
INITITATION AU LANGAGE JAVA

Licence

Support de cours

2012-2023

Auteur: SANDA MAHAMA Amadou Tidjani

Tel : +229 95 05 33 43/ 96 36 80 96

Email: sandatidjani@google.com

CHAPITRE I INTRODUCTION

Java est un langage de programmation moderne développé par Sun Microsystems (aujourd'hui racheté par Oracle). Il présente beaucoup d'intérêts ce qui justifie le nombre de plus en plus grandissant de la communauté des programmeurs java. Au nombre de ces intérêts, on peut citer le fait que :

- Un programme java peut s'exécuter sur différents ordinateurs, tels que PC, Apple et autres, sans modification. En fait, les programmes Java ne savent même pas où ils s'exécutent, car ils le font à l'intérieur d'une enveloppe logicielle spéciale appelée Machine Virtuelle Java, ou plus simplement Java.
- Java permet de traduire facilement tes programmes (écrans, menus et messages) en différentes langues.
- Java est un langage de programmation orienté objet, fortement typé, disposant d'un grand nombre de paquetages facilitant la vie aux programmeurs.
- Java est gratuit.

Né par hasard, java répond à la préoccupation des informaticiens qui cherchaient depuis les années 80, le moyen de s'affranchir des différentes plates-formes matérielles. Le C et le C++ sont très fortement dépendants du type de processeur, de l'infrastructure matérielle et du système d'exploitation. En général, pour qu'un programme fonctionne sur des systèmes différents, il faut le compiler dans chacun de ces environnements, et parfois, faire des modifications partielles de code.

Java a donné une réponse simple à cette préoccupation. Le langage Java se contente alors de :

- proposer des API universelles pour régler les problèmes "classiques" d'un langage de programmation
- fournir un compilateur en "pseudo-code" universel.

Les éditeurs de système d'exploitation devront quant à eux fournir la "machine virtuelle" qui permet d'exécuter ce "pseudo-code" tenant compte de l'environnement local.

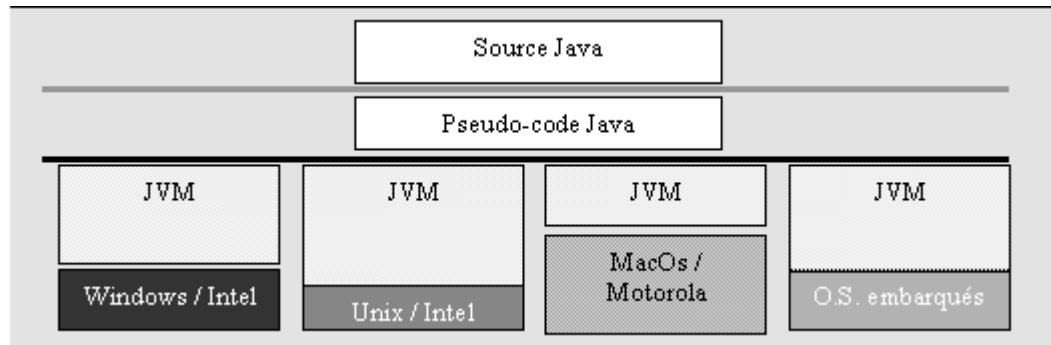


Fig 1 : Illustration du fonctionnement d'un programme java

Java dispose d'une pile de paquets qui facilitent la vie des programmeurs.

