

Groupe d'UEs NDC:

UE Développement d'applications



Email: kahina.hassam@yncrea.fr; vincent.lefevere@yncrea.fr;
carol.habib@yncrea.fr

Bureau: T336

Département: Organisation , Management et Informatique

Objectif de l'UE

Développer une application disposant d'une interface graphique

Résultats d'apprentissage

- Concevoir des algorithmes
- Utiliser un framework d'interfaces graphiques pour développer une IHM
- Développer en équipe en utilisant un outil collaboratif

Modalités

- Évaluations :
 - Contrôle intermédiaire écrit individuel (30%)
 - 2 TPs notés:
 - TP1 (10%) – Travail en binôme
 - TP2(25%) – Travail Individuel
 - Fin de projet noté (35% de la note finale) – Travail à 5
- Ressources mises à disposition :
 - Bibliothèque Vauban
 - Internet
 - Enseignants !
 - Kahina HASSAM-OUARI
 - Vincent LEFEVERE
 - Carol HABIB

Remarque

- Les TDs, TP_s projet se feront sur un logiciel qui s'appelle Eclipse, dont la procédure est décrite dans un fichier « prise en main de Eclipse » que vous trouverez sur itslearning.

Principes et concepts de base du langage Java

Pourquoi programmer?

- Répondre à des besoins, i.e. :
 - Résoudre un problème exprimé dans un langage compréhensible par l'ordinateur
 - Effectuer un traitement automatisé des données
 - ...
- Grande variété de besoins :
 - Applications mobiles
 - Secteur bancaire, Calcul scientifique
 - Industrie du loisir
 - Commande de machines

Un programme c'est quoi?



Une suite d'instructions indiquant, dans un langage compréhensible par la machine, ce qu'elle doit faire.



Dans ce cours, nous utiliserons le langage Java pour écrire nos programmes



Java...

Pourquoi faire ?

- On peut réaliser de nombreux types de programmes avec le Java:
 - des applications, sous forme de fenêtre ou de console ;
 - des applets, qui sont des programmes Java incorporés à des pages Web ;
 - des applications pour appareils mobiles, comme les smartphones, avec J2ME (Java 2 Micro Edition) ;
 - des sites web dynamiques, avec J2EE (Java 2 Enterprise Edition, maintenant JEE) ;
 - et bien d'autres : J3D pour la 3D. . .

Syntaxe du langage java

Un langage informatique est composé de :

- **Mots-clés :**
 - Constituent le vocabulaire du langage
- **Syntaxe : structures et règles**
 - La « grammaire » du langage (= la forme requise des instructions)
- **Conventions**
 - Constituent des règles de notations adoptées par tous les programmeurs

Vocabulaire Java...

les variables

- Une variable est un endroit de la mémoire auquel on a donné un nom de sorte que l'on puisse y faire facilement référence dans le programme
- Créer une variable implique de lui donner :
un nom + une valeur + un type
- La valeur d'une variable peut changer (varier) au cours du temps et de l'exécution du programme.
Par contre son type ne change pas !

Vocabulaire Java...

Trouver un nom

- Interdiction d'utiliser les mots-clés
- Le 1^{er} caractère d'un nom peut être:

AUTORISES	INTERDITS
Lettre	Chiffre
\$	Signe de ponctuation
—	

- Le nom ne doit jamais contenir :
 - des espaces
 - \approx caractères internationaux (accents, etc.)

Vocabulaire Java...

Types primitifs

- Java fournit plusieurs types prédéfinis : ce sont les types primitifs, leur nombre est limité

En français	Type en java	Valeurs possibles
Nombre entier	byte	-128 à 127
	short	-32768 à 32767
	int	-2^{31} à $2^{31}-1$
	long	-2^{63} à $2^{63}-1$
Nombre flottant (à virgule)	float	1.4×10^{-45} à 3.4×10^{38}
	double	4.9×10^{324} à 1.8×10^{308}
Caractère	char	tous?
Vrai ou faux	boolean	true ou false

Variable en Java...

Déclaration et initialisation

Type ET nom ET valeur

Type	Exemple de création et initialisation
int	int a = 12;
short	short b = 32;
long	long c = 200L;
byte	byte d = 010;
double	double e = 10.5;
float	float f = 23.2323f;
char	char g = 't';
boolean	boolean h = true;

Variable en Java...

