Classes et Objets

Daniel Porumbel

nombreaux slides dus à
Pierre Cubaud

Plan

- la notion théorique d'objet
- un exemple d'objet `Compte`
- un exemple d'objet `Ratio`
- méthodes publiques et privées, mots clé `static` et `this`

La notion théorique d'objet

Un objet

- possède un état constitué de valeurs (attributs)
- possède des actions (méthodes) qui peuvent agir sur ce cet état pour le modifier
  - les méthodes définissent le comportement d'un objet

Un objet est une instance (un exemplaire) d'une classe

La notion de classe

- Une classe est une sorte de modèle pour décrire les objets

Toute classe définit les éléments suivants :

- les caractéristiques (attributs) de ses objets
  - chaque objet d'une classe possède une copie de l'ensemble de ces attributs
- l'ensemble des actions (méthodes) que l'on peut effectuer sur les objets
- Des constructeurs : des méthodes qui permettent d'initialiser les attributs avec des valeurs par défaut

Mécanisme de création d'un objet

Tout objet utilisé par un programme doit porter un nom

Déclaration de variable (nom) :

```
Rectangle unRectangle;
Compte leMien;
Ratio r;
```

Association de chaque variable à un objet

```
unRectangle = new Rectangle(3,2);
leMien      = new Compte(0);
r           = new Ratio(1,3);
```

- Cette étape appelle les constructeurs pour initialiser les objets

Quel est le résultat du code ?

```
class Compte{
    int solde;
    Compte(){           //constructeur
        solde = 0;      //sans arguments
    }
    public void ajouter(int montant){
        solde = solde + montant;
    }
}
void setup(){
    Compte c = new Compte();
    c.ajouter(10);
    println(c.solde);
}
```

Exemples

- Comment ajouter un nom de titulaire à la classe `Compte`
  - Ajouter une deuxième constructeur qui reçoit comme argument le nom du titulaire

Exemples

- Comment ajouter un nom de titulaire à la classe `Compte`
  - Ajouter une deuxième constructeur qui reçoit comme argument le nom du titulaire
- Comment accorder par défaut un crédit de 10 euros

Exemples

- Comment ajouter un nom de titulaire à la classe `Compte`
  - Ajouter une deuxième constructeur qui reçoit comme argument le nom du titulaire
- Comment accorder par défaut un crédit de 10 euros
- Comment retirer de l'argent

Exemples

- Comment ajouter un nom de titulaire à la classe `Compte`
  - Ajouter une deuxième constructeur qui reçoit comme argument le nom du titulaire
- Comment accorder par défaut un crédit de 10 euros
- Comment retirer de l'argent
- Comment verser tout l'argent d'un compte `c1` dans un compte `c2`

La méthode `toString()`

- renvoie une chaîne de caractères qui représente l'objet

```
String toString(){
    return "" +solde;
}
```

- lorsqu'on appelle `println(c)` sur un objet `c`, le programme affiche le résultat renvoyé par `toString()`

