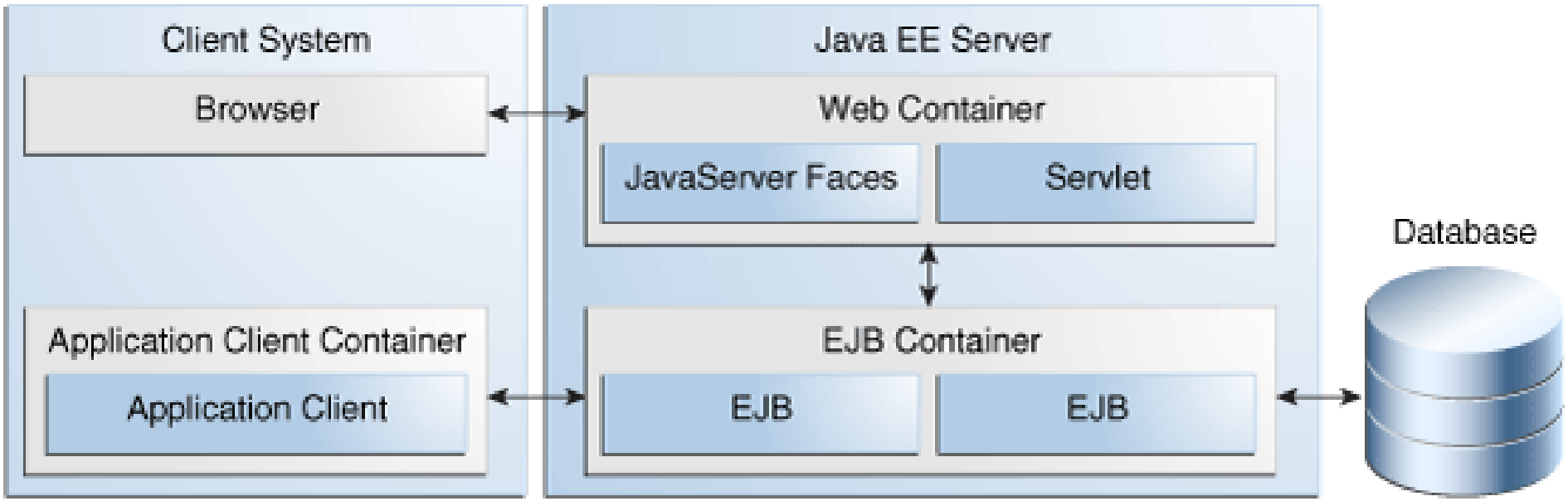
Java Standard Edition



JEE

- Il s’agit d’une version JSE enrichies des API de toutes les briques techniques serveurs dite JEE
- Ces API définissent en particulier les contrats de collaboration entre les composants métiers que vous développez et les environnements d’exécution dans lesquels ils seront déployés
- Elles apportent en sus des API pour la manipulation du XML ou l’utilisation de protocoles de communication tels
  - CORBA
  - SOAP
  - REST
  - …

Java Enterprise Edition



JME

- Une déclinaison de la plateforme Java destinée aux machines à ressources limitées pour :
  - l'informatique embarquée :
    - PDA (palm, …)
    - Téléphones portables
  - l'électronique grand public :
    - Télévisions,
    - Fours, aspirateurs,…
- Pour tous les matériels électroniques dotés de 128Ko de RAM ou plus et de processeurs aux fonctionnalités et aux performances limitées

Java en bref

- Java est un langage de programmation par objets créé par SUN Microsystems (James Gosling) (appartient aujourd'hui à Oracle).
- C'est un véritable langage objets qui supporte l'ensemble des mécanismes (classe, héritage, polymorphisme), portable et indépendant de toute plateforme.
- Le langage est un bon compromis entre le langage C++ (plus simple et syntaxe proche du C++) et smalltalk (gestion dynamique).
- Associé à trois plateformes d'exécution : JME, JSE et JEE, proposant un ensemble de librairies standard (entrées-sorties, graphique, accès base de données, middlewares pour applications Web et distribuées, ...)

Le langage Java est portable

- La taille des types primitifs est indépendante de la plateforme.
- Java supporte un code source écrit en Unicode.
- Le compilateur Java génère du byte code.
- La Java Virtual Machine (JMV) est présente sur Unix, Linux, Windows, Mac, OS/2, Android, ...

