المدرسة العليا للتكنولوجيا - الداخلة
ⵜⵜⵉⵛⵏ ⵜⴰⵙⴷⴰⵡⵜ ⵏ ⵜⵜⴽⵏⵓⵍⵓⵊⵢⴰ - ⴷⴰⵅⵍⴰ
ÉCOLE SUPÉRIEURE DE TECHNOLOGIE - DAKHLA

Programmation objet avancée

2ème année GI

Pr. Naoual Nassiri

N.Nassiri@uiz.ac.ma

Année universitaire 2023-2024

Introduction

- La Programmation Orientée Objet (POO) propose une méthodologie de programmation centrée sur les objets. Son principe consiste:
  - À identifier, dans un premier temps, les objets qui doivent être utilisés (ou manipulés) par le programme: on commence par décider quels objets doivent être inclus dans le programme.
    - On ne commence pas par ce que doit faire l'application mais plutôt par ce qu'elle doit manipuler.
  - À définir ensuite, les traitements : pour chaque objet définir les traitements associés.

En résumé : Le principe de la POO consiste à :

- Déterminer les objets présents dans le programme et identifier leurs données.
- Définir les traitements à faire sur ses objets.

Introduction

Définition : un objet est une entité regroupant un ensemble de propriétés (structures de données) et de traitements associés (opérations). Donc un objet regroupe dans la même entité informatique les structures de données et les moyens de traitement de ces données.

- Les structures de données définies dans l'objet sont appelés attributs (on les appelle aussi champs ou propriétés).
- Les opérations définies dans l'objet sont appelés méthodes : c'est un ensemble de procédures et/ou de fonctions.
- Les attributs et méthodes d'un objet sont appelés ses membres.
- L'ensemble des valeurs des attributs d'un objet, à un instant donné, est appelé état interne.

Introduction

Exemple : Un objet de type « Etudiant » est conçu pour modéliser (pour représenter) un étudiant. Il est caractérisé par:

- Les attributs (par exemple : le nom, le numéro d'inscription, les notes, etc…).
- les traitements associés (les méthodes de traitement associées): par exemple les méthodes pour le calcul de la moyenne, l'affichage des résultats, etc….

Introduction

Définition : Une classe (ou type d'objets) définit la structure d'un objet, i.e., une classe déclare l'ensemble des membres (attributs et méthodes) qui composeront un objet. Elle représente les objets similaires qui ont :

- La même structure de données (les mêmes attributs).
- Le même comportement (les mêmes méthodes).

Introduction

Propriétés : Les classes servent pour la création des objets.

- Un objet est une **instance d'une classe**. C'est une variable du type défini par la classe.
- Un programme orienté objet est constitué de classes qui permettent de créer des objets qui s'envoient des messages.
- L'ensemble des interactions entre les objets défini un algorithme.
- Les relations entre les classes reflètent la décomposition du programme.

Principe d'encapsulation des données

- Le principe d'encapsulation au sens strict consiste à cacher tous les attributs : cela veut dire que les attributs sont invisibles depuis l'extérieur de la classe.

- Par conséquent un objet est vu par le reste du programme comme une boite noire. Ses attributs ne peuvent être accédés que par l'intermédiaire de ses méthodes ; par exemple, par l'intermédiaire d'une méthode de lecture et d'une méthode d'écriture.

- Par contre les méthodes sont visibles de l'extérieur de la classe.

Principe d'encapsulation des données

Définition : L'ensemble des méthodes publics d'un objet (les méthodes accessibles depuis l'extérieur) sont appelées interface. Elles caractérisent le comportement de l'objet.

- En java, il est possible de n'encapsuler que certains attributs (ne pas suivre le principe d'encapsulation strict) et ceci grâce aux mots clés « private » et « public » pour préciser si un attribut est visible (public) depuis l'extérieur de la classe ou invisible (private) depuis l'extérieur de la classe.