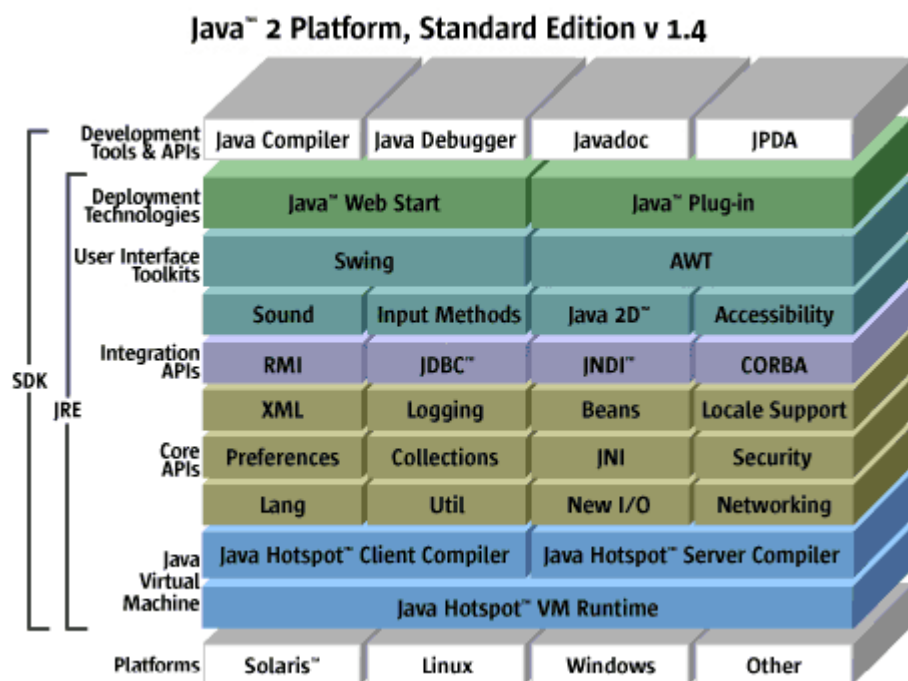


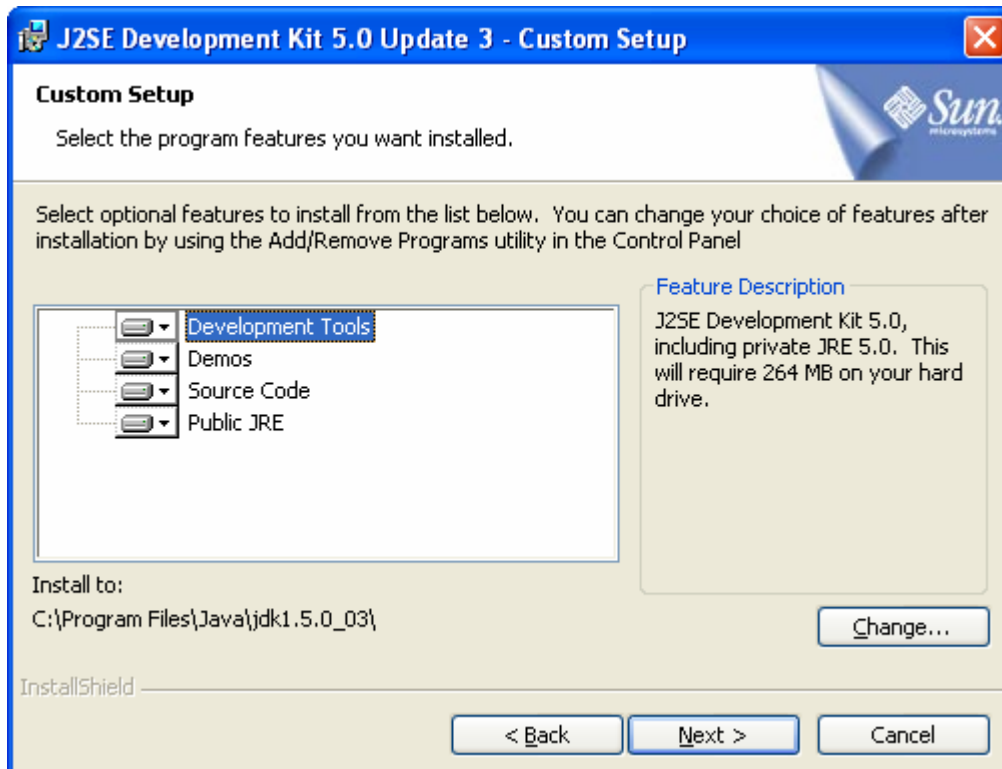
Fig. 2 : plateforme java 2 Edition standard

I. Installation de Java

1. Télécharger java 2 Software Development Kit (J2SDK).

<http://java.sun.com/j2se> Sélectionne la version (release) **J2SE 1.5.0** ou la plus récente.

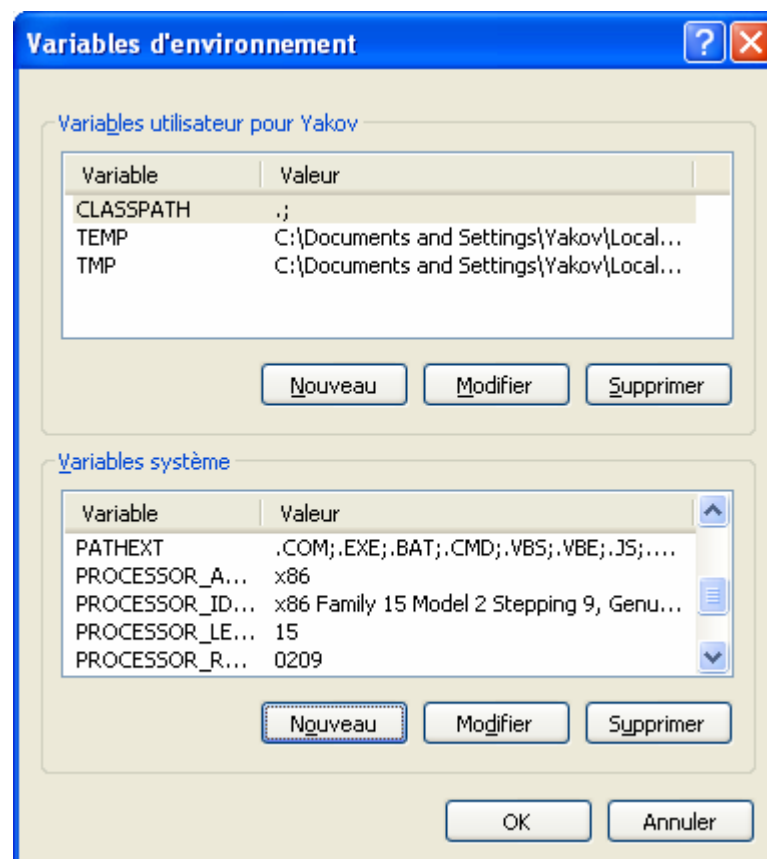
2. Lancer l'exécution du setup d'installation