Architecture d’un projet JAVA

Architecture

- Ensemble de classes dont une « principale »
- Nom de classe homographe du fichier source (« .java »)
- Point d’entrée par la méthode « main() » de la classe principale

Classe minimale

Regardons le fichier source Demo.java

```
public class Demo {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Règle de codage

- Nom de classe commence par une majuscule
- Nom de méthode commence par une minuscule
- Fin d’instruction par un point-virgule
- Indentation
- CamelCase dans le nommage
- Pas d’accent dans les noms
- Attention aux noms réservés au système (merci l’IDE)

La syntaxe JAVA

Types primitifs

- Entier
  - int : entier de base (-2^{31} à $2^{31}-1$)
  - short : entier court (-32768 à 32767)
  - long : entier long (-2^{63} à $2^{63}-1$)
  - byte : octet (-128 à 127)
- Réel (ou décimal)
  - float : simple précision
  - double
- Caractère
  - char
- Booléen
  - boolean

Déclaration de variable

- Avant d'être utilisée, une variable doit être « déclarée », identifiée à un type de donnée particulier

```
char myChar ;
```

- Possibilité d'initialiser directement une valeur lors de la déclaration

```
int myInt = 17 ;
```

- 'final' permet de spécifier que la valeur ne pourra plus être modifiée par la suite

```
final double ratio = 1.1d ;
```

Les littéraux

- Entiers : 22, 010, 0xFF
- Réels : 3.1415f, 12e-3
- Caractères : ‘A’, ‘9’, ‘*’
- Chaines de caractères : "Ceci est une chaîne", "", "A"
- Représentation de caractères spéciaux :
  - Retour arrière ‘\b’
  - Tabulation horizontale ‘\t’
  - Retour à la ligne ‘\n’
  - Retour chariot ‘\r’

Opérateurs

- Opérateurs arithmétiques

  +, -, *, /, %

- Opérateurs relationnels

  >, <, >= , <=, ==, !=

- Affectation

  =

- Opérateurs logiques

  &&, ||, !

Opérateurs

- Incrémentation et Décrémentation
  - Incrémentation : ++
  - Décrémentation : --
- Opérateurs suffixés et préfixés

```
int res=0, i=3 ;
res = i++ ;       //res = 3, i = 4
res = ++i ;       //res = 5, i = 5
res += 2 ;        // res = 7
```

- Opérateur conditionnel (ternaire)
  - exp1 ? expr2 : expr3 ;

```
int x=2,  y=5, max ;
max = (x>y) ? x : y ;
```

Les entrées/sorties formatées

- La gestion des entées-sorties se fait par un ensemble de fonctions standard définies dans la librairie de base
- Java propose une classe System qui contient des flux standards prédéfinis (in, out) nous permettant de manipuler les entrées-sorties standards
- Exemple de sortie formatée sur la console :

```
public class Demo {

    public static void main(String[] args) {

        for (int i = 0; i < args.length; i++) {
            System.out.println("args[" + i + " ]=" + args[i]);
        }
    }
}
```

Les entrées/sorties formatées

- La classe Scaner (JDK1.5) permet la lecture d'information formatée

```
import java.util.Scanner;

public class Demo {

    public static void main(String[] args) {
        String message;
        Scanner keyboardInput = new Scanner(System.in);
        System.out.println("Entrez un message :");
        message = keyboardInput.next();
        int size = message.length();
        System.out.println("Taille du message '" + message + "' : " + size);
    }
}
```

- next() prendra seulement le premier mot, avec nextLine() on peut saisir une phrase complète
- La classe Scanner propose des méthodes de lecture formatée pour tous les types primitifs : nextInt, nextLong, nextFloat, nextDouble, nextBoolean, ...

Opérateurs conditionnels : if/else

- Ils permettent d'exécuter une instruction ou une suite d'instructions en fonction de l'évaluation d'une condition.

```
if (expression) {
    instruction1 ;
}
else {
    instruction2 ;
}
```

- Lorsqu'il n'y a qu'une seule instruction, la définition d'un bloc par les accolades n'est pas obligatoire. Cependant, il est **préférable de toujours en définir**.
- L'instruction « else » est facultative.

Opérateurs conditionnels : if/else

```
import java.util.Scanner;


public class Demo {

    public static void main(String[] args) {
        int x, y, result;
        Scanner keyboardInput = new Scanner(System.in);
        System.out.println("Enter an integer:");
        x = keyboardInput.nextInt();
        y = keyboardInput.nextInt();
        if (x > y) {
            result = x;
        } else {
            result = y;
        }
        System.out.println("Max is " + result);
    }
}
```

Opérateurs conditionnels : if/else if / else

- Elle permet de choisir une séquence d'instructions parmi plusieurs, en fonction de l'évaluation d'une expression.