Introduction à l'environnement de la programmation sous Java

- Java est un langage de programmation orienté objet créé en 1991 par « SUN Microsystem : c'est une société qui est rachetée par Oracle en avril 2009 ». L'objectif initial était la programmation de petits appareils comme des télécommandes, la programmation d'applications Web.
- Il est devenu actuellement l'un des langages de programmation généraliste les plus utilisés. Le langage Java a connu beaucoup d'évolution depuis sa création. Plusieurs versions ont été développées: la première version « Java 1.0 » en 1996 …..
- Plusieurs environnements java sont disponibles:
  - Java SE (Standard Edition)
  - Java ME (Mobile Edition)
  - Java EE (Enterprise Edition)

Environnement Java SE

- Outil JDK : Outils de développement JDK (Java Developpment Kit) de SUN (voir site : http://java.sun.com).
  - Il est gratuit.
  - Il contient un ensemble de paquetages très riches et très variés.
  - Il contient un ensemble d'outils : le compilateur : « javac », la JVM (Java Virtual Machine) « java », le générateur de documentation.
- Des environnements de développements gratuits sont disponibles, par exemple :
  - NetBeans : http://www.netbeans.org/
  - Eclipse : http://www.eclipse.org/

Quelques caractéristiques

- Java est un langage purement orienté objet. En effet, l'entité de base de tout code Java est la classe : c'est-à-dire qu'en Java, tout se trouve dans une classe, il ne peut y avoir de déclarations ou de code en dehors du corps d'une classe.
- La syntaxe du langage Java est très proche du langage C++ (et par conséquent du langage C). Ci après on donne une simple comparaison entre ces deux langages :
  - **Création des langages:**
    - C++ est crée dans les laboratoires « AT&T Bell » en 1983.
    - Java est crée dans les laboratoires de « Sun Microsystems » en 1991.

Quelques caractéristiques

- **Java évite les points "critiques" du langage C++:**
  - Les pointeurs
  - La surcharge des opérateurs
  - L'héritage multiple
- **Gestion de la mémoire :**
  - En java, la libération de la mémoire s'effectue de manière implicite : il existe un mécanisme de gestion automatique de la mémoire, connu sous le nom de ramasse miettes (en anglais Garbage Collector) qui est chargé de détecter les objets à détruire. Ceci permet un gain de fiabilité au détriment d'une perte en rapidité par rapport à C++.

Quelques caractéristiques

- **Portabilité :**
  - Le compilateur C++ produit un exécutable qui dépend de l'environnement de travail (processeur, système d'exploitation, etc.) où le code source est compilé. Par conséquent, à priori, pour une exécution sur un autre environnement, on doit donc créer des exécutables propres à chaque type d'architecture sur laquelle on veut exécuter le programme.

Quelques caractéristiques

- **Portabilité :**
  - En Java, la compilation ne traduit pas directement le programme source dans le code natif de l'ordinateur. Le code source est d'abord traduit dans un langage intermédiaire appelé "bytecode", langage d'une machine virtuelle (JVM – Java Virtual Machine) définie par Sun.
  - Le bytecode, généré par le compilateur, ne dépend pas de l'environnement de travail ou le code source est compilé. Au moment de l'exécution, il sera traduit dans le langage machine relatif à la machine sur laquelle il sera exécuté.

Compilation

- Le nom du fichier source à compiler doit être identique au nom de la classe auquel on rajoute l'extension « **.java** ». *Sun* fournit le compilateur « ***javac*** » avec le JDK. Par exemple, si « **NomDeClasse.java** » est le nom du fichier source à compiler, alors la commande :

javac **NomDeClasse.java**

compile la classe « **NomDeClasse** » dont le code source est situé dans le fichier « **NomDeClasse.java** ».

- Cette compilation crée un fichier nommé «**NomDeClasse.class**» qui contient le « *bytecode* ». C'est ce bytecode qui sera exécuté par une JVM. En générale, si un système possède une JVM alors il peut exécuter tous les bytecodes (fichiers .class) compilés sur n'importe quel autre système.

Compilation

Important : Si le fichier « **NomDeClasse.java** » fait référence à des classes, situées dans des répertoires différents du répertoire courant (répertoire de travail), alors il faut les spécifier:

- Soit pendant la compilation à l'aide de l'option « -classpath ».

Exemple : supposons qu'on fait références à des classes situées dans les répertoires « /prog/exemple et /cours », alors on doit les spécifier de la façon suivante:

- sous windows: javac -classpath /prog/exemple ; /cours; NomDeClasse.java
- sous Linux : javac -classpath /prog/exemple:/cours : NomDeClasse.java

- Soit dans la variable d'environnement « CLASSPATH ». Cette variable indique le chemin ou se trouve les classes (par défaut la recherche des classes se fait dans le répertoire courant).