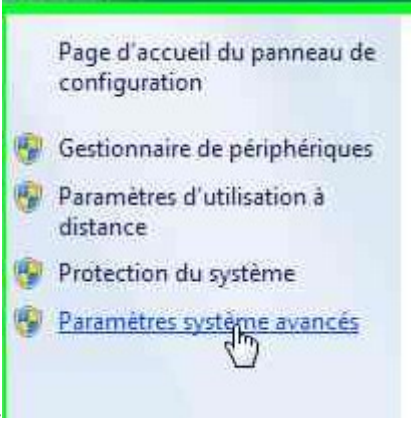
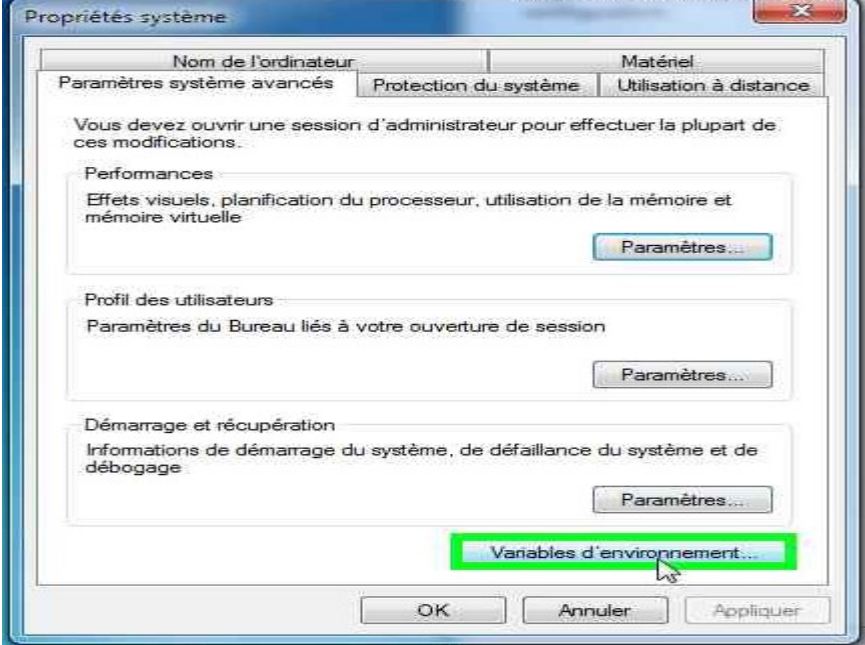
Utiliser l'assistant d'installation pour terminer l'installation.

3. Définir sous Windows les variables d'environnement **path** et **classpath**

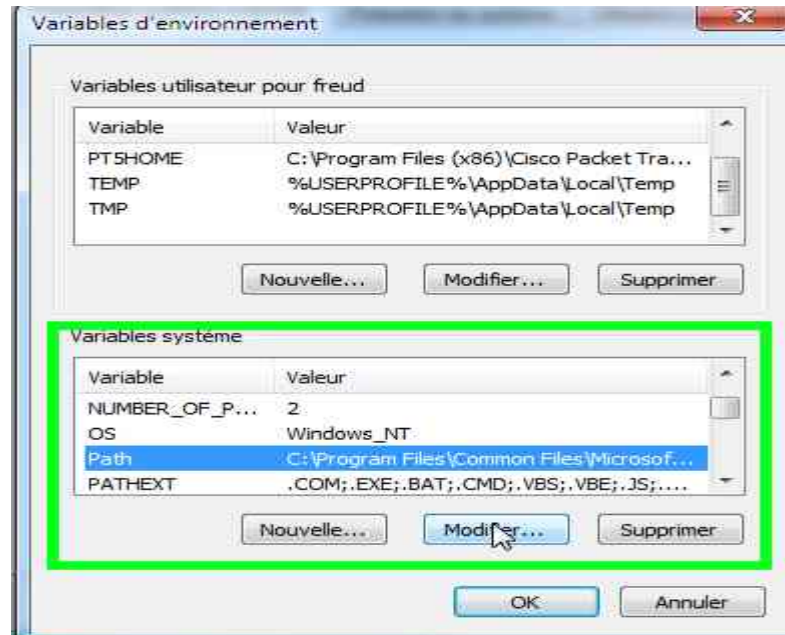
Si vous utiliser Windows XP la procédure est la suivante :



Sous Windows Vista et 7 la procédure est la suivante

1. 
2. 
3. 

4.



II. Les bases du langage Java

1. Les types de données prédéfinis

Type	Taille(octet)	Valeurs possibles
char	2	Caractères Unicode
int	4	$[-2^{31}, 2^{31}-1]$
long	8	$[-2^{63}, 2^{63}-1]$
byte	1	$[-2^7, 2^7-1]$
short	2	$[-2^{15}, 2^{15}-1]$
float	4 octets	$[3.4 \cdot 10^{-38}, 3.4 \cdot 10^{+38}]$ en valeur absolue
double	8 octets	$[1.7 \cdot 10^{-38}, 1.7 \cdot 10^{+38}]$ en valeur absolue
boolean	1 bit	true, false
String		Dépend du nombre de caractères

Date		Date
void		

2. Les opérateurs

On distingue les opérateurs arithmétiques, les opérateurs relationnels et les opérateurs logiques

- Les opérateurs arithmétiques

Opérateur	Désignation
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo

Il existe diverses fonctions mathématiques courantes dont les signatures se présentent comme suit :

double sqrt(double x) pour effectuer la racine carrée d'un réel

double cos(double x) pour calculer le Cosinus d'un réel

double sin(double x)) pour calculer le sinus d'un réel

double tan(double x)) pour calculer la Tangente d'un réel

double pow(double x, double y) pour calculer la valeur d'un réel x à la puissance y (x>0)

double exp(double x) pour calculer la valeur Exponentielle d'un réel

double log(double x) pour calculer le Logarithme népérien d'un réel

double abs(double x) pour calculer la valeur absolue d'un réel

- Les opérateurs logiques

- && : et
- || : ou
- ! : non

- Les opérateurs relationnels

- == Egalité.
- != différence
- < strictement inférieur.

