Université
Internationale
de Casablanca

Développement mobile Android et E-commerce

+

+

Wilfried Y. H. Adoni

Université Internationale de Casablanca

Informations pratiques

- 46 heures de cours + TP
- Evaluation : 1 Contrôle, 1 Examen sur machine + Mini Projet en binôme

Université Internationale de Casablanca

Plan du cours

- **Partie I: Rappel des concepts de base et programmation orienté Objet**
  - Affectation, opérations, entrée/sortie, conditions, les itérations, les tableaux et les fonctions.
  - Les objets, les classes, les attributs de classe, les constructeurs, les méthodes de classes, l'héritage, les packages et les collections d'objets.

- **Partie II: Programmation Android**
  - Installation et configuration des outils, activités et vues, EDI entre activités
  - Menu, Menu Contextuel, Fragment, WebView, flux RSS, Audio Stream, et Video Stream
  - Google services (Google Map, Google Admob)
  - Base de données SQLite, MySQL, Google Firebase, Streaming
  - Android Swatch , Google TV

- **Partie III: E-commerce**
  - Introduction aux plateformes e-commerce, Structures mécanismes et impacts, Comportement consommateur, sondage online et CRM, Commerce collaborative, portails d'entreprise et chaine d'approvisionnement électronique, Sécurité e-commerce, E-commerce mobile et ubiquitaire

Rappel des bases du Java

Cours 1

Affectation, opérations, entrée/sortie, conditions, les itérations, les tableaux et les fonctions.

Université Internationale de Casablanca

Programme de la séance

- Histoire du java
- L'environnement de travail
- Typage des variables et affectation
- Les entrées/sorties
- Les opérateurs
- Les itérations
- Les tableaux
- Les fonctions

Université Internationale de Casablanca

Introduction générale au langage Java

- Historique du langage:

  - ❑ Java a été publié en 1995 par Sun Microsystems
  - ❑ Licence libre: Oracle Binary Code Licence

- Java est un langage de programmation :

  ## ❑ Multi-paradigme :

  Il supporte la programmation fonctionnelle et la programmation orientée objet.

  ## ❑ Multi-plateforme:

  Les applications java s'exécutent sur différents systèmes d'exploitation comme Windows, OS / 2, Mac, DOS ou Linux et Système Android

  ## ❑ Très haut-niveau:

  Il utilise des mots usuels des langues naturelles et des symboles mathématiques familiers.

  Il fait abstraction des caractéristiques techniques du matériel utilisé pour exécuter le programme.

Introduction générale au langage Java

- Il favorise la programmation impérative structurée :
  - ❑ Organisation hiérarchique simple du code
  - ❑ Utilisation de structures de contrôles : while, for, if, else, …
  - ❑ Décomposition du code en packages.
- Il favorise la programmation orienté objet :
  - ❑ Les objets, les classes, les méthodes et méthodes spéciales
  - ❑ L'encapsulation, l'héritage et le polymorphisme
- Il possède une riche bibliothèque standard :
  - ❑ Bibliothèque standard est organisée hiérarchiquement par packages
  - ❑ Exemple de bibliothèques :
    - java.lang.Math : fonctions mathématiques (*sin*, *cos*, *sqrt*, *abs*, *log*, *exp,* …)
    - java.awt.TextArea : affiche du texte sur une interface graphique
- Il est similaire au C++, C# et Python.