Exécution du bytecode

- Le *bytecode est* exécuté par une JVM (simulée par un programme) qui :
  - ✓ lit les instructions (en *bytecode*) du programme **.class**,
  - ✓ les traduit dans le langage natif de la machine sur laquelle il sera exécuté.
  - ✓ lance l'exécution
- Sun fournit le programme « *java* » qui simule une JVM. Pour l'exécution, il suffit d'utiliser la commande: java NomDeClasse

Exécution du bytecode

Important :

- Il ne faut pas rajouter l'extension « **.class** »
- Si des classes d'autres répertoires sont nécessaires, alors :
  - ❑ Soit, il faut les spécifier à l'aide de l'option –classpath. Si par exemple on fait références à des classes situées dans les répertoires «/prog/exemple et /cours », alors on doit les spécifié pendant l'exécution de la façon suivante:
    - sous windows: java -classpath /prog/exemple; /cours NomDeClasse
    - sous Linux: java -classpath /prog/exemple:/cours: NomDeClasse
  - ❑ Soit, il faut les rajouter dans la variable d'environnement «CLASSPATH »

Un exemple de programme java

Considérons la classe « **MonPremProg** » définie ci-desous:

```
public class MonPremProg {
public static void main(String args[]) {
System.out.println(" Bonjour: mon premier programme Java " );
}
}
```

- Ce code doit être sauvegardé obligatoirement dans un fichier texte (fichier source) dont le nom est le nom de la classe avec l'extension «.java» : le nom de la classe est «**MonPremProg** » donc le fichier source sera nommé « **MonPremProg.java** »
- Cette classe est exécutable car elle posséder la méthode « *main()* » ayant la signature :

public static void main(String[] args).

Un exemple de programme java

NB: Pour compiler le programme précédent, il faut tout d'abord installer l'environnement de développement **JDK** (**J**ava **D**eveloppment **K**it).

Installation sous Windows

- Télécharger la version correspondante à votre architecture (32 ou 64 bits) à partir de : http://www.oracle.com/technetwork/java/javase/downloads
- Exécuter le fichier téléchargé et ajouter « C:\Program Files\Java\jdk…_xy\bin » au chemin, en modifiant la variable **path**. La valeur de **path** doit ressembler à ce qui suit : « C:\WINDOWS\system32;C:\WINDOWS;C:\Program Files\Java\Jdk…\bin »

Un exemple de programme java

- Pour compiler, on utilise la commande ***javac***:

  javac MonPremProg.java
- La compilation génère un fichier nommée « MonPremProg.class » c'est la bytecode.
- Pour l'exécution, on utilise la commande ***java***:

  java MonPremProg
- La machine virtuelle JVM interprète le *bytecode* de la méthode «*main()*» de la classe « MonPremProg ». L'exécution du programme «MonPremProg » affiche à l'écran la chaîne de caractères:

  Bonjour: mon premier programme Java
- Ceci grâce à l'instruction: System.out.println(" Bonjour: mon premier programme Java ");

Description du premier programme

System.out.println(" Bonjour: mon premier programme Java ") ;

- **System** : est une classe
- **out** : est un objet dans la classe System
- **println()** : est une méthode (fonction) dans l'objet out. Les méthodes sont toujours suivi de ().
- Les points séparent les classes, les objets et les méthodes.
- Chaque instruction doit se terminer par « ; »
- **Bonjour: mon premier programme Java** : est une chaîne de caractères.

Description du premier programme

De manière générale :

- Tout programme destiné à être exécuté, doit contenir une méthode particulière nommée « main() ». Elle contient le «programme principal» à exécuter. Elle est définie de la manière suivante:

```
public static void main(String args[]) {
/* corps de la méthode */
}
```

- Le paramètre args est un tableau d'objets de type String. Il est exigé par le compilateur Java.
- La méthode « main() » ne peut pas retourner d'entier comme en C.
- La classe contenant la méthode main() doit obligatoirement être public afin que la machine virtuelle y accède.
- Dans l'exemple précédent, le contenu de la classe « MonPremProg » est réduit à la définition d'une méthode main().