- \leq inférieur ou égal.
- $>$ strictement supérieur.
- \geq supérieur ou égal.

- Les opérateurs d'affectation

L'opérateur d'affectation est =

En java, on utilise également les opérateurs d'affectation élargie. Dans ce cas l'opérateur d'affectation est précédé d'un des opérateurs arithmétiques déjà vu plus haut.

$+=$

$-=$

$*=$

$/=$

$\%=$

- Les opérateurs d'incrémentation et de décrémentation

++ Incrémentation

Cet opérateur, lorsqu'il est appliqué à une variable, augmente la valeur de cette dernière de 1. Mais il faut faire attention selon que l'opérateur est post fixé ou préfixé.

$a++$; /* équivaut à $a=a+1$; */

Dans une expression, l'opérateur d'incrémentation lorsqu'il est préfixé, devient prioritaire ; l'incrément s'effectue avant toute autre opération. Le contraire s'observe lorsque cet opérateur est post fixé.

Ainsi considérons la portion de code java suivante :

Cas 1

$a = 5$

$b = ++a$;

Après exécution, la valeur de $a = 6$ mais celle de $b = 6$

Cas 2

$a = 5$

$b = a++$;

Après exécution, la valeur de $a = 6$ mais celle de $b = 5$

- - décrémentation

Cet opérateur, lorsqu'il est appliqué à une variable, diminue la valeur de cette dernière de 1. Mais il faut faire attention selon que l'opérateur est post fixé ou préfixé (voir opérateur d'incrément).

3. La déclaration des données

Java est un langage sensible à la casse. Il faut donc faire attention lors de la déclaration et de l'utilisation des constantes et des variables.

- Déclaration des constantes

La syntaxe de déclaration d'une constante est la suivante :

final type nom=valeur;

Exemple: **final float** PI=3.141592F;

- Déclaration des variables

La syntaxe de déclaration d'une ou plusieurs variables est :

identificateur_de_type variable1,variable2,...,variablen;

où identificateur_de_type est un type prédéfini ou un type défini par le programmeur.

Exemple : **int** a ;

double e, f ;

on peut également initialiser une variable lors de sa déclaration.

int a = 6 ;

4. Déclaration des tableaux de données

Un tableau Java est un objet permettant de rassembler sous un même identificateur des données de même type. Sa déclaration est la suivante :

type nom_Tableau[]=new type[n] ; ou type[] nom_Tableau=new type[n] ;

NB : les indices des tableaux en java, commencent à partir de 0

Un tableau à deux dimensions pourra être déclaré comme suit :

type nom_Tableau[][]=new type[n][p] ; ou type[][] nom_Tableau=new type[n][p] ;

exemple : **int** mon_tableau[]= new int[5] ;

Un tableau est un objet possédant l'attribut length : c'est la taille du tableau.

5. Les instructions élémentaires de Java

- Instruction d'écriture

La syntaxe suivante est celle de l'instruction d'écriture sur une sortie standard (écran):

System.out.println(expression) ;

où expression est tout type de donnée qui puisse être converti en chaîne de caractères.

Exemple

System.out.println("bonjour les amis") ;

System.out.println(taux[1].length);

System.out.println(taux[1][1]);

Il est possible d'afficher une constante chaîne de caractère et une variable. Dans ce cas, la constante et la variable sont séparées par l'opérateur +.

System.out.println("la valeur finale est :" + a) ;

- Instruction de lecture

Le flux de données provenant du clavier est désigné par l'objet System.in de type InputStream. Ce type d'objets permet de lire des données caractère par caractère. C'est au programmeur de retrouver ensuite dans ce flux de caractères les informations qui l'intéressent. Le type InputStream ne permet pas de lire d'un seul coup une ligne de texte. Le type BufferedReader le permet avec la

méthode `readLine`. si vous ne comprenez pas grand-chose c'est bien normale. Vous verrez plus clair dès que nous allons définir et commencer à utiliser les objets.

Syntaxe :

```
BufferedReader IN=new BufferedReader(new InputStreamReader(System.in));
```

A chaque fois qu'une méthode est susceptible de générer une exception, le compilateur Java exige qu'elle soit gérée par le programmeur. Aussi, pour créer le flux d'entrée précédent, il faudra gérer les exception (erreurs éventuelles) :

```
BufferedReader IN=null;
```

```
try{
```

```
IN=new BufferedReader(new InputStreamReader(System.in));
```

```
} catch (Exception e){
```

```
System.err.println("Erreur " +e);
```

```
System.exit(1);
```

```
}
```

Une fois l'objet `IN`, créé la lecture proprement dite se fera à l'aide de l'instruction `readLine`

```
String ligne;
```

```
ligne=IN.readLine();
```

Comme vous pouvez le constater, le résultat de la lecture est mis dans une variable de type `String` quit à la convertir en numérique par la suite si le besoin se fait sentir. Dans ce cas les fonctions dont les signatures sont les suivantes sont utilisées :