Université Internationale de Casablanca

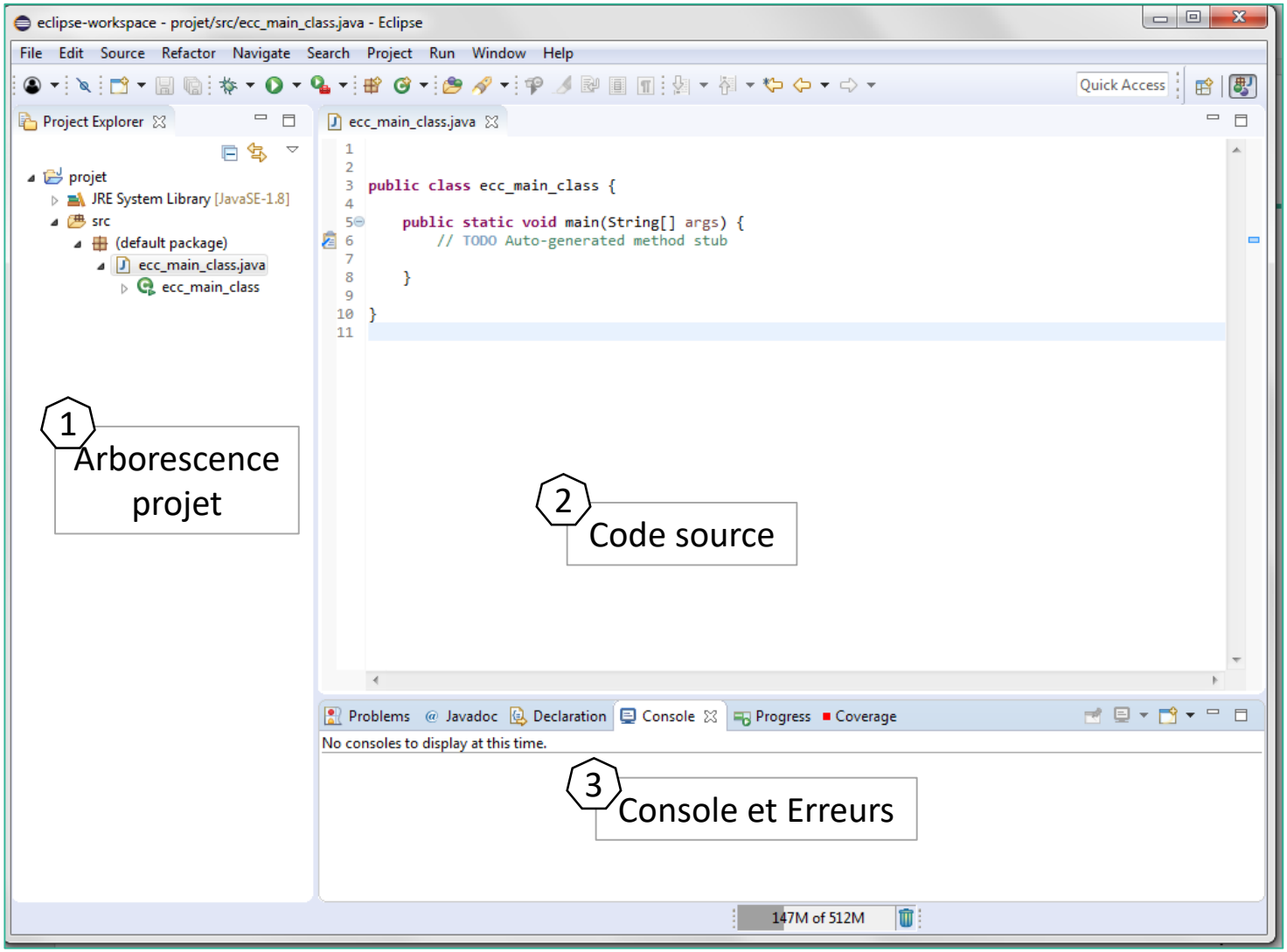
JVM, JRE et JDK

- Langage compilé:
  - ❑ Le code source (.java) est transformé en byte code (.class).
  - ❑ Le byte code n'est pas directement lisible par l'ordinateur.
  - ❑ Il nécessite une dernière étape de compilation qui est effectuée au moment de l'exécution.
- Java Virtual Machine (JVM):
  - ❑ Machine virtuelle.
  - ❑ Il permet la lecture du byte code généré après compilation.
- Java Runtime Environment (JRE):
  - ❑ Il permet d'exécuter des programmes java.
  - ❑ Il utilise la JVM pour l'exécution du programme.
- Java Development Kit (JDK):
  - ❑ Ensemble d'outils pour compiler et déboguer votre code.
  - ❑ Il contient le JRE.

L'espace de travail

- Eclipse

  - ❑ Eclipse est l'environnement de développement Java le plus utilisé.
  - ❑ Il existe plusieurs versions d'Eclipse (elios, kepler, oxygen, mars ...).



Les commentaires

- Les commentaires sont très utiles.
  - ❑ Les commentaires servent à expliquer certaines parties du code du programme.
  - ❑ Les commentaires facilitent la compréhension du code par d'autres personnes.

```
public class ecc_main_class {

    public static void main(String[] args) {
        // Commentaire court sur une seule ligne

        /* Plusieurs commentaires ecrirent
         * sur plusieurs lignes
         */

    }

}
```

Université Internationale de Casablanca

Structure d'un programme classe Java

- Une classe est un ensemble de codes contenant plusieurs instructions.
  - ❑ Une classe est composée d'une entête et d'un corps.
  - ❑ L'entête contient toutes les bibliothèques de la classe.
  - ❑ Le corps du programme contient les déclarations des variables et des instructions.
  - ❑ La fonction main() est la fonction par laquelle démarre le programme.

Entête

Corps

```
// les bibliotheques

public class Main {
    // Variables globales

    // Définition des fonctions

    public static void main(String[] args) {
        // Variables locales
        // Les instructions
    }

    // Déclaration des fonctions
}
```

- En Java, une instruction se termine toujours par un point virgule « ; ».

Université Internationale de Casablanca

Affectation de valeur à une variable

- Variable: nom qui référence une valeur en mémoire
  - ❑ Utilisable à la place des valeurs dans les expressions
  - ❑ Prend le type de la valeur référencée.

- Affectation: création de la liaison entre une variable et une valeur
  - ❑ Exemple: `int mon_age = 20 ;`
  - ❑ On peut réaffecter une nouvelle valeur à une variable existante, la valeur précédente est supprimée et remplacée par la nouvelle valeur.
  - ❑ La valeur peut être le résultat d'une expression, exemple: `float deux_pi = 2*3.14;`

type variable = valeur ;

  - ❑ L'affectation en 3 étapes:
    - ➢ Le programme Java commence par évaluer l'expression (**2*3.14**) et note le type du résultat (6.28).
    - ➢ Il vérifie si le type correspond au type double, si oui, il crée une variable nommée deux_pi et lui affecte la valeur 6.28.
    - ➢ Dans le cas ou le type du résultat ne correspond pas au type de la variable, une erreur d'alerte s'affiche.

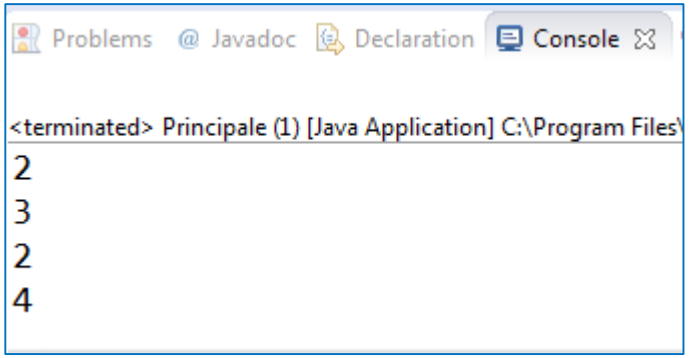
Université Internationale de Casablanca

Incrémentation / Décrémentation

- Incrémentation: **a = a + 1** ⟺ **a ++**
- Décrémentation: **a = a - 1** ⟺ **a --**
- Incrémentation par pas de **n**: **a = a + n ou a += n**
- Décrémentation par pas de **n** : **a = a - n ou a -= n**

A (opération) = B ⟺ *A = A (opération) B*

```
public class Principale {

    public static void main(String[] args) {

        int var = 2;
        int i = 2;

        System.out.println(var);

        var++; // ou var = var + 1;
        System.out.println(var);

        var--; // ou var = var + 1;
        System.out.println(var);

        var*=i; // ou var = var + 1;
        System.out.println(var);
    }
}
```



Université Internationale de Casablanca

Convention de nommage des variables

Pour les noms des variables et des fonctions:

- Début a..zA..Z ou _ ensuite a..zA..Z0..9 ou _
  - ❑ Mots clés du langage sont interdits
  - ❑ Accents à éviter
  - ❑ Les espaces sont remplacés par le symbole « _ »
  - ❑ Java est case sensitive: distinction entre majuscule et minuscule

- Exemples:
  - ❑ X1
  - ❑ y_min
  - ❑ maxValue
  - ❑ mon_age,
  - ❑ _num
  - ❑ PI
  - ❑ …

Mots clés du langage Java				
and	del	from	none	true
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	false	in	pass	yield
continue	Finally	is	raise	
def	For	lambda	return	

Type primitifs et chaines de caractères

- Il existe 8 types primitifs définis en Java
  - **boolean**: true or false.
  - **char** : un seul caractère.
  - **byte, short, int, long** : entiers.
  - **float, double** : flottants.