Utilisation

○ Pour être utilisée, une variable en Java doit avoir été

- Déclarée (définir son nom et son type)

{
■ Initialisée (lui donner sa valeur de départ)
Peut se faire en même temps que la déclaration
■ **ou**
ré-Assignée (sa valeur a été modifiée)

○ Syntaxe:

- **int** t; Déclaration d'un entier t (t et le nom de la variable)

- **int** u = 3; Déclaration et initialisation d'un entier u

- t=7; Initialisation de t à la valeur 7

- u=t; Assignation (affectation) de la valeur de t à u

Grammaire Java

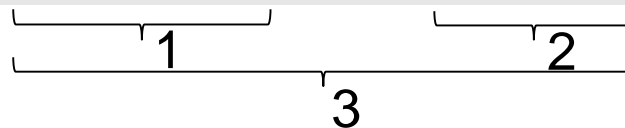
Opérateurs numériques 1/2

Niveau	Symbole	Signification
1	()	Parenthèse
2	*	Produit
	/	Division
	%	Modulo
3	+	Addition ou concaténation
	-	Soustraction

```
int i1 = 4;
```

```
int i2 = i1 + 4;
```

```
int i5 = (i1 % i2) - 5 * 4;
```



Grammaire Java

Opérateurs numériques 2/2

- Comparer des valeurs numériques :

Opérateur	Exemple	Renvoie TRUE si
>	<code>v1 > v2</code>	v1 plus grand que v2
>=	<code>v1 >= v2</code>	Plus grand ou égal
<	<code>v1 < v2</code>	Plus petit que
<=	<code>v1 <= v2</code>	Plus petit ou égal à
==	<code>v1 == v2</code>	égal
!=	<code>v1 != v2</code>	différent

```
int i1 = 4;  
int i3 = i1 + 4;  
boolean inferieur = i1 < i3;
```

Grammaire Java

Opérateurs d'affectation

- On remplace la valeur d'une variable avec le « = » :

```
int i = 5;  
i = 6;
```

- Il existe aussi des opérateurs faisant affectation et opération arithmétique et/ou logique en même temps

Opérateur	Exemple	Équivalent à
+=	expr1 += expr2	expr1 = expr1 + expr2
-=	expr1 -= expr2	expr1 = expr1 - expr2
*=	expr1 *= expr2	expr1 = expr1 * expr2
/=	expr1 /= expr2	expr1 = expr1 / expr2
%=	expr1 %= expr2	expr1 = expr1 % expr2
++	expr1++	expr1 = expr1 + 1
--	expr1--	expr1 = expr1 - 1

Grammaire Java

Opérateurs logiques

- Opérateurs logiques:

Opérateur	Usage	Renvoie TRUE si
&& &	expr1 && expr2 expr1 & expr2	expr1 et expr2 sont vraies Idem mais évalue toujours les 2 expressions
 	expr1 expr2 expr1 expr2	Expr1 ou expr2, ou les deux sont vraies idem mais évalue toujours les 2 expressions
!	! expr1	expr1 est fausse
!=	expr1 != expr2	si expr1 est différent de expr2

```
int i1 = 4;  
int i3 = i1 + 4;  
boolean inferieur = i1 < i3;  
boolean res = inferieur && (i3>0);
```

Vocabulaire Java...

Type chaîne de caractère

- Les chaînes de caractères sont essentielles et omniprésentes dans les programmes informatiques
- Il n'existe pas de type primitif « string » ☹
- En java, il existe **String**, c'est une classe

Grammaire Java

Opérateur pour les String

○ Déclaration `String s1, s2;`

○ Initialisation `s1 = "Je pense";`
`s2 = "donc je suis";`

○ Déclaration et initialisation en une ligne

```
String s3 = "Hello";
```

○ On peut « concaténer » (= mettre bout à bout) deux Strings pour n'en faire qu'une seule.

```
String s4 = "Je pense" + s2;
```

L'affichage de S4 à l'écran donnerait : Je pensedonc je suis

Grammaire Java...