Description du premier programme

De manière générale :

- Un fichier source peut contenir plusieurs classes mais une seule doit être public (dans l'exemple c'est la classe: MonPremProg).
- Le nom du fichier source est identique au nom de la classe publique qu'il contient avec l'extension « .java ». Dans l'exemple précédent, le fichier source doit obligatoirement avoir le nom: « MonPremProg.java ».
- Dans Java, toutes les déclarations et les instructions doivent être faites à l'intérieure d'une classe.

public class **MonPremProg { … }**

veut dire que vous avez déclaré une classe qui s'appelle **MonPremProg**.

Les identifiants

Le nom d'une classe (**identifiant**) doit respecter les contraintes suivantes :

- L'identifiant doit commencer par une lettre (arabe, latin, ou autre), par _ ou par $. Le nom d'une classe ne peut pas commencer par un chiffre.
- Un identifiant ne doit contenir que des lettres, des chiffres, _, et $.
- Un identifiant ne peut être un mot réservé.
- Un identifiant ne peut être un des mots suivants : **true**, **false** ou **null** . Ce ne sont pas des mots réservés mais des types primitifs et ne peuvent pas être utilisés par conséquent.

Mots réservés

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Recommandations

En java, il est, par convention, souhaitable de commencer les noms des classes par une lettre majuscule et utiliser les majuscules au début des autres noms pour agrandir la lisibilité des programmes.

Exemples

Nom de la classe	description
Etudiant	Commence par une majuscule
PrixProduit	Commence par une majuscule et le le deuxième mot commence par une majus-cule
AnnéeScolaire2014	Commence par une majuscule et ne contient pas d'espace

Exemples non recommandées

Nom de la classe	description
etudiant	ne commence pas par une majuscule
ETUDIANT	le nom en entier est en majuscule
Prix_Produit	_ n'est pas utilisé pour indiqué un nouveau mot
annéescolaire2014	ne commence par une majuscule ainsi que le deuxième, ce qui le rend difficile à lire

Données primitifs

- Java est un langage (presque) purement orienté objets, puisqu'il permet la déclaration de types de données primitifs.
- Lors de la déclaration d'une variable sans qu'aucune valeur ne lui soit affecté, la variable sera non initialisée.
- Par exemple, lors de déclaration de la variable **age** de type **int** : «**int age;**» si vous essayez de l'utiliser dans une expression ou de l'afficher, vous allez recevoir une erreur lors de la compilation (contrairement au langage **C**) indiquant que la variable n'est pas initialisée (The local variable age may not have been initialized).

Types numériques

En java, il existe 4 types entiers et 2 types réels :

Type	Valeur Minimale	Valeur Maximale	Taille en octets
byte	−128	127	1
short	−32 768	32 767	2
int	−2 147 483 648	2 147 483 647	4
long	−9 223 372 036 854 775 808	9 223 372 036 854 775 807	8
float	$-3.4 * 10^{38}$	$3.4 * 10^{38}$	4
double	$-1.7 * 10^{308}$	$1.7 * 10^{308}$	8

Types numériques

Remarques :

1. Par défaut, une valeur comme 3.14 est de type **double**, donc pour l'affecter à une variable de type **float**, il faut la faire suivre par la lettre **f** (majuscule ou minuscule).

float pi=3.14F, min=10.1f;

2. Par défaut les entiers sont de type **int**. Pour affecter une valeur inférieure à **- 2 147 483 648** ou supérieure à **2 147 483 647** à un **long**, il faut la faire suivre par la lettre **l** (majuscule ou minuscule).

long test=4147483647L;

Caractères

On utilise le type **char** pour déclarer un caractère. Par exemple :

```
char c='a';
char etoile='*';
```

Les caractères sont codés en utilisant **l'unicode**. Ils occupent 2 octets en mémoire. Ils permettent de représenter **65 536** caractères, ce qui permet de représenter (presque) la plupart des symboles utilisés dans le monde.

Séquences d'échappement

Séquence d'échappement	Description
\n	nouvelle ligne (new line)
\r	retour à la ligne
\b	retour d'un espace à gauche
\\	\ (back slash)
\t	tabulation horizontale
\'	apostrophe
\"	guillemet

Type boolean

Il permet de représenter des variables qui contiennent les valeurs vrai (**true**) et faux (**false**).

```
double a=10, b=20;
boolean comp;
comp=a>b; //retourne false
comp=a<=b; //retourne true
```

Constantes

Pour déclarer une constante, il faut utiliser le mot clés **final**. Par convention, les constantes sont écrites en majuscule.