```
if (expression1) {
    instruction1 ;
}
else if(expression2) {
    instruction2 ;
}
else if(expression3) {
    instruction3 ;
}
else {
    instruction ;
}
```

- Lorsqu'il n'y a qu'une seule instruction, la définition d'un bloc par les accolades n'est pas obligatoire. Cependant, il est préférable de toujours en définir.
- L'instruction « else » est facultative.

Opérateurs conditionnels : if/else if / else

```java
import java.util.Scanner;

public class Demo {

    public static void main(String[] args) {
        Scanner keyboardInput = new Scanner(System.in);
        System.out.println("Entrez un jour dans la semaine (1-7): ");
        int dayNb = keyboardInput.nextInt();
        if (dayNb == 1)
            System.out.println("Lundi");
        else if (dayNb == 2)
            System.out.println("Mardi");
//          ...
        else if (dayNb == 7)
            System.out.println("Dimanche");
        else
            System.out.println("Petit malin !");
    }
}
```

Opérateurs conditionnels : switch

- Elle permet de choisir un groupe précis d'instructions parmi plusieurs.

```
switch(expression) {
    case valeur1 :
        instruction1 ;
    case valeur2 :
        instruction2 ;
    case valeur3 :
        instruction3 ;
    default :
        instruction ;
}
```

Opérateurs conditionnels : switch

Multi-conditions avancé : switch

```
import java.util.Scanner;

public class Demo {

    public static void main(String[] args) {
        Scanner keyboardInput = new Scanner(System.in);
        System.out.println("Entrez un jour dans la semaine (1-7): ");
        int dayNb = keyboardInput.nextInt();
        switch (dayNb) {
            case 1:
                System.out.println("Lundi");
                break;
            case 2:
                System.out.println("Mardi");
                break;
//              ...
            case 7:
                System.out.println("Dimanche");
                break;
            default:
                System.out.println("Petit malin !");
        }
    }
}
```

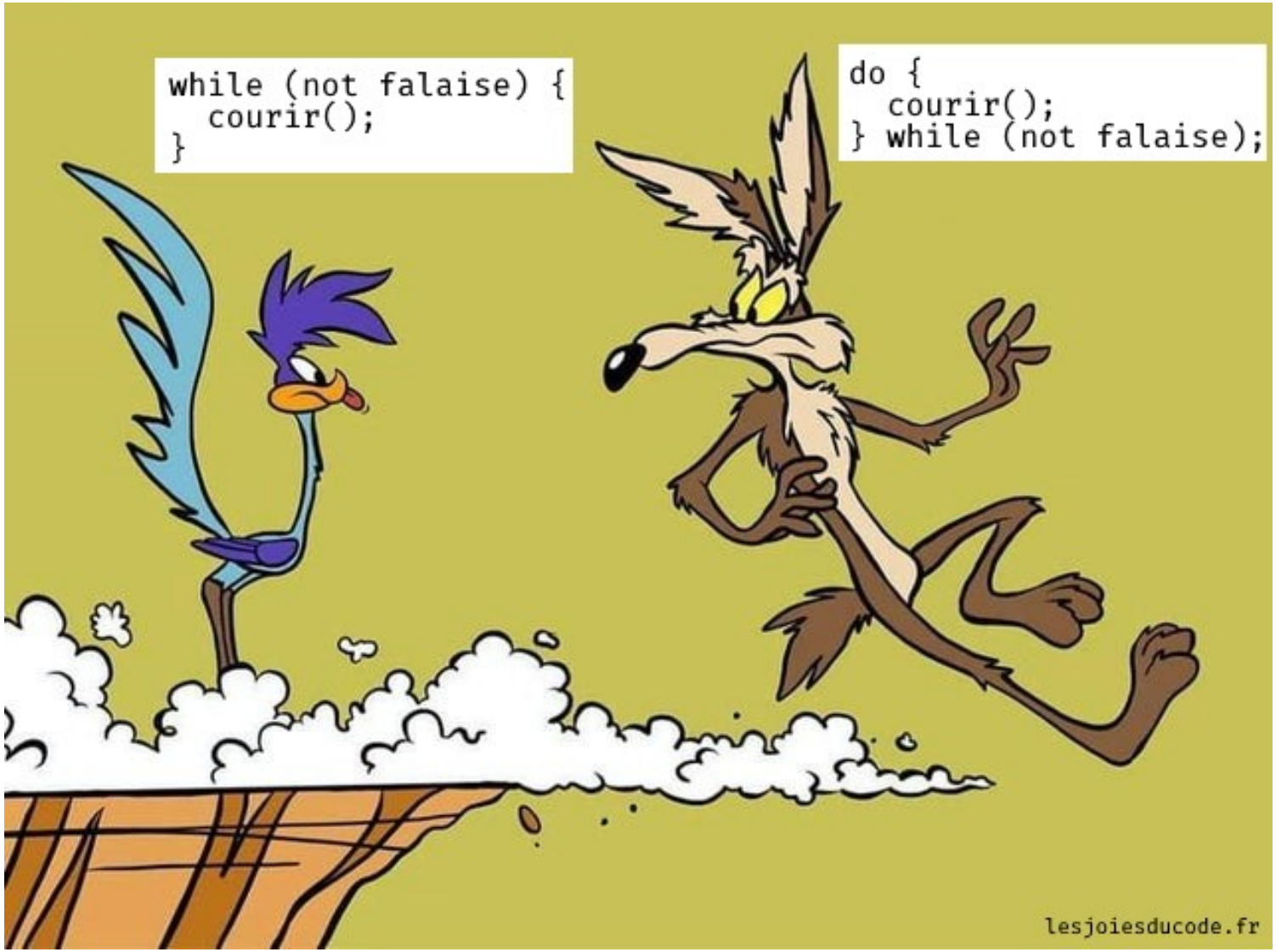
Opérateurs conditionnels : la boucle while

- Elle permet de répéter une séquence d'instructions tant qu'une condition est vérifiée.