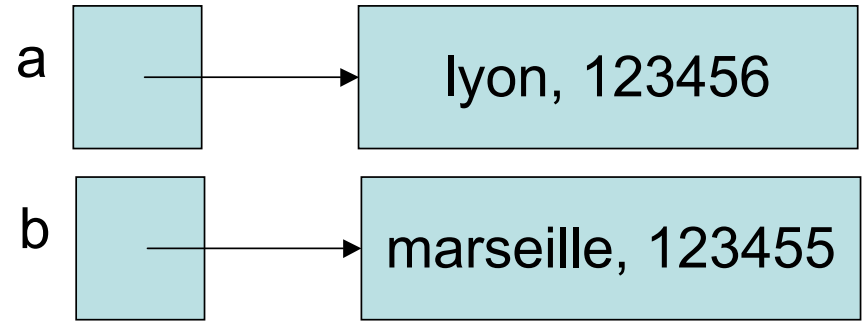
Objets et affectation

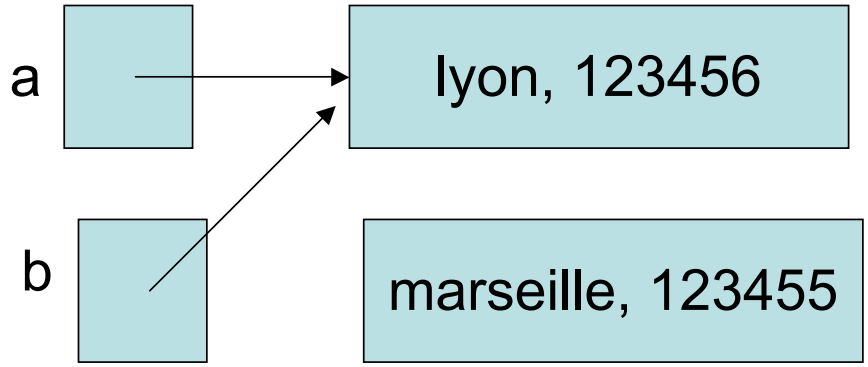
affectation

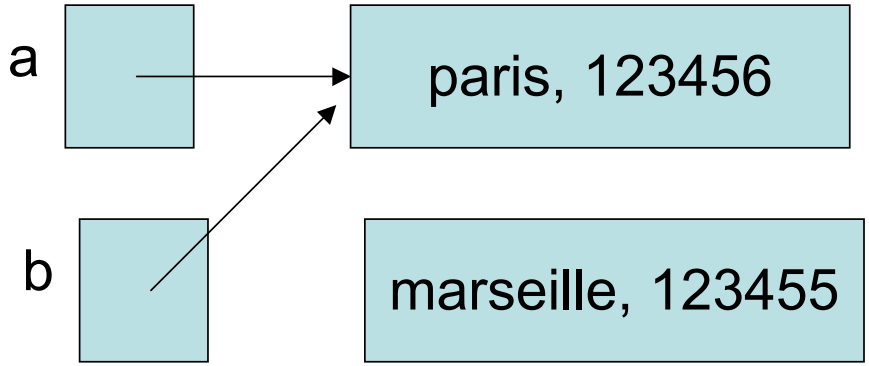
```
class Commune {
 String nom;int popu;
 Commune(String n,int p){
  nom=n;popu=p;
 }
}
Commune a=new Commune("Lyon",123456);
Commune b=new Commune("Marseille",123455);
println(a.nom);
println(b.nom);
b = a;
println(b.nom);
a.nom = "Paris";
println(a.nom);
println(b.nom);
```

Que vont afficher les println() ??

explication :







a et b sont des
<u>références</u> à
du contenu ≠ contenu

Rappel : gestion de comptes bancaires

Données (attributs) d'un compte

- nom de titulaire, solde courant
- historique des dernières opérations (String)

Opérations (méthodes) sur un compte

- obtenir le solde courant, réaliser un retrait, un dépôt
- garder une trace de tout retrait ou dépôt

Rappels Objets

Objet

Collection de valeurs internes (attributs) et d'opérations (méthodes) agissant sur ces valeurs

Un objet *compte bancaire* comporte

état interne variables titulaire, solde, historique, etc.

opérations méthodes applicables sur l'état interne de l'objet (retrait, dépôt)

Rappel : Classe = moule à objets

un objet est construit à partir d'une classe (même moule)

```
Compte c = new Compte (1000, "Alice");
```

- On dit : *c* est un objet de classe `Compte`, ou une instance de la classe `Compte`

Le mot clé `Compte` désigne

- le nom de la classe pour construire l'objet `c`
- le type pour déclarer la variable `c`

Un autre exemple

```
PrintWriter fic;
fic=createWriter(nomFichier);
fic.println(....);
```

- Le mot clé `PrintWriter` désigne un nom de classe
- `fic` est une variable objet
- `println` est une méthode de la classe `PrintWriter`

Que affiche le programme ?

```
class Ratio{
    int a,b;              //ratio = a/b
    Ratio(int num, int den){
        a = num;
        b = den;
    }
    public void ajouter(int c){
        a = a+ b *c ;
    }
    public String toString(){
        return a+"/"+b;
    }
}
void setup(){
    Ratio frac = new Ratio (1,3);
    println(frac);
}
```

5) Mécanismes importants

- this
- public / private
- static

La référence d'objet `this`

```
public class Jour{
  int j; int m; int a;
  public Jour(){}
  public Jour( int j,int m,int a ){
       this.j = j;this.m = m;this.a = a; }

  public Jour( int m,int a ){
       this.m = m;this.a = a; }
  ...
  public void println(){
       System.out.println( this );
       // réalise this.toString()
  }
}
```

Autre utilisation de `this`

```
public class Jour{
  int j;int m;int a;
  public Jour( int j ){
       this( j,getMois(),getAn());
       // getMois et getAn() retournent une valeur
       // pour m et a
  }

  public Jour( int m, int a ){
       this.m = m;this.a = a;
   }

  public Jour( int j,int m, int