- Les chaines de caractères
  - Les chaînes de caractères en Java ne sont pas représentées avec un type primitif.
  - On utilise plutôt une classe d'objet : **String**
  - On met la valeur de la chaines entre des guillemets **("").**

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

    int a=2;
    int b = -2;
    double c = 2.5;
    double d = 1.2e-3;
    boolean e = true;
    char f = 'a';
    String g = "Centrale Casablanca";
}
```

```
blems @ Javadoc Declaration Console
nated> ecc_main_class [Java Application] C:\Program F
2
-2
2.5
0.0012
true
a
Centrale Casablanca
```

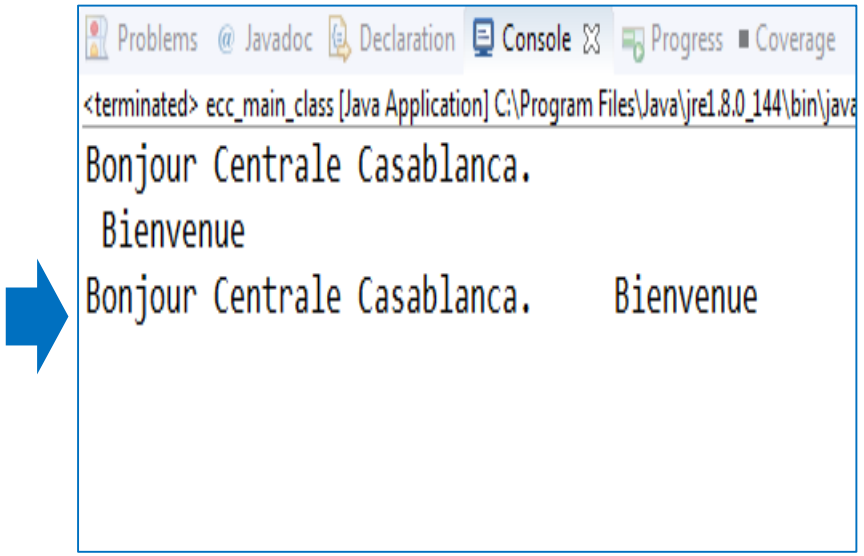
Université Internationale de Casablanca

Chaîne de caractères

- Pour les chaînes qui contiennent des sauts de lignes, on utilise un « **\n** » (antislash n).
- Pour les chaînes qui contiennent des tabulations, on utilise un « **\t** » (antislash t).
- Caractère d'échappement dans une chaîne: \

\n → retour à la ligne
\t → tabulation

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

    String chaine1="Bonjour Centrale Casablanca.\n Bienvenue";
    String chaine2="Bonjour Centrale Casablanca.\t Bienvenue";
    System.out.println(chaine1);
    System.out.println(chaine2);
}
```



Université Internationale de Casablanca

Opérations sur les chaines de caractères

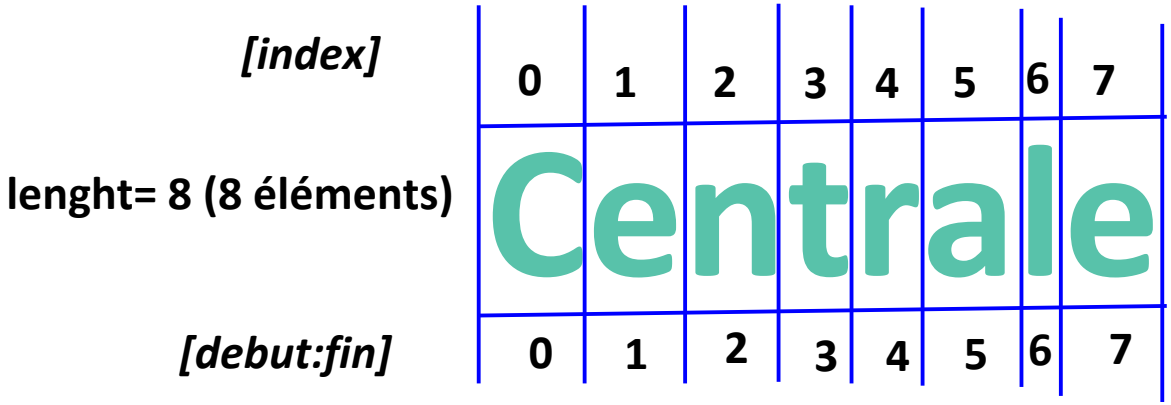
- **Concaténation :** On peut concaténer des chaînes avec la méthode **concat(String s)**
- **Longueur :** On peut connaitre la longueur d'une chaîne via la méthode **lenght()**
- **Comparaison :** On peut comparer deux chaines avec l'opérateur ==
  - **int compareTo (String s)** : compare deux chaines. Renvoie true si les valeurs sont égales.
  - **int compareToIgnoreCase (String s)** : compare deux chaines, ignore les casses. Renvoie true si les valeurs sont égales.
  - **boolean equals(Object o)** : compare une chaine à un objet et retourne *true* en cas d'égalité et *false* sinon.
- **Formatage de chaînes de caractères :** On peut supprimer les espaces dans une chaine. La méthode **trim()** permet d'effectuer cette opération.
  - **String toLowerCase()**: retourne une chaîne égale à la chaîne convertie en minuscules.
  - **String toUpperCase()**: retourne une chaîne égale à la chaîne convertie en majuscules.
  - **String replace(char a, char b)**: retourne une chaîne en remplaçant tous les caractères **a** par **b**.
- **Caractère et sous-chaines :** La méthode **substring()** permet l'extraction d'une chaine
  - **Char charAt(int i)** : retourne le caractère à l'indice **i**.
  - **String substring(int i)**: retourne une sous-chaîne depuis l'indice **i** jusqu'à la fin.
  - **String substring(int i, int j)**: retourne une sous-chaîne comprise entre l'indice **i** et **(j-1)**.

Université Internationale de Casablanca

Indexation des éléments de chaîne de caractères

Un caractère **[*index*]**

Une sous-chaîne **[*début*:*fin*]**




```
String chaine = "Centrale"
```

chaine.lenght() ⟶ 8 chaine.substring(1) ⟶ "entrale" chaine.substring(1,5) ⟶ "entr"

chaine.toUpperCase() ⟶ " CENTRALE " chaine.charAt(2) ⟶ "n"

Entrée / Sortie

- **Afficher un message à l'écran**

En java, pour afficher un message dans la console, utiliser les instructions:

```
System.out.println("votre message")
```

Retour à la ligne après l'affichage du message

```
System.out.print("votre message")
```

Pas de retour à la ligne

- **Afficher la valeur d'une variable**

```
System.out.println(var)
```

```
System.out.print(var )
```

- **Affichage message + variable**

Pour mettre du texte en même temps que vos variable, utiliser l'opérateur de concaténation « **+** » qui sert aussi à mélanger du texte brut et des variables.