`int parseInt(String)` pour convertir la chaîne de caractère en entier

`double parseDouble(String)` pour convertir la chaîne de caractère en double

`float parseFloat(String)` pour convertir la chaîne de caractère en float

III. Les structures de contrôle

1. Les instructions alternatives

Structure de choix simple

syntaxe :

if (condition)

 {actions_condition_vraie;}

else

 {actions_condition_fausse;}

exemple

if (x>0) {

 x++;

}

else

 x--;

Structure de cas ou de choix multiples

switch(expression) {

 case v1:

 actions1;

 break;

 case v2:

 actions2;

 break;

 default: actions_sinon;

```
}
```

2. Structure de répétition

- **La boucle for**

Syntaxe

```
for (initialisation; condition; instruction){
    actions;
}
```

Initialisation : instruction qui permet d'initialiser le compteur de la boucle

Condition : expression booléenne qui donne la condition d'arrêt de la boucle

Instruction : instruction qui définit le pas d'incrémentation ou de décrémentation du compteur

Notes

- les 3 arguments du for sont à l'intérieur d'une parenthèse.
- les 3 arguments du for sont séparés par des points-virgules.
- chaque action du for est terminée par un point-virgule.
- l'accolade n'est nécessaire que s'il y a plus d'une action.
- l'accolade n'est pas suivie de point-virgule.
- Chacun des trois arguments est optionnel

Exemple

```
For (i=0 ; i<10 ; i++){
    t[i]= 1;
}
```

Cette instruction permet d'initialiser les éléments d'un tableau de 10 éléments à 1.

- **la boucle Tantque**

Nombre de répétitions inconnu

```
while(condition){
```

```
actions;

}
```

Cette boucle est réputée passe partout car elle peut être adaptée à chacune des situations étudiées pour les autres boucles.

- **La boucle répéter jusqu'à (do while)**

La syntaxe est la suivante :

```
do{

instructions;

}while(condition);
```

On boucle jusqu'à ce que la condition devienne fausse ou tant que la condition est vraie. Le programmeur doit faire attention en utilisant cette la boucle car elle s'exécute toujours au moins une fois même si la condition est fausse au départ.

Remarque :

Sous java, le commentaire se présente sous trois formes ;

- i. // mon commentaire
- ii. /* mes commentaires
 Suite commentaire
 Suite commentaire */
- iii. /** commentaires
 Suite commentaire
 Suite commentaire
 */

IV. La structure d'un programme Java

Un programme Java n'utilisant pas de classe définie par l'utilisateur ni de fonctions autres que la fonction principale main pourra avoir la structure suivante :

```
public class test1 {

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

... code du programme

```
// main
```

```
// class
```

La fonction main appelée aussi méthode est exécutée la première lors de l'exécution d'un programme Java. Elle doit avoir obligatoirement la signature précédente :

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

