
TD 1 IJA

Introduction

Objet, méthode, attribut

Classe, instance

Règles de base du nommage

Mes premiers programmes

Introduction

- Langages de programmation
 - Interprétés
 - visual basic (VB), basic, shells (bash, ksh, tcsh, sh), perl, tcl-tk, python, ruby, java, c#, awk
 - ++ code peut tourner sans modifications sur toute machine ayant un interpréteur, ++ pas d'étape de compilation donc pas de gestion d'1 version par plate-forme cible, -- parfois un peu plus lent à l'exécution, -- chaque machine doit avoir un interpréteur, --qualité exécution fonction de qualité interpréteur
 - Compilés
 - C, C++, pascal
 - ++parfois plus rapide à l'exécution, ++/--code non modifiable, --doit gérer une version compilée par plate-forme cible
 - Java
 - compilé (en un langage intermédiaire unique) puis interprété
 - write once, run everywhere

Enfin mon premier programme

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

main(String [] arg)



HelloWorld.java

Compilateur

javac HelloWorld.java

HelloWorld.class

Machine Virtuelle

java HelloWorld

HelloWorld

L'environnement de travail

- Sous linux
 - un shell avec vi pour écrire les sources java
 - un shell pour compiler
 - un shell pour exécuter
 - les 3 shells positionnés dans le même répertoire
 - (on fera mieux dans les séances suivantes)
- 1 répertoire par projet /exo
 - 1 fichier source par classe
 - nom fichier = NomClasse.java
 - par conséquent, 1 fichier compilé par classe

Exo 1-1 : faire HelloWorld, compiler (debugger of course!), lancer

Objet

- Morceau de programme UNIFIE ayant

- des attributs

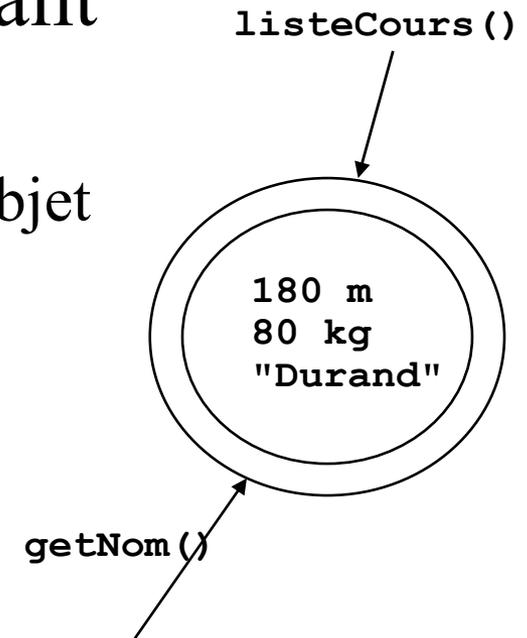
- Représentés par des valeurs données à l'objet
- Représentent un état de l'objet

- des méthodes

- qui permettent de manipuler ces attributs
- fonctions que l'objet sait faire
- Représentent le comportement de l'objet

- PremièreLettreEnMinusculePuisPremiereEnMajuscule

Exo 1-2 : définir sur papier plusieurs instances de différents cercles et différentes manières de les exprimer



Classe

- Usine de fabrication d'objets
- Permet de définir
 - le comportement des objets de cette classe
 - signature et code des méthodes
 - la structure de l'état des objets de cette classe
 - liste et type des attributs (mais pas leurs valeurs)
- PremièreLettreEnMajusculePourChaqueMot

Exo 1-3 : créer une classe Cercle vide. Compiler

Création d'un objet = instantiation

- Pour pouvoir **manipuler** un objet, on déclare une **référence** de type la classe de cet objet

```
Circle c; //c est une référence qui va me permettre de  
manipuler un objet de la classe Circle
```

- Pour **créer** un objet, on instancie une classe en appliquant l'opérateur **new**. Une nouvelle instance de cette classe est alors allouée en mémoire :

```
c = new Circle(); //je fabrique un objet Circle et je le  
référence par c
```

- `Circle()` est une méthode particulière appelée **constructeur** ; elle a exactement le même nom que la classe

Exo 1-4 : Utilisez la classe cercle à partir de la classe HelloWorld

Classe - exemple et exo

```
public class Circle {
    private double x, y, r;
    public void affiche(){
        System.out.println( "x=" +this.x+"y=" +this.y+ "r=" +this.r);
    }
    public double getPerimeter(){
        return (Math.PI*2*this.r);
    }
    public double getArea(){
        return (Math.PI*this.r*this.r);
    }
    public void initialiser(int x1, int y1, int r1){
        this.x=x1; this.y=y1;this.r=r1;
    }
}
```