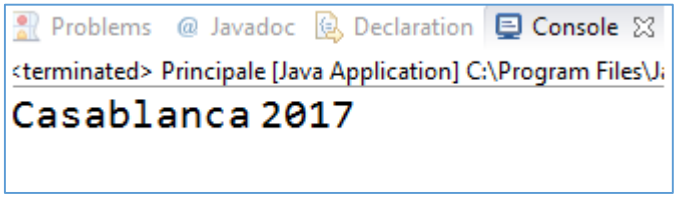
```
System.out.println("votre message" + var)
```

```
System.out.print ("votre message" + var)
```

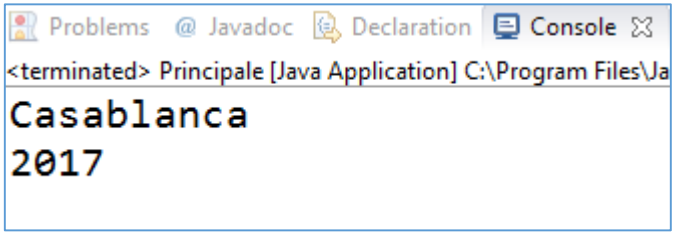
Entrée / Sortie

- Exemple

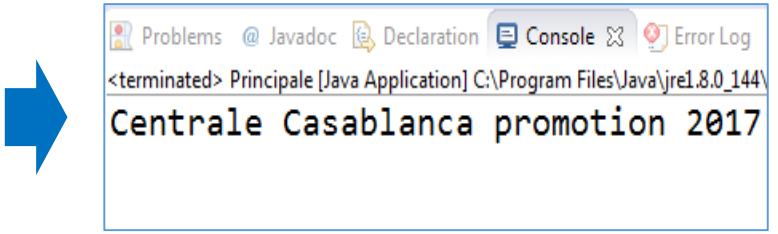
```
public class Principale {

    public static void main(String[] args) {

        String ville= "Casablanca";
        int annee = 2017;
        System.out.print(ville);
        System.out.print(annee);
    }
}
```




```
public class Principale {

    public static void main(String[] args) {

        String ville= "Casablanca";
        int annee = 2017;
        System.out.println(ville);
        System.out.println(annee);
    }
}
```




```
public class Principale {

    public static void main(String[] args) {

        String ville= "Casablanca";
        int annee = 2017;
        System.out.println("Centrale " + ville + " promotion " + annee);
    }
}
```



Entrée / Sortie

- **La classe Scanner**

Pour que Java puisse lire ce que vous tapez au clavier, vous allez devoir utiliser une instance de la classe **Scanner** en rajoutant en paramètre, l'entrée standard : **System.*in***.

La classe **Scanner** se trouve dans le package **java.util,** il faut donc l'importer via la syntaxe:
import java.util.Scanner

```
Scanner sc =new Scanner(System.in)
```

- **Récupérer une saisie**

Pour récupérer du texte : `String t = sc.next();`

Pour récupérer un nombre entier: `int e = sc.nextInt();`

`long l = sc.nextLong();`

Pour récupérer un nombre décimal: `double d = sc.nextDouble();`

Pour récupérer une valeur booléenne : `boolean b = sc.nextBoolean();`

Entrée / Sortie

- Exemple

```
public class Principale {

    public static void main(String[] args) {
        Scanner sc =new Scanner(System.in);

        System.out.println("Saisissez votre nom: ");
        String nom = sc.next();

        System.out.println("Saisissez votre age: ");
        int age = sc.nextInt();

        System.out.println("Saisissez votre moyenne: ");
        double moyenne = sc.nextDouble();



        System.out.println("Votre nom est: "+ nom);
        System.out.println("Votre age est: "+ age);

        if(moyenne >= 10){
            System.out.println("Vous etes admin :)");
        }
        else {
            System.out.println("Vous avez échoué :(");
        }
    }
}
```

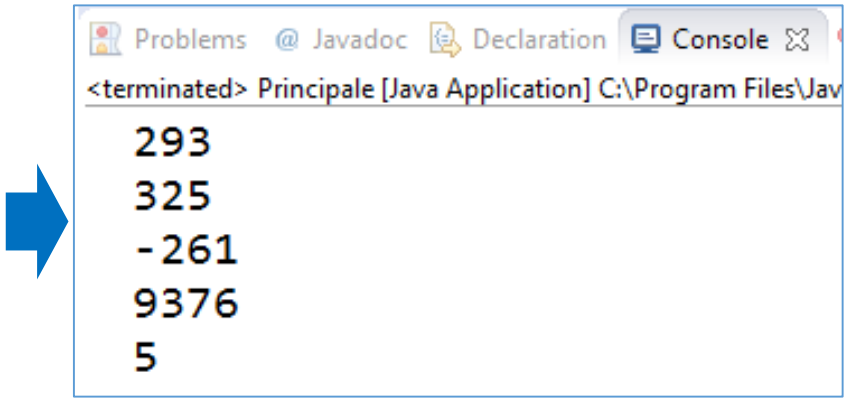
```
Problems  @ Javadoc  Declaration  Console  Error Log
<terminated> Principale [Java Application] C:\Program Files\Java\jre1.8.0_144\b
Saisissez votre nom:
Landry
Saisissez votre age:
24
Saisissez votre moyenne:
16
Votre nom est: Landry
Votre age est: 24
Vous etes admin :)
```

Université Internationale de Casablanca

Opérations arithmétiques

- Opérations sur les nombres:
  - **+ addition**
  - **- soustraction**
  - *** multiplication**
  - **/ division. La division intégrale est tronquée, non arrondie.**
  - **% modulo. reste de la division entière (ne s'applique pas pour le point flottant).**

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

    int i = 32;
    int j = 5 + 9 * i; // j = 293

    int add= i + j; // add = 325
    int sous = i - j; // sous = -261
    int mult = i * j; // mult = 9376
    int mod = j % i; // mod = 5