```
ou public static void main(String[] arg){
```

V. Utiliser l'IDE Eclipse pour programmer en Java

Eclipse est l'un des IDE les plus utilisés par la communauté des programmeurs Java. Après le téléchargement de l'archive yyyyyyyyy puis décompresser la dans un répertoire. On n'a pas besoin d'une procédure particulière d'installation. Dans le répertoire de décompression, retrouver l'exécutable de Eclipse puis créer un raccourci si nécessaire.

1. Présentation rapide de l'interface d'Eclipse

Après le lancement de Eclipse la barre de navigation présente l'aspect suivant :



1 : "nouveau " général . Cliquer sur ce bouton revient à faire "Fichier - Nouveau "

2 : enregistrer. Revient à faire CTRL + S.

3 : imprimer.

4 : exécuter la classe ou le projet spécifié. Nous verrons ceci plus en détail.

5 : créer un nouveau projet Java. Revient à faire "Fichier - Nouveau - Java project ".

6 : créer une nouvelle classe dans un projet . Revient à faire "Fichier - Nouveau - Classe".

2. Création d'un nouveau projet

New Java Project

Create a Java project
Create a Java project in the workspace or in an external location.

Project name:

Contents

- ☒ Create new project in workspace
- ☐ Create project from existing source

Directory:

JRE

- ☒ Use default JRE (Currently 'jre1.6.0_03') [Configure default...](#)
- ☐ Use a project specific JRE:
- ☐ Use an execution environment JRE:

Project layout

- ☐ Use project folder as root for sources and class files
- ☒ Create separate folders for sources and class files [Configure default...](#)

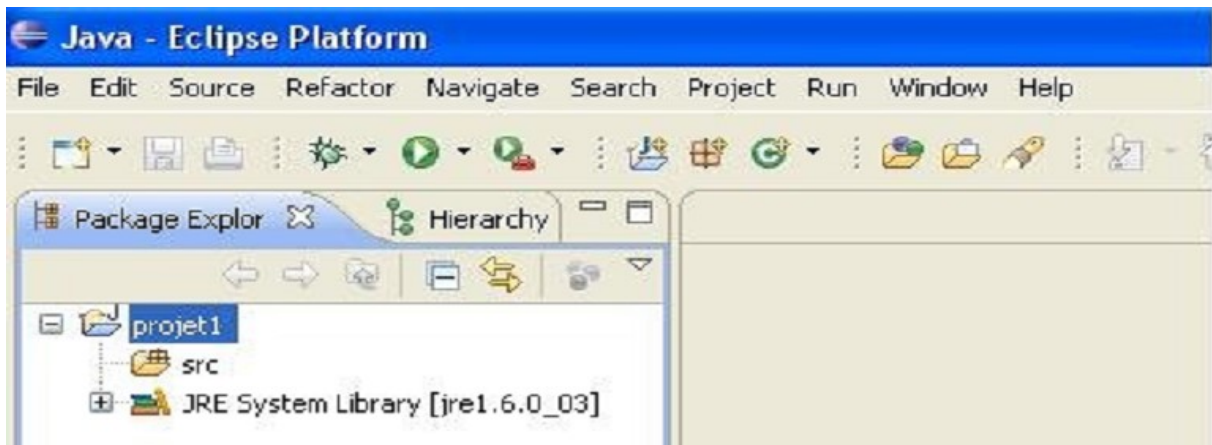
Working sets

☐ Add project to working sets

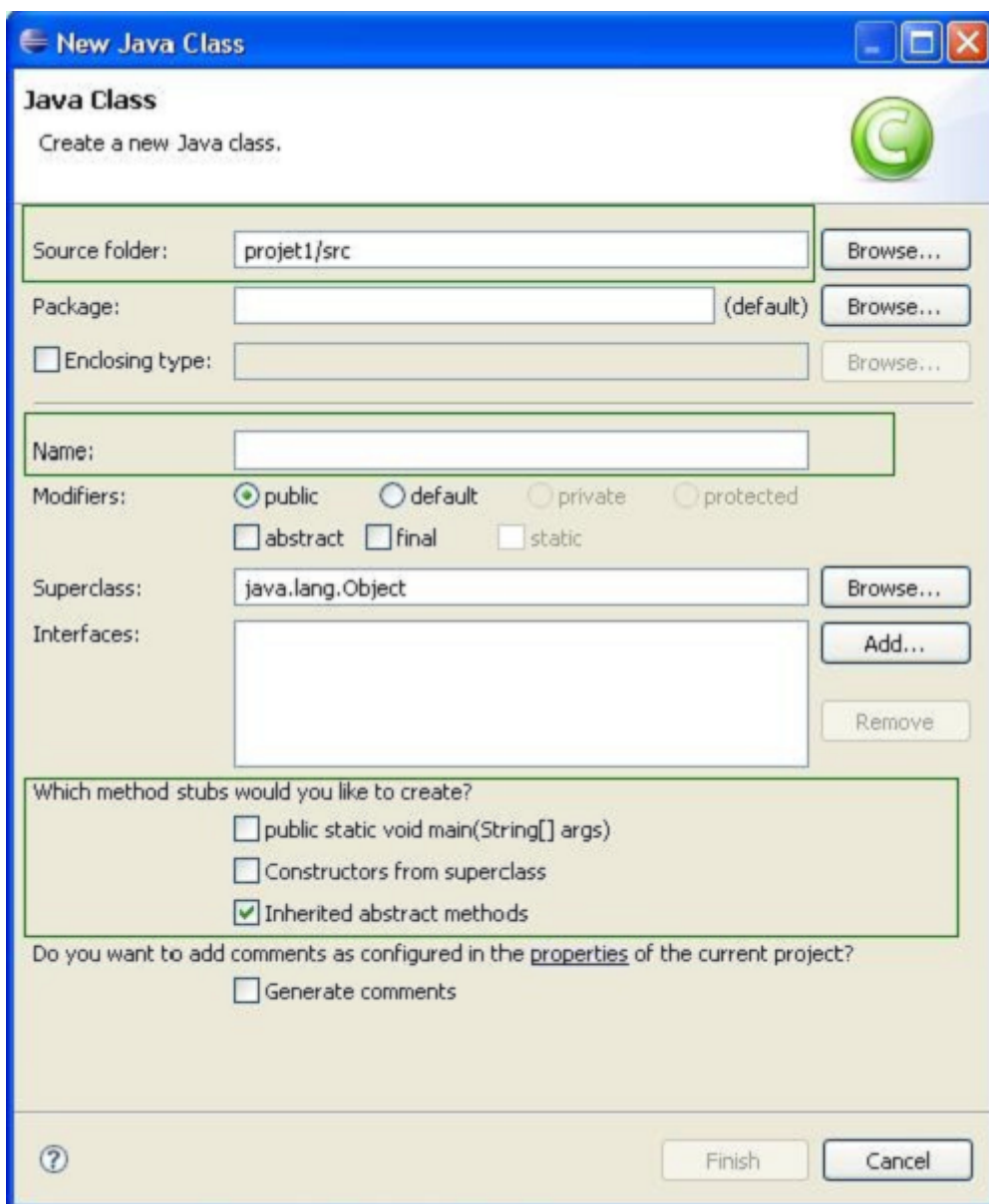
Working sets:

Après avoir renseigné le nom de votre projet (encadré 1), précisez si nécessaire l'emplacement où sera enregistré ce projet (encadré 2). Dans le cas où vous en auriez plusieurs, vous pouvez aussi

spécifier à Eclipse quel JRE utiliser pour ce projet.



Ajoutons maintenant une nouvelle classe au projet.



Dans l'encadré 1, nous pouvons voir où seront enregistrés nos fichiers Java.

Dans l'encadré 2, nommez votre classe Java. Ici nous avons choisi le nom Test. Attention ! le nom d'une classe doit toujours commencer par une lettre majuscule. Cette règle de programmation internationalement reconnu facilite la lisibilité des programme et le partage des ressources. Dans l'encadré 3, cochez "public static void main(String [] args) " si vous souhaitez intégrer une méthode main, puis cliquez sur "finish

CHAPITRE 2 LES CLASSES ET LES OBJETS

I. Quelques concepts fondamentaux de la programmation orienté objet

- **Classe**

« Type de données abstrait » qui décrit des objets ayant des propriétés communes. On entend par propriété l'ensemble des attributs et méthodes. Une classe peut être comparée à un moule, qui permet de créer les objets de la classe.

- **Objet**

Si la classe présente un aspect abstrait, l'objet est un élément concret de la classe. C'est le gâteau qui sort de notre moule. Le moule est la classe et le gâteau est l'objet manipulable.

- **Attribut**

C'est une propriété qui décrit l'état d'un objet.

- **Méthode**

Ensemble d'instructions qui décrivent le comportement d'un objet. Pour faire simple, c'est une fonction liée à une famille d'objet.

- **Constructeur**

C'est une méthode particulière qui est appelée lors de la création d'un objet (instanciation) de la Classe; il sert généralement à 'initialiser' les attributs de l'objet.

- **Héritage**

C'est un mécanisme qui permet à une classe (classe héritière), de bénéficier des propriétés d'une autre classe (classe de base). L'héritage est un concept très important de la programmation orientée objet car elle permet la réutilisation de ressources existantes.

- **Polymorphisme**

Il représente la faculté d'une même opération de s'exécuter différemment suivant le concept de la classe où elle se trouve. Ainsi une opération définie dans une superclasse peut s'exécuter de façon différente selon la sous-classe où elle est héritée.

- **Encapsulation**

C'est le principe qui permet de sécuriser les propriétés d'une classe. Elle se manifeste à travers les modificateurs :

private : la propriété ne sera accessible que par les seules méthodes internes de la classe

public : la propriété est accessible par toute fonction qu'elle appartiennent ou non à la classe.

protected : n'est accessible que par les seules méthodes internes de la classe ou à partir d'une classe dérivée (voir ultérieurement le concept d'héritage).

II. Introduction à la programmation orientée objet

1. Création d'une classe

Nous abordons maintenant, par l'exemple, la programmation objet en Java.

Syntaxe :