Exo 1-5 : Ecrire et utiliser la classe cercle suivante

Constructeur

- Tout objet doit avoir ses valeurs positionnées.
 - Un constructeur peut permettre de fixer ces valeurs à la création de l'objet (attention les structures de données sont de toutes façons réservées en mémoire)
- Un constructeur est une méthode qui possède le même nom que la classe, mais qui ne renvoie rien
- Si le programmeur n'écrit pas de constructeur, le compilateur crée un constructeur par défaut, implicite (sans paramètre et avec corps vide).
- Une classe peut avoir plusieurs constructeurs qui diffèrent par le nombre et la nature de leurs paramètres.

Exo 1-6 : définir différents constructeurs pour la classe Cercle et supprimer la fonction d'initialisation. Compiler et exécuter

DANGER : Attribut != Variable

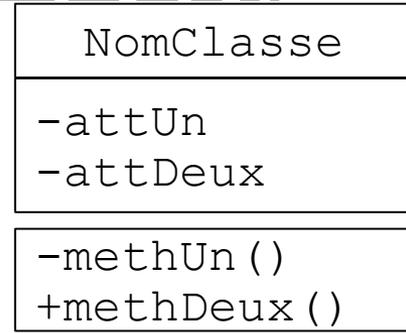
- Les attributs sont relatifs aux objets
 - leur liste est définie dans la classe
 - chaque objet a une valeur pour chaque attribut
 - l'objet s'en souvient tout le temps
 - il faut les utiliser avec “this” devant leur nom
- Les variables sont relatives à une méthode
 - Elles sont définies dans une méthode
 - leur valeur change d'un appel de la méthode à l'autre

```
//suite classe Circle  
void zoom(int r){  
    for(int i=1 ; i <= r ; i++){  
        r=r*2;  
    }  
}
```

Exo 1-7 : Intégrer cette fonction dans votre classe cercle que ce passe-t'il et pourquoi ?

Standard de nommage et de représentation graphique (UML)

- Classe
 - PremièreLettreEnMajusculePourChaqueMot
 - rectangle
- Attribut et méthode
 - premièreLettreEnMinusculePuisPremiereEnMajusculePourChaqueMot
 - rectangles accrochés sous la classe (facultatifs)
 - + pour public / - pour private
- Objet
 - premièreLettreEnMinusculePuisPremiereEnMajusculePourChaqueMot
 - rectangle, référence soulignée



Exo 1-8 : Représenter les classes Cercle et HelloWorld

Hello World version 2

- La première version ne possède pas de constructeur
- Définir un constructeur dans la classe HelloWorld qui se charge d'écrire la phrase ("Hello, World !");
- Modifier la méthode écrite en version 1 pour qu'elle instancie un objet de la classe HelloWorld
- Compiler (et déboguer of course!), lancer

Mon second programme

- Ecrire le code d'une classe `DivisionO` comportant des méthodes permettant de diviser des floats entre eux
 - (2 façons de faire...lequelles)
 - Compiler (et deboguer of course!)
- Ecrire le code d'une classe `Client`, utilisant une classe `DivisionO`
 - Compiler (et deboguer of course!)
- Lancer le programme
- NB : faites les 2 façons!!

Classe Circle

- V1
 - Saisir le code de la classe Circle, comme donné dans la diapo 5 (c'est-à-dire sans constructeur), compiler (et déboguer of course!)
 - Ecrire une classe TestCircle qui contient une méthode `public static void main(String[] arg)`. Cette méthode crée un objet Circle et affiche les coordonnées, la surface et le périmètre de l'objet créé.
 - Compiler(et déboguer of course!), lancer
 - Pourquoi TestCircle réussit-il à créer un objet Circle alors qu'il n'y a pas de constructeur?
 - Quels sont les valeurs des attributs x,y,r de l'objet créé?

Classe Circle suite

- V2
 - Modifier la classe Circle en lui ajoutant les 2 constructeurs `public Circle(double lx, double ly, double lr)` et `public Circle(Circle c)`.
 - Compiler cette classe (et déboguer of course!), lancer, conclusion?
- V3
 - Ajouter le troisième constructeur de la classe Circle, compiler, lancer
 - Modifier la classe TestCircle pour qu'elle crée 3 cercles différents (un par constructeur, donner des valeurs à x,y et r différentes de 0) et affiche leurs caractéristiques. Compiler, lancer