Type chaîne de caractères

- Pour comparer deux chaînes de caractères, il faut utiliser la fonction java *equals()*.
- Cette fonction renvoie :
 - ✓ true : si les deux chaînes de caractères sont identiques
 - ✓ false : si elles ne le sont pas
- Exemple:

```
String chaine1, chaine2;  
boolean res;  
chaine1= "Bonjour";chaine2="Bonjour";  
res=chaine1.equals(chaine2);
```

L'affichage de res à
l'écran donnera :
true

Grammaire Java

Instructions et blocs

○ Une instruction

- Une seule par ligne
- Est presque toujours suivie par un « ; »
- Réalise un traitement particulier
- Renvoie éventuellement le résultat d'un calcul
- Il en existe plusieurs types: déclaration, assignation...etc


○ Un bloc

- Est une suite d'instructions entre accolades « { » et « } »
- Doit toujours être refermé (autant de « { » que de « } »)
- Délimite la portée des variables
- Suit (presque) toujours la déclaration de classe et méthodes
- Définit aussi le contenu des boucles et structures conditionnelles

Structure d'une classe en java

Simple exemple

```
public class nomDeVotreClasse {  
    //déclaration de variables globales  
  
    //1 méthode principale  
    public static void main(String[]  
        args) {  
        //déclaration de variables locales  
        //appel des méthodes  
  
    }  
  
    //d'autres méthodes de la classe  
  
}
```



Il n'existe
qu'un seul
main par
application

Les entrées- sorties

- Les composants d'entrées/sorties sont très divers:
 - claviers,
 - écrans,
 - imprimantes,
 - disques souples ou durs,
 - souris,
 - mais aussi tout appareil que l'on souhaite piloter par ordinateur

Les sorties

Affichage à l'écran

- **Ecriture:** Communiquer des valeurs vers l'extérieur (utilisateur). Se fait en général via un affichage à l'écran. En Java, cet affichage se fait sur la console.
- Elle permet :
 1. D'Afficher un texte :
 2. D'Afficher la valeur d'une variable :
 3. De mélanger de texte et des valeurs :

```
System.out.println("Texte à afficher");
```

```
System.out.println(nom_de_la_variable);
```

```
System.out.println("Texte"+nom_de_la_variable+  
"texte" +nom_de_la_variable);
```

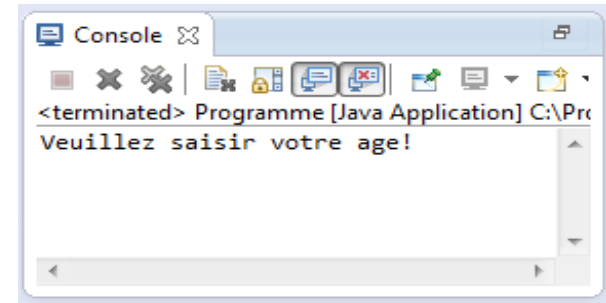
Les sorties

Par l'exemple

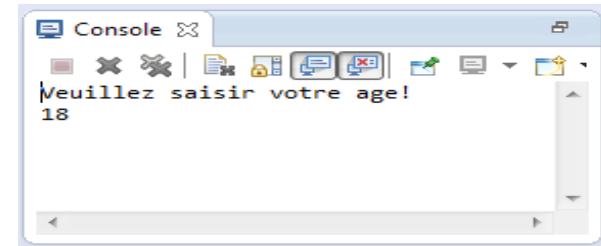
Programme en java:

```
public class Programme {  
  
    public static void main(String[] args) {  
        int age; ❶  
        System.out.println("Veuillez saisir  
votre age!"); ❷  
        age=18; ❸  
        System.out.println(age); ❹  
        System.out.println("Votre age est de "+age); ❺  
    }  
  
}
```

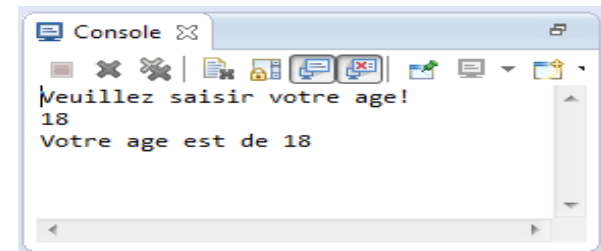
Instruction 2: affichage du texte



Instruction 4: affichage de la valeur



Instruction 5: texte et valeur



- ❑ **Lecture:** Entrer des valeurs externes (utilisateur) pour qu'elles soient utilisées dans la suite du programme. Par exemple par une saisie au clavier.
- Pour chaque type primitif il existe une méthode de récupération de données (**sauf les char**) :
 - `nextLine()`: pour récupérer une `String`
 - `nextInt()`: pour récupérer un entier (**int**),
 - `nextDouble()`: pour récupérer un double (**double**),
 - `nextFloat()`: pour récupérer un réel (**float**),
 - `nextBoolean()`: pour récupérer un booléen (**boolean**)