Exemple :

```
final int MAX = 100;
final double PI = 3.14;
...
final int MAX_2 = MAX * MAX;
```

Expressions et opérateurs

Comme pour le langage C, Java possède les opérateurs :

- Arithmétiques usuels (+, -, *, /, %)
- De comparaison (<, <=, >, >=, ==, !=).
- On retrouve aussi les expressions arithmétiques, les comparaisons et les boucles usuelles du langage C.
- D'autres façons propres au langage Java seront vues dans les chapitres suivants.

Exemple 1 :

```
public class Expressions {
    public static void main(String[] args) {
        double a=10, b=20, max, min;

        max = b;
        min = a;
        if (a > b) {
            max = a;
            min = b;
        }
        //Equivalent a printf du langage C
        System.out.printf("max = %f\tmin =%f\n", max
            , min);

    }
}
```

Exemple 2 :

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

        //Carrees des nombres impairs de 1 a 30
        for(int i=1; i<=30; i+=2)
            System.out.printf("%d^2 = %d\n",i,i*i);
    }
}
```

Exercices

Exercice 1

Parmi les identifiants suivants, quels sont ceux qui sont valides ?

a - nomEtudiant
b - prenom Etudiant
c - static
d - BONJOUR
e - 23code
f - code13
g - serie#
h - Test_Comp
i - goto

Exercices

Exercice 1 - Solution

Parmi les identifiants suivants, quels sont ceux qui sont valides ?

a - nomEtudiant
b - prenom Etudiant
c - static
d - BONJOUR
e - 23code
f - code13
g - serie#
h - Test_Comp
i - goto

Exercices

Exercice 2

Donnez le résultat des expressions suivantes :

a - 15 / 2

b - 15 % 2

c - 14 % 2

d - 31 % 7

e - 5 + 6 * 3

f - 25 + 3 / 2

Exercices

Exercice 2 -Solution

Donnez le résultat des expressions suivantes :

a - 15 / 2 (**7**)

b - 15 % 2 (**1**)

c - 14 % 2 (**0**)

d - 31 % 7 (**3**)

e - 5 + 6 * 3 (**23**)

f - 25 + 3 / 2 (**26**)

Exercices

Exercice 3

Donnez le résultat des expressions booléennes suivantes :

a - 4 <= 9

b - 12 >= 12

c - 3 + 4 == 8

d - 7 < 9 – 2

e - 5 != 5

f - 15 != 3 * 5

g - 9 != -9

h - 3 + 5 * 2 == 16

Exercices

Exercice 3-Solution

Donnez le résultat des expressions booléennes suivantes :

a) 4 <= 9 (true)
b) 12 >= 12 (true)
c) 3 + 4 == 8 (false)
d) 7 < 9 – 2 (false)
e) 5 != 5 (false)
f) 9 != -9 (true)
g) 3 + 5 * 2 == 16 (false)

Exercices

Exercice 4

Si j = 5 et k = 6, alors la valeur de j++ == k est :

a - 5

b - 6

c - true

d - false

Exercices

Exercice 4-Solution

Si j = 5 et k = 6, alors la valeur de j++ == k est :

a. 5
b. 6
c. true
d. False

False : j++ == k ⇔ j == k puis j + +

Exercices

Exercice 5

Quel est le résultat de sortie du code suivant ?

```
for ( int i = 0; i < 3; ++i )
        for ( int j = 0; j < 2; ++ j )
                System.out.print ( i+ " " + j + " " ) ;
```

a - 0 0 0 1 1 0 1 1 2 0 2 1
b - 0 1 0 2 0 3 1 1 1 2 1 3
c - 0 1 0 2 1 1 1 2
d - 0 0 0 1 0 2 1 0 1 1 1 2 2 0 2 1 2 2

Exercices

Exercice 5-Solution

Quel est le résultat de sortie du code suivant ?

```
for ( int i = 0; i < 3; ++i )
        for ( int j = 0; j < 2; ++ j )
                System.out.print ( i+ " " + j + " " ) ;
```

a) 0 0 0 1 1 0 1 1 2 0 2 1
b) 0 1 0 2 0 3 1 1 1 2 1 3
c) 0 1 0 2 1 1 1 2
d) 0 0 0 1 0 2 1 0 1 1 1 2 2 0 2 1 2 2