    System.out.println(j);
    System.out.println(add);
    System.out.println(sous);
    System.out.println(mult);
    System.out.println(mod);
}
```



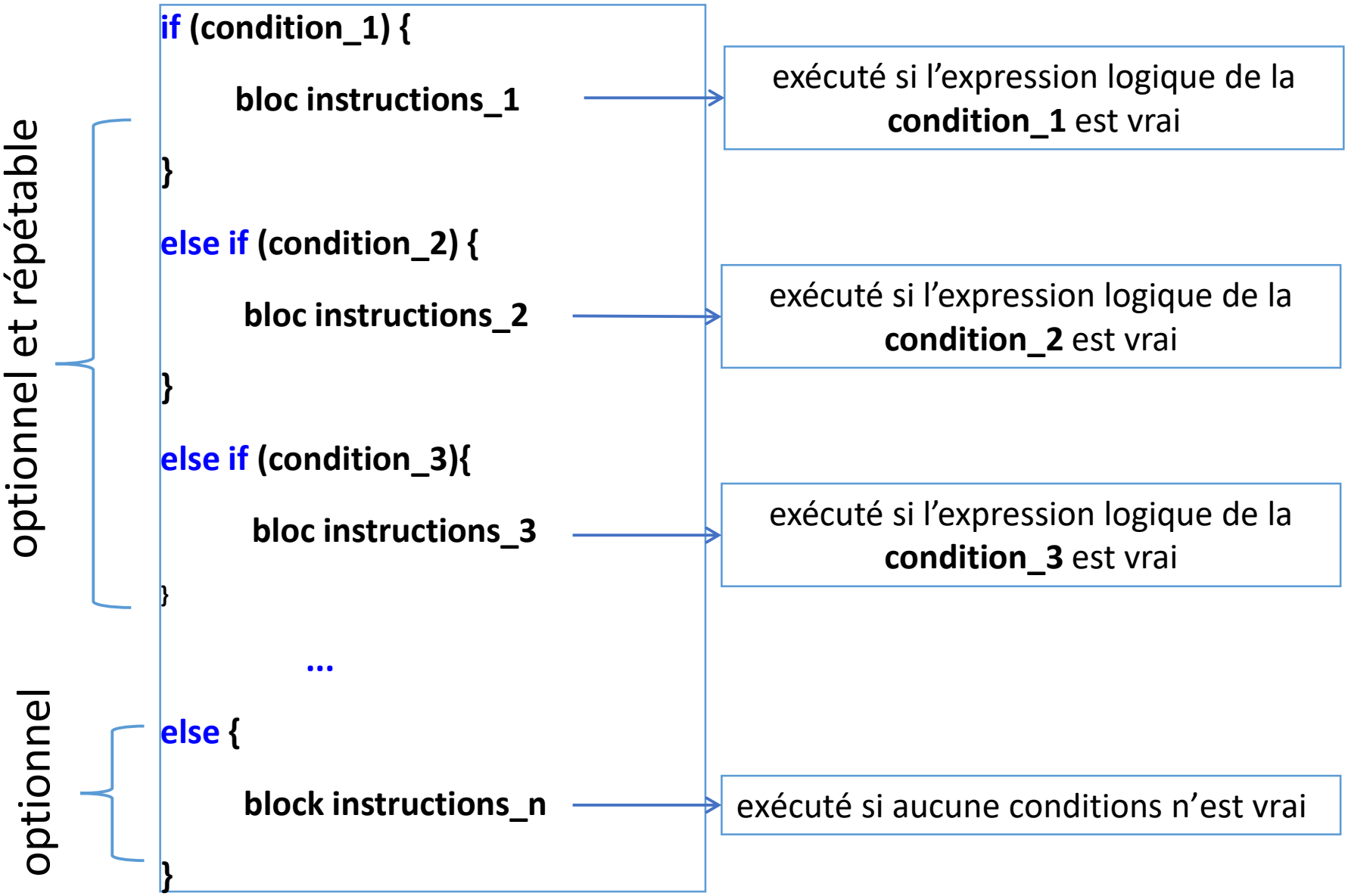
Université Internationale de Casablanca

Opérations booléennes

- Valeurs:
  - Valeur constante vrai: **True**
  - Valeur constante faux: **False**
- Opérateurs booléens: **!**, **&&**, **||**
  - Négation logique : **! a**
  - Conjonction logique (les deux en même temps) : **a && b**
  - Disjonction logique (l'un ou l'autre ou les deux) : **a || b**
- Operateurs de comparaisons:
  - Inférieur: **<**
  - Supérieur: **>**
  - Inférieur ou égal: **<=**
  - Supérieur ou égal: **>=**
  - Egalité: **==**
  - Différence: **!=**

Structure conditionnelle : If

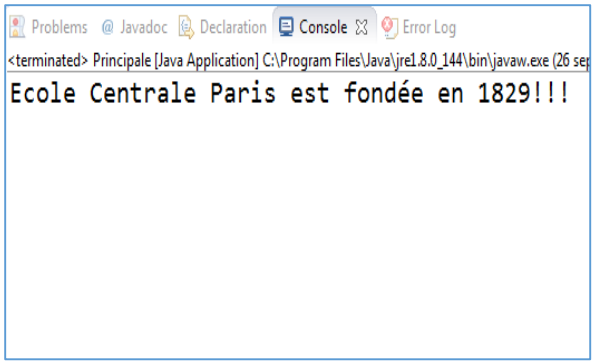
- L'instruction **if** prend une condition booléenne entre parenthèses.



Exemples

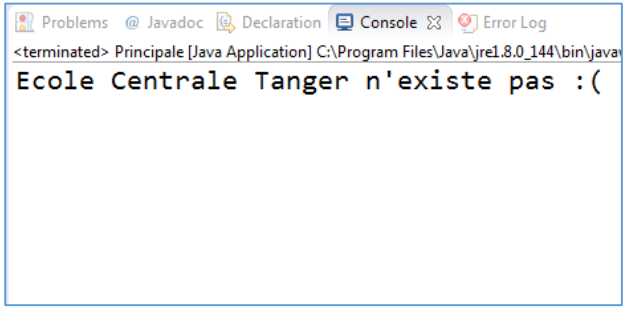
- **If** et **else**

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

    String ville = "Tanger";

    if (ville.equals("Paris")) {
        System.out.println("Ecole Centrale Paris est fondée en 1829!!!");
    }
    else {
        System.out.println("Ecole Centrale Tanger n'existe pas :(");
    }
}
```



- **If, else if** et **else**

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

    String ville = "Tanger";

    if (ville.equals("Paris")) {
        System.out.println("Ecole Centrale Paris est fondée en 1829!!!");
    }
    else if (ville.equals("Casablanca")){
        System.out.println("Ecole Centrale Casablanca est fondée en 2013!!!");
    }
    else if (ville.equals("Nantes")){
        System.out.println("Ecole Centrale Nantes est fondée en 1919!!!");
    }
    else {
        System.out.println("Ecole Centrale Tanger n'existe pas :(");
    }
```



Université Internationale de Casablanca

Structure conditionnelle : switch

- L'instruction **switch** permet de gérer plusieurs **if** imbriquées.

