TD 1 IJA

Introduction

Objet, méthode, attribut

Classe, instance

Règles de base du nommage

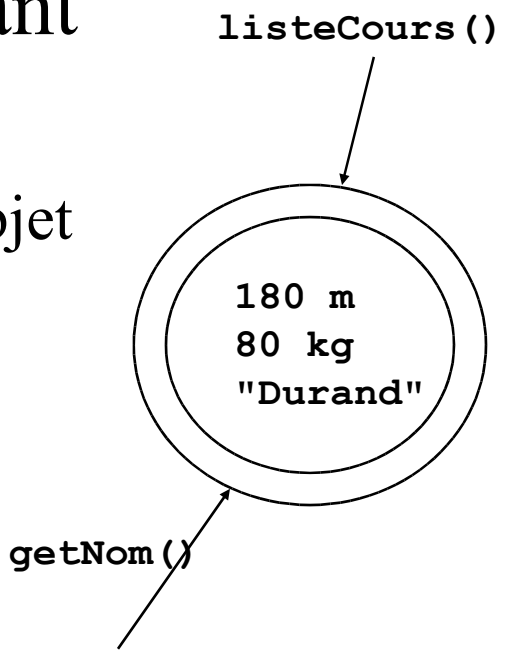
Mes premiers programmes

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

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 : quels attributs et méthodes pour les objets :
  - eleveDurand ? uneConnexion ?

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

Classe - exemple et exo

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

- Exo : écrire la déclaration des classes Eleve et Connexion

Création d'un objet = instanciation

- 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
```

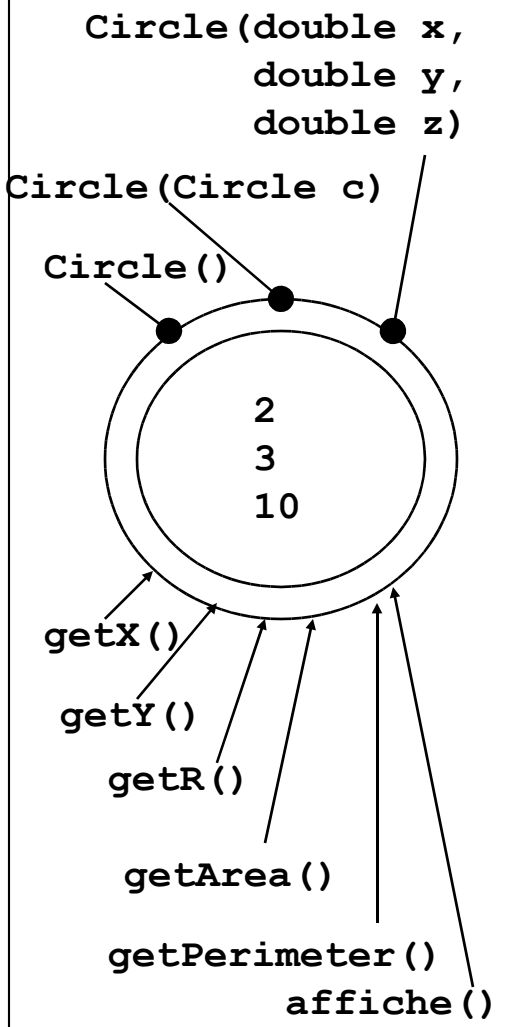
  - `Circle()` est une méthode particulière appelée **constructeur** ; elle a exactement le même nom que la classe

Constructeur

- Tout objet doit avoir ses valeurs positionnées.
  - Un constructeur permet de fixer ces valeurs à la création de l'objet
- 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.

Exemple constructeurs

```
class Circle {
   private double x, y, r;
   public Circle(double lx, double ly, double lr) {
       this.x = lx; this.y = ly; this.r = lr;
   }
   public Circle(Circle c) { //constructeur par copie
       this.x = c.getX();
       this.y = c.getY();
       this.r = c.getR();
   }
   public Circle() {
       this(0.0, 0.0, 0.0);
   }
   public double getX(){return this.x;}
   …//etc les autres méthodes de la classe
}
```



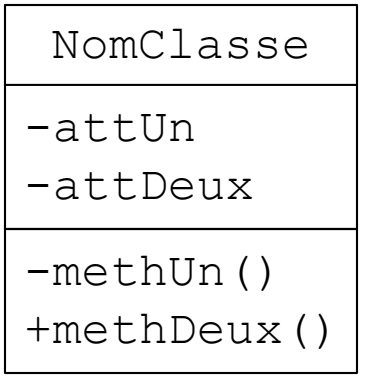
DANGER : Attribut != Variable

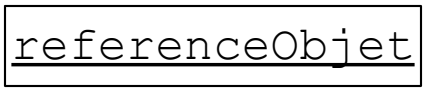
- Les attributs sont relatifs aux objets
  - leur liste est définie dans la classe
  - chaque objet a une valeur pour chaque attribut
  - l'objet s'en souvient tout le temps
  - on peut les utiliser avec « this. » devant leur nom
- Les variables sont relatives à une méthode
  - Elles sont définies dans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