```
class nom_de_la_classe {  
    modificateur attribut1 ;  
    modificateur attribut2 ;  
    .  
    .  
    modificateur attribut_n ;  
    modificateur méthode1 ;  
    modificateur méthode2 ;  
    .  
}
```

```

    .
    modificateur méthode1 ;
}

```

Exemple : soit la classe Ville caractérisée par :

- un nom sous forme de chaîne de caractères,
- un nombre d'habitants,
- un pays apparenté de type chaîne de caractères.

```

class Ville{
    String nom ;
    int nbre_habitant ;
    String pays ;
    public Ville(){
        nbre_habitant=0 ;
    }

}

```

Le constructeur ici est la méthode Ville(). Ce constructeur est sans paramètre. Généralement le programmeur va prévoir un constructeur avec paramètre pour permettre d'initialiser les attributs lors de la création d'un objet.

On peut proposer une autre définition de la classe Ville :

```

class Ville{
    String nom ;
    int nbre_habitant ;
    String pays ;
    public Ville(String n , int h, String p){
        nom = n;
        nbre_habitant=0 ;
        pays = p ;
    }

}

```

Remarque : Une classe peut avoir plusieurs constructeurs.

2. Création d'un objet de la classe

Une instance d'objet se fait grâce au mot-clé new:

```
Ville une_ville = new Ville() ;
```

L'instruction précédente est correcte si l'on utilise la première version de la classe Ville. Pour la seconde version de la classe Ville on écrira :

```
Ville une_ville = new Ville("cotonou", 500000, "Bénin") ;
```

Comme pour n'importe quelle fonction, il faut respecter scrupuleusement l'ordre des paramètres passés lors de la création de votre objet. Pour pouvoir accéder aux données privées ou protégées de nos objets, nous allons utiliser des méthodes qualifiées d'**ACCESSEURS** et pour modifier les données privées ou protégées, on fait appelle également aux méthodes qualifiées de **MUTATEURS**.

Pour accéder aux données de l'objet une_ville par exemple on complètera la définition de la classe Ville par les méthodes :

```
public String getNom(){  
  
    return nom ;  
  
}  
  
public int getHabitant(){  
  
    return nbre_habitant ;  
  
}  
  
public String getPays(){  
  
    return pays ;  
  
}
```

Pour modifier le nombre d'habitant par exemple on peut utiliser la méthode :

```
public void modif_habitant(int nouvelle_valeur){

    nbre_habitant= nouvelle_valeur ;

}
```

Ce qu'il faut retenir

- Une classe permet de définir des objets. Ceux-ci ont des attributs et des méthodes.
- Une classe, par défaut, est déclarée public.
- On instancie un nouvel objet grâce au mot clé **new**.
- **Le ou les constructeurs d'une classe doivent porter le même nom que la classe, et pas de type de retour.**
- Dans une classe, on accède aux variables de celle-ci grâce au mot clé **this**.
- **L'ordre des paramètres passé dans le constructeur doit être respecté.**

III. Les packages, l'héritage, le polymorphisme, la surcharge

1. Les packages

Les packages sont des groupes de classes basées sur une organisation de répertoires. Un paquet correspond à un sous-répertoire. Dans le nom du paquet, les / sont remplacé par des '.'.

Ex : java.lang ou java.io ou java.swing

Lorsqu'une classe est dans un paquet_n son nom complet est de la forme : nom.du.paquet.LaClasse;

ex : java.lang.Math

Pour éviter de répéter le nom du paquet d'une classe, on peut utiliser les déclarations :

- a. pour une classee : `imports nom.du.paquet.LaClasse ;`
- b. pour toutes les classes : `nom.du.paquet.* ;`

2. L'héritage

Grâce à elle, nous pourrons créer des classes héritées (appelées aussi classes dérivées) de nos classes mères (appelées aussi classes de base). Nous pourrons créer autant de classes dérivées, par rapport à notre classe de base, que nous le souhaitons. Nous pourrons aussi nous servir d'une classe dérivée comme d'une classe de base pour créer une autre classe dérivée.