```
switch (clef) {
case valeur_1:
        instructions_1
        break;
 case valeur_2:
        instructions_2
        break;
case valeur_3:
        instructions_3
        break;
            ...
default:
        instructions_n
        break;
```

exécuté si l'expression logique: (**clef == valeur_1**) est vrai

exécuté si l'expression logique: (**clef == valeur_2**) est vrai

exécuté si l'expression logique: (**clef == valeur_3**) est vrai

exécuté si aucune conditions n'est vrai

Exemples

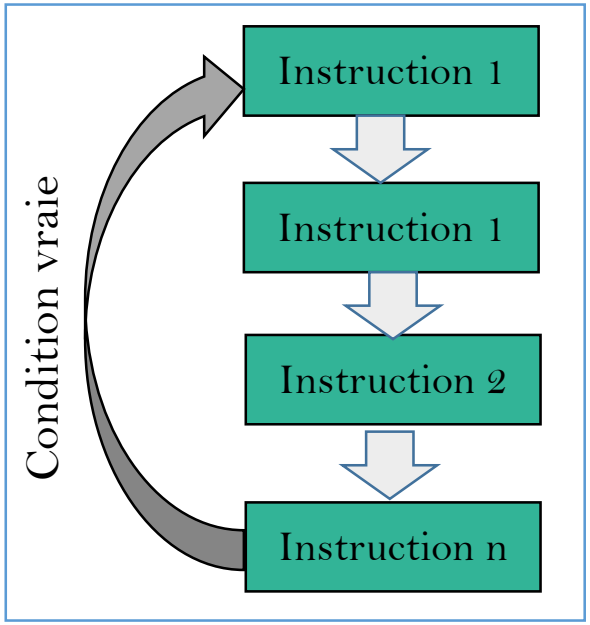
```
public static void main(String[] args) {

    String ville = "Tanger";

    switch (ville) {
    case "Paris":
        System.out.println("Ecole Centrale Paris est fondée en 1829!!!");
        break;
    case "Casablanca":
        System.out.println("Ecole Centrale Casablanca est fondée en 2013!!!");
        break;
    case "Nantes":
        System.out.println("Ecole Centrale Nantes est fondée en 1919!!!");
        break;
    default:
        System.out.println("Ecole Centrale Tanger n'existe pas :(");
        break;
    }
```

Problems @ Javadoc Declaration Console Error Log

```
Ecole Centrale Tanger n'existe pas :(
```

Itérations

- La notion de boucle (itération) est utilisée quand on doit exercer plusieurs fois le même traitement sur un même objet.
- L'arrêt d'une boucle est basée sur un **test de continuité**.
- Le test de continuité appelé **condition d'arrêt** sert à sortie de la boucle.
- On sort de la boucle quand la condition d'arrêt n'est plus vérifiée.
- Il existe 3 types de boucle en java:
  - ❑ **Boucle while**
  - ❑ **Boucle do … while**
  - ❑ **Boucle for**



Université Internationale de Casablanca

Boucle while

•**while** signifie « tant que ».

•Si on souhaite répéter une séquence d'instructions un nombre de fois non déterminé à l'avance, on utilise une boucle **while**. La syntaxe en java est la suivante :

```
while (condition d'arrêt) {
   instructions
}
```

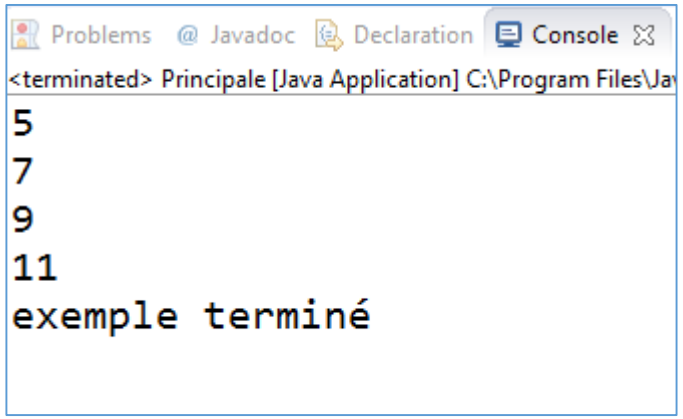
•**condition** est une expression booléenne.

•Les instructions à l'intérieur du bloc s'exécuteront tant que la condition reste vraie.

Boucle while

❑Instruction while : exemple

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

    int i = 3;
    while (i < 10) {
        i+=2;
        System.out.println(i);
    }
    System.out.println("exemple terminé");
}
```



Boucle do … while

•**do … while** signifie « faire tant que ».

•Il peut arriver que l'on ne veuille effectuer le test de continuation qu'après avoir exécuté l'instruction. Dans ce cas, la boucle do ...while est utile.

•Boucle très similaire à la boucle **while**. La syntaxe en java est la suivante :

```
do {
   instructions
} while(condition d'arrêt)
```

•**condition** est une expression booléenne.

•Les instructions à l'intérieur du bloc s'exécuteront tant que la condition d'arrêt sera fausse.

•Avec la boucle **do** … **while**, on exécute les instructions avant de tester la condition d'arrêt. Contrairement à la boucle **while** ou on test la condition d'arrêt avant d'exécuter les instructions.

Boucle do … while

❑Instruction do … while : exemple

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

    int nombre;
    Scanner sc=new Scanner(System.in);

    do{
        System.out.println("Saisir un nombre entre 1 et 10: ");
        nombre=sc.nextInt();
    }while (nombre  < 1 || nombre > 10);

    System.out.println("exemple terminé");
}
```

```
Problems  Javadoc  Declaration  Console  Error Log
<terminated> Principale [Java Application] C:\Program Files\Java\jre1.8.0_144\
Saisir un nombre entre 1 et 10:
0
Saisir un nombre entre 1 et 10:
15
Saisir un nombre entre 1 et 10:
-5
Saisir un nombre entre 1 et 10:
8
exemple terminé
```

Boucle for

- **for** signifie « pour ».
- La boucle for est l'équivalent de la boucle while
- Ce type de boucle est très utilisée pour les itérations.
- Syntaxe :