```
//suite classe Circle
public void grossis(int zoom){
  int tmp;
  for(tmp=2 ; zoom>=2 && tmp <= zoom ; tmp++){
    this.r*=tmp;
  }
}
```

Standard de nommage et de représentation graphique (UML)

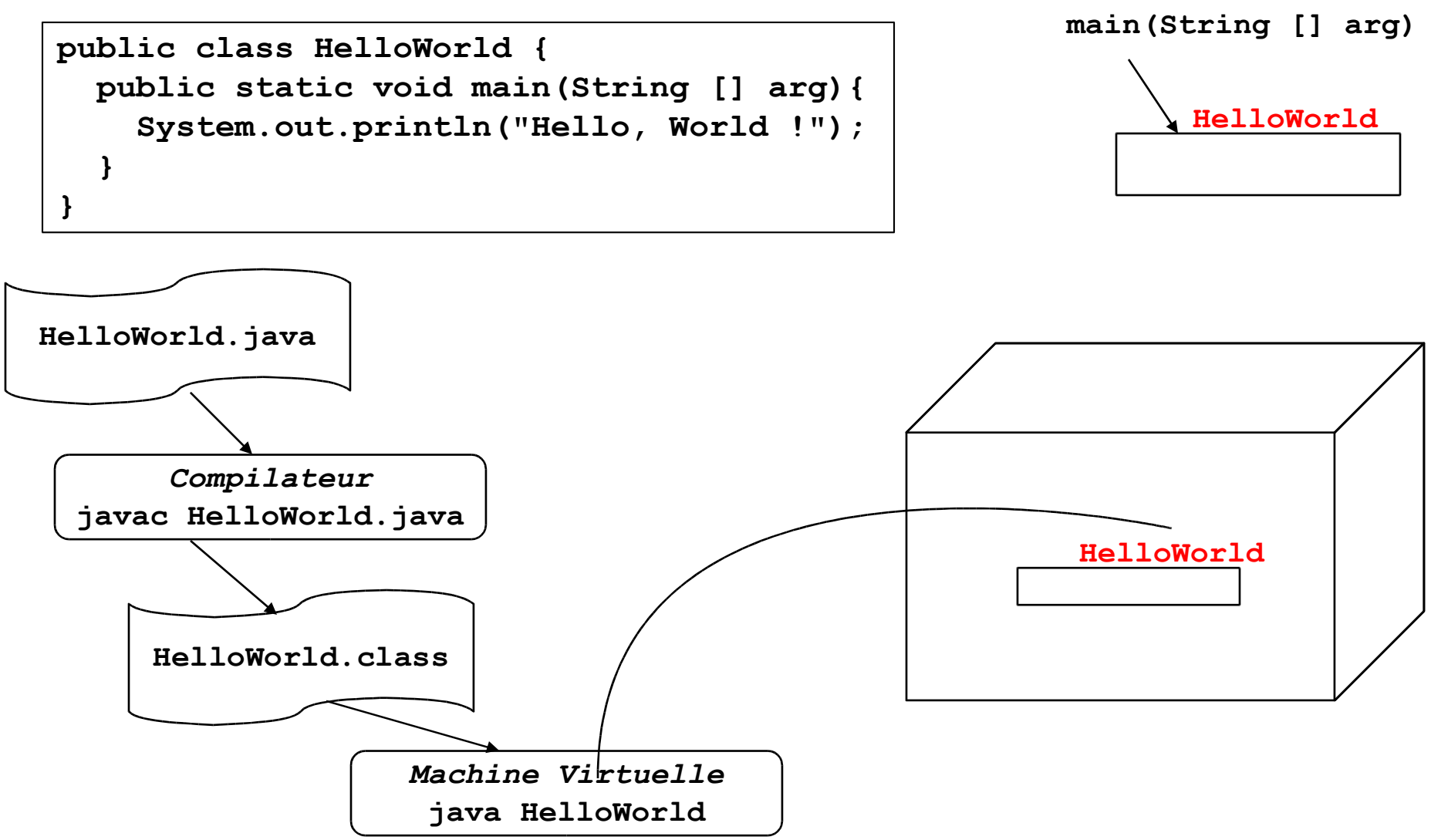
- Classe
  - PremièreLettreEnMajusculePourChaqueMot
  - rectangle
- Attribut et méthode
  - premièreLettreEnMinusculePuisPremiereEnMajusculePourChaqueMot
  - rectangles accrochés sous la classe (facultatifs)
  - + pour public / - pour private
- Objet
  - premièreLettreEnMinusculePuisPremiereEnMajusculePourChaqueMot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





Enfin mon premier programme

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



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

=> faire HelloWorld, compiler (deboguer of course!), lancer

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 deboguer 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 deboguer 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 deboguer 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 deboguer 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