```
class Capitale extends Ville {
}
```

C'est le mot-clé **extends** qui informe notre application que la classe Capitale est héritée de Ville. Vous n'aurez donc **JAMAIS** ce genre de classe :

```
class Toto extends Titi, Tutu{
}
```

Nous devons créer un constructeur par défaut et un constructeur d'initialisation pour notre objet Capitale. Pour faire appel aux variables de la classe mère on utilise le mot-clé **super**. Ce qui aura pour effet de récupérer les éléments de l'objet de base, et de les envoyer à notre objet hérité.

```
class Capitale extends Ville {
    private String president;

    /**
     *Constructeur par défaut
     */
    public Capitale() {
        //Ce mot clé appelle le constructeur de la classe mère.
        super();
        president = "aucun";
    }
}
```

3. Le polymorphisme

Ce concept complète parfaitement celui de l'héritage. Quand une classe hérite d'une autre classe, elle hérite des méthodes. On peut redéfinir, substituer une méthode de la classe parente par une méthode de même nom dans la classe enfant.

4. La surcharge

C'est la possibilité de définir plusieurs méthodes de même nom mais possédant des paramètres différents (en nombre et/ou en type).

Ce qu'il faut retenir

- Une classe hérite d'une autre classe par le biais du mot clé `extends`.
- **Une classe ne peut hériter que d'une seule et unique classe en JAVA!**
- Si aucun constructeur n'est défini dans une classe fille, **la JVM en créera un et appellera automatiquement le constructeur de la classe mère.**
- La classe fille hérite de toutes les propriétés et méthodes **public** et **protected** de la classe mère.
- Les méthodes et propriétés **private** d'une classe mère ne sont pas accessibles dans la classe fille.
- On peut redéfinir (changer tout le code) d'une méthode héritée.
- On peut utiliser le polymorphisme sur une méthode par le biais du mot clé **super**.
- Le polymorphisme permet, entre autres, d'ajouter des traitements spécifiques à une méthode en continuant d'utiliser les traitements déjà définis (utilisation de `super`).
- Grâce à l'héritage et au polymorphisme, nous pouvons utiliser la covariance des variables

IV. Interface et implémentation

Une interface est un ensemble de prototypes de méthodes ou de propriétés qui forme un contrat.

Une classe qui décide d'implémenter une interface s'engage à fournir une implémentation de toutes les méthodes définies dans l'interface. C'est le compilateur qui vérifie cette implémentation.

```
public interface I{
    public void A();
    public String B();
}
```

Et pour faire en sorte qu'une classe utilise une interface, il suffit d'utiliser le mot clé **implements**


```

public class X implements I{
    public void A(){
        //.....
    }

    public String B(){
        //.....
    }
}

```

Vous pouvez implémenter plusieurs interfaces

```

public class X implements I, I2{
    public void A(){
        //.....
    }

    public String B(){
        //.....
    }

    public void C(){
        //.....
    }

    public String D(){
        //.....
    }
}

```

Ce qu'il faut retenir

- Une interface est une **classe 100 % abstraite, qui ne peut être instanciée.**
- Toutes les méthodes d'une interface n'ont pas de corps.
- Une interface sert à définir un **super type** et à **utiliser le polymorphisme.**
- Une interface s'implémente dans une classe en utilisant le mot clé **implements.**
- Vous pouvez implémenter autant d'interfaces que vous voulez dans vos classes.

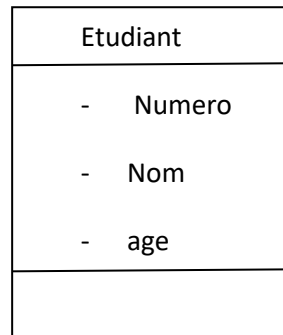
EXERCICES : les bases du langage java

- 1- Ecrire un programme java pour comparer deux entiers saisis au clavier(utiliser la classe Scanner pour la lecture des entiers au clavier).
- 2- Ecrire un programme java pour afficher la table de multiplication d'un nombre saisi au clavier.

- 3- Soient deux entiers A et B. on donne la formule $A \times B = A + A + A + \dots + A$ (B fois). Ecrire un programme java pour effectuer le produit de deux entiers en utilisant la formule énoncée plus haut.

EXERCICES : les classes et les objets

Soit la classe dont la représentation sous forme UML est la suivante :



- 1- Créez la classe Etudiant et prévoyant les constructeurs nécessaires ainsi que les accesseurs (get_) et les mutateurs (set_).
- 2- Dans une classe utilisatrice contenant la méthode main, créer deux objets de la classe Etudiant. Les données pour le faire seront saisies au clavier.
- 3- Affichez le nom de l'étudiant le plus âgé des deux.