```
for (initialisation; expression booléenne; étape) {
  instructions
}
```

- ❑ **initialisation**: cette instruction est exécutée une fois, juste avant la première itération de la boucle. Il est souvent utilisé pour déclarer la variable de contrôle et lui attribue une valeur initiale.
- ❑ **expression booléenne**: évaluée avant chaque itération de la boucle (après l'initialisation pour la première). Le corps de la boucle sera exécuté lorsque l'expression est vraie. Sinon, la boucle est terminée.
- ❑ **étape**: évaluée à la fin de chaque itération de la boucle, avant que l'expression booléenne soit évaluée. Il est souvent utilisé pour incrémenter la variable de contrôle.

Université Internationale de Casablanca

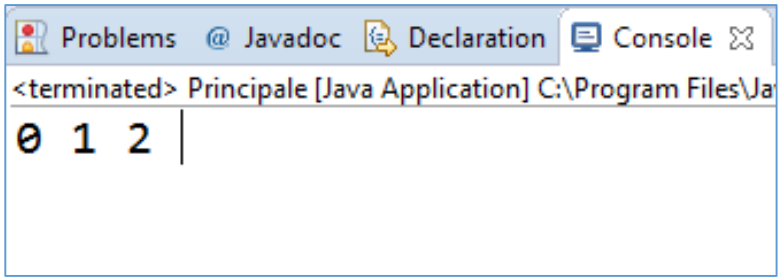
break et continue

❑Break et continue :

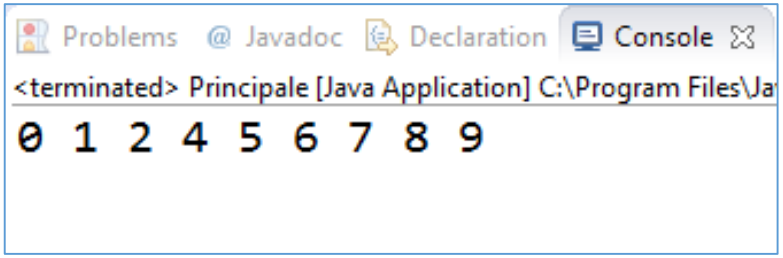
•L'instruction **break** permet de sortie d'une boucle.

•L'instruction **continue** est similaire, mais au lieu d'interrompre la boucle, elle revient au début.

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

    for (int i = 0; i < 10; i++) {
        if(i==3){
            break;
        }
        System.out.print(i+" ");
    }
}
```




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

    for (int i = 0; i < 10; i++) {
        if(i==3){
            continue;
        }
        System.out.print(i+" ");
    }
}
```



Université Internationale de Casablanca

Les tableaux

- Un tableau est une structure de données contenant un groupe d'éléments tous du même type.
  - Le type des éléments peut être un type primitif ou un objet.
  - Utiliser les [] pour spécifier qu'il s'agit d'un tableau
- **Tableau unidimensionnel** est un tableau à une dimension
  - Déclaration d'un tableau : `type tableau [] = {valeur1, valeur2, …., valeurn};`
  - Déclaration d'un tableau de taille spécifique: `type tableau [] = new type [taille];`
  - Pour connaitre la taille d'un tableau, utiliser la fonction **length** : `int taille = tableau.lenght;`
- **Tableau multidimensionnel** est un tableau contenant au minimum deux dimensions.
  - Déclaration d'un tableau : `type tableau [][] = {{valeur1, … ,valeurn}, …., {valeur1, … ,valeurn}};`
  - Déclaration d'un tableau de taille spécifique: `type tableau [] [] = new type [taille1] [taille2];`

Structure des tableaux

- Exemple tableau unidimensionnel

```
int notes [] = {12, 15, 9, 8, 12, 14};
```

❑ Tableau de taille **6**.

12	15	9	8	12	14

- Exemple tableau bidimensionnel (deux dimensions)

```
int notes [] [] = { {12, 15, 9, 8, 12, 14}, {20, 14, 16, 8, 10, 11}, {13, 16, 9, 14, 15, 6} };
```

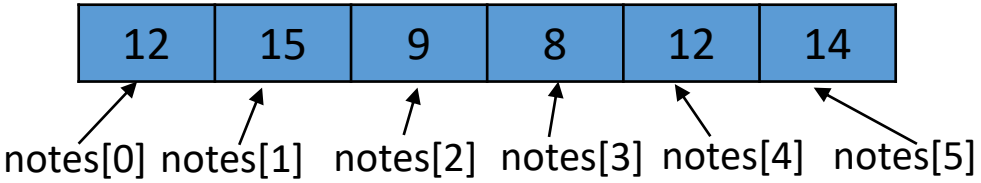
❑ Tableau de taille **3 x 6**.

1	→	12	15	9	8	12	14
2	→	20	14	16	8	10	11
3	→	13	16	9	14	15	6

Recherche dans un tableau

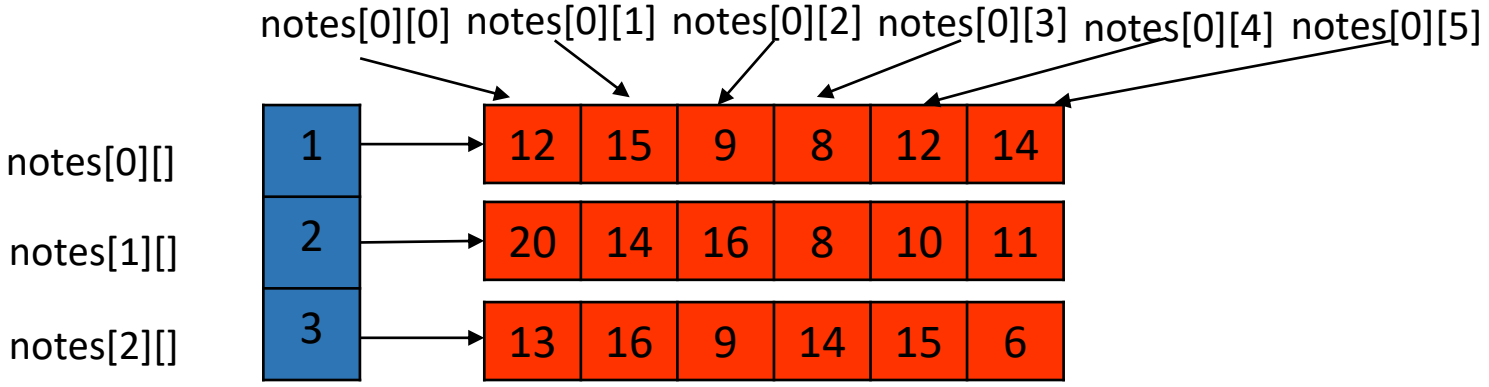
- On accède à un élément d'un tableau grâce à son **indice**.
- Un tableau débute toujours par **l'indice 0**.

```
int notes [] = {12, 15, 9, 8, 12, 14};
```



- Le nombre d'indice correspond à la dimension du tableau.

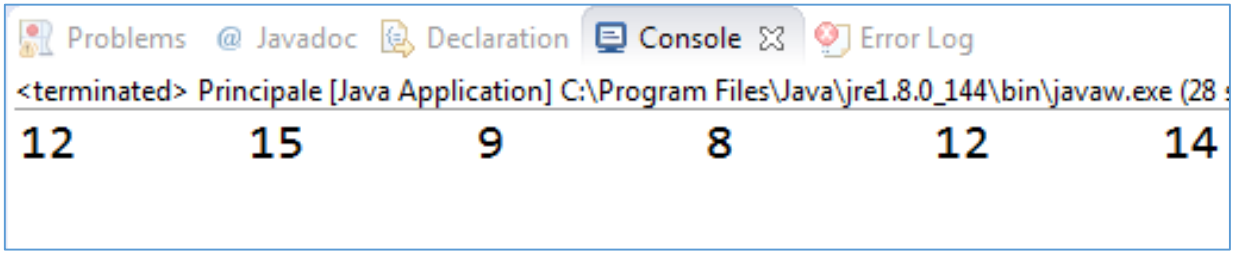
```
int notes [] [] = { {12, 15, 9, 8, 12, 14}, {20, 14, 16, 8, 10, 11}, {13, 16, 9, 14, 15, 6} };
```



Parcours d'un tableau unidimensionnel

- Utiliser une boucle **for** pour parcourir un tableau unidimensionnel

```
public class Principale {

    public static void main(String[] args) {

        int notes [] = {12, 15, 9, 8, 12, 14};

        for(int i=0; i<notes.length; i++){
            System.out.print(notes[i]+"\t");
        }
    }
```



Parcours d'un tableau multidimensionnel

- Utiliser deux boucles **for** pour parcourir un tableau bidimensionnel

```
public class Principale {

    public static void main(String[] args) {

        int notes[][] ={{12,15,9,8,12,14},{20,14,16,8,10,11},{13,16,9,14,15,6}};

        for(int ligne=0; ligne<notes.length; ligne++){
            for(int colonne=0; colonne<notes[ligne].length; colonne++){
                System.out.print(notes[ligne][colonne]+"\t");
            }
            System.out.println();
        }
    }
```

Problems @ Javadoc Declaration Console Error Log

<terminated> Principale [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (28 s

```
12      15      9       8       12      14
20      14      16      8       10      11
13      16      9       14      15      6
```

Université Internationale de Casablanca

Fonctions

- Une fonction est un ensemble d'instruction permettant d'effectuer un traitement.
- Syntaxe:

```
public static type nom_fonction ([liste_parametre]) {
        instructions …
}
public static void nom_fonction ([liste_parametre]) {
        instructions …
}
```

- La déclaration d'une fonction se fait à l'extérieur de la fonction principale **main().**
- **L**'appel de la fonction se fait à l'intérieur de la fonction principale **main()**.

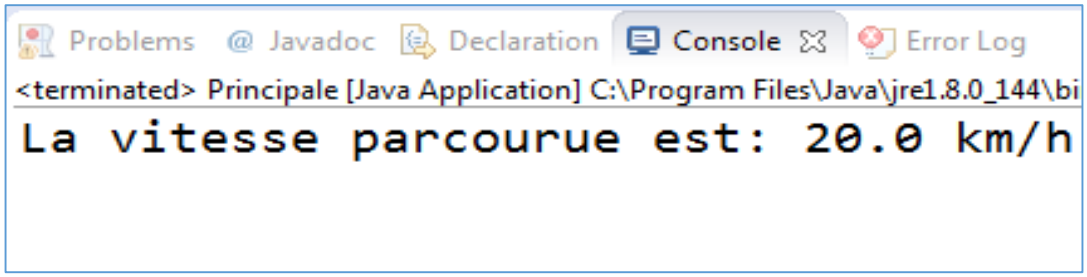
Fonctions

- Exemple: méthode qui calcul et retourne la vitesse d'un véhicule

```
public class Principale {

    public static void main(String[] args) {
        int distance = 100;
        int  temps= 5;
        double v=vitesse(distance, temps);
        System.out.print("La vitesse parcourue est: " + v + " km/h");
    }

    public static double vitesse(int d, int t){
        return d/t;
    }
}
```

2 Appel fonction

1 Déclaration fonction



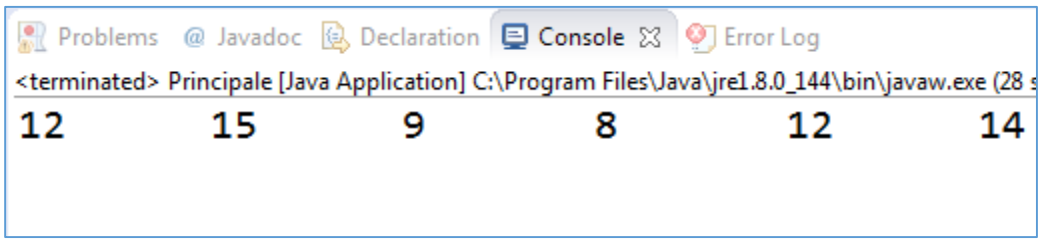
Fonctions

- Exemple: afficher le contenu d'un tableau multidimensionnel

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

    int tableau [] = {12, 15, 9, 8, 12, 14};
    parcourirTableau(tableau);

}

public static void parcourirTableau(int tab[]){
    for(int i=0; i<tab.length; i++){
        System.out.print(tab[i] + "\t");
    }
}
```

2 Appel fonction

1 Déclaration fonction



La récursivité

❑Exemple: la suite de Fibonacci

Remarques:

- La version itérative est beaucoup plus rapide
- La version récursive effectue un nombre exponentiel de multiplication
- Car elle ne mémorise pas les calculs déjà effectués
- La solution récursive n'est pas toujours la meilleure

Université Internationale de Casablanca

Ressources

- Documentation officielle :
  - https://docs.oracle.com/javase/7/docs/api/
- Tutoriel Java officiel :
  - https://docs.oracle.com/javase/tutorial/
- Livre :
  - “Elements of Programming Interviews in Java” par Adnan Aziz, Tsung-Hsien Lee et Amit Prakash.

QUESTIONS ?