Les entrées

Syntaxe

○ Syntaxe pour lire des données clavier :

```
java.util.Scanner sc= new java.util.Scanner(System.in);  
System.out.println("Pour saisir un booléen (true /false)");  
boolean b=sc.nextBoolean();
```

```
java.util.Scanner sc1= new java.util.Scanner(System.in);  
System.out.println("Pour saisir un entier");  
int b=sc1.nextInt();
```

```
java.util.Scanner sc2= new java.util.Scanner(System.in);  
System.out.println("Pour saisir une chaine de caractères");  
String b=sc2.nextLine();
```

```
java.util.Scanner sc3= new java.util.Scanner(System.in);  
System.out.println("Pour saisir un float");  
float b=sc3.nextFloat();
```

Les entrées

Syntaxe

- Syntaxe pour lire des données clavier :

```
java.util.Scanner sc= new java.util.Scanner(System.in);  
System.out.println("Pour saisir un booléen  
(true /false)");  
boolean b=sc.nextBoolean();
```

```
java.util.Scanner sc1= new  
java.util.Scanner(System.in);  
System.out.println("Pour saisir un entier");  
int b=sc1.nextInt();
```

```
java.util.Scanner sc2= new  
java.util.Scanner(System.in);  
System.out.println("Pour saisir une chaine de  
caractères");  
String b=sc2.nextLine();
```

```
java.util.Scanner sc3= new  
java.util.Scanner(System.in);  
System.out.println("Pour saisir un float");  
float b=sc3.nextFloat();
```

1-Créer une variable de type Scanner

2- Mettre une phrase pour guider l'utilisateur dans sa saisie.

3- Selon le type qu'on veut saisir utiliser la méthode adéquate

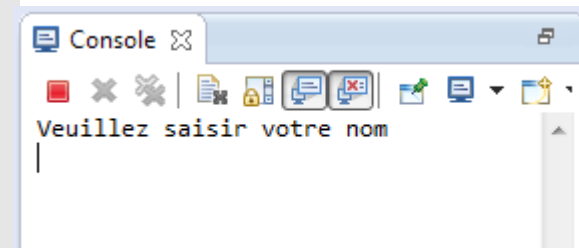
Les entrées

Par l'exemple

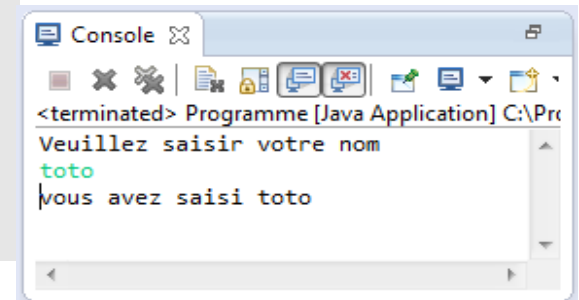
Programme en java:

```
public class Programme {  
  
    public static void main(String[] args) {  
        java.util.Scanner sc =  
            new java.util.Scanner(System.in);  
        System.out.println("Veuillez saisir  
votre nom"); ❶  
        String nom=sc.nextLine(); ❷  
        System.out.println("vous avez saisi  
"+nom); ❸  
    }  
  
}
```

Instruction 1 et 2: affichage
du texte+ apparition d'un
curseur pour la saisir



Instruction 3: affichage de la
saisie utilisateur



Les entrées

Génération aléatoires..

Pour mettre une valeur aléatoire dans une variable aléatoirement (non saisie par l'utilisateur):

1. On crée une variable Random:

```
java.util.Random generateur = new java.util.Random(System.currentTimeMillis());
```

2. Selon le type des valeurs qu'on souhaite générer, on appelle la méthode correspondante sur la variable créée:

Appel	Génère
<code>generateur.nextBoolean() ;</code>	true ou false
<code>generateur.nextInt(25) ;</code>	Un entier compris entre [0,24]
<code>generateur.nextDouble() ;</code>	Un double compris entre [0.0,1.0[
<code>generateur.nextFloat() ;</code>	Un float compris entre[0.0, 1.0f[

Grammaire Java

Structures conditionnelles

- Si *condition* alors ... : la structure **if**

```
if (CONDITION_VRAIE) {  
  
}
```

- Si *condition* alors ... sinon ... la structure **if..else**

```
if (CONDITION_VRAIE) {  
  
} else {  
  
}
```

Exercice

Votre premier programme

- Ecrire un programme java qui demande à un étudiant 5 valeurs correspondant aux notes de 5 matières.
- Si la moyenne de ses 5 notes est supérieure à 10 alors on affiche que l'étudiant est admis, sinon qu'il est recalé.