```
public class Demo {

    public static void main(String[] args) {
        int nb = 0;
        while(nb++ < 10) {
            System.out.println(nb);
        }
    }
}
```

Opérateurs conditionnels : do … while … do



Opérateurs conditionnels : do … while

- Variante de while, permet d'exécuter une séquence d'instructions. Mais contrairement à while, elle test sa condition d'arrêt en fin de boucle.

```
public class Demo {

    public static void main(String[] args) {
        int nb = 0;
        do {
            System.out.println(nb);
        } while(nb++ < 10);
    }
}
```

Opérateurs conditionnels

- Faire une boucle avec une règle définie : for

```
public class Demo {

    public static void main(String[] args) {
        for(int i = 0; i < 10; i++) {
            System.out.println(i);
        }
    }
}
```

Tableaux

- Un tableau est un ensemble d'éléments de même type
- Mise en place grâce à l'opérateur [ ]

  int table[] ;
- Objet : création via l'opérateur « new »

  int table[] = new int[4] ;
- Initialisé à la création
  - 0 pour les entiers et décimaux
  - Null pour les objets
- Possibilité d'initialiser à la création (absence de « new »)

  int table[] = { 0, 5, 10, 15, 20 } ;

Manipulation de tableaux

- Toute opération se fait élément par élément
- Accès aux éléments par le biais d'index

```
int table[] = {2, 4, 6, 8, 10};
for (int i=0; i < table.length; i++) {
    System.out.println(table[i]);
}
```

- Evolution/Variante de l'utilisation de la boucle for : itération sur chaque élément du tableau

```
char table[] = {'I', 'S', 'I', 'S'};
for (char i : table) {
    System.out.println(i);
}
```

Manipulation de tableaux

- Attention, les opérations d'affectation et égalité sont applicables mais se font sur les références d'objets !!!!

```
int tabOne[] = {2, 4, 6, 8, 10};
int tabTwo[] = new int[3] ;
tabTwo = tabOne;
tabTwo[0] = 100 ;
System.out.printl( tabOne[0] + " VS " + tabTwo[0]);
```

Manipulation de tableaux

- Java fournit la classe « Arrays » permettant différentes opération sur les tableaux
- Copie

```
int tabOne[] = {2, 6, 10, 4, 8};
int tabTwo[] = Arrays.copyOf(tabOne, tabOne.length);
tabTwo[0] = 100 ;
System.out.println( tabOne[0] + "VS " + tabTwo[0]);
```

- Recherche

```
System.out.println("Position de 10 :" +
Arrays.binarySearch(tabOne, 10)) ;
```

Manipulation de tableaux

- Tri

```
Arrays.sort(tabOne) ;
System.out.println("Position de 10 :" +
Arrays.binarySearch(tabOne, 10)) ;
```

- Remplissage

```
Arrays.fill(tabOne, 0) ;
for(int i : tabOne)
    System.println(tabOne[i]) ;
```

- Etc...

Tableaux multidimensionnels

- Il est possible d'avoir des tableaux à plusieurs dimensions (matrices)

```
public class Demo {

    public static void main(String[] args) {
        int[][] matrice1 = new int[2][3];
        int[][] matrice2 = { {10, 20, 30},
                             {40, 50, 60} };

        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.println(matrice2[i][j]);
            }

        }
    }
}
```

TP

TP : Jour de Noël

- Calculer le jour de la semaine où tombe Noël pour une année, entre 2000 et 2099, saisie au clavier, sachant que :
  - En 2000, Noël était un lundi
  - D'une année sur l'autre, hors année bissextile, Noël progresse d'un jour
  - Pour les années bissextiles, la progression est de deux jours
  - 2000 était bissextile, ainsi que 2004 ...

Code Lutin

02 • 40 • 50 • 29 • 28

www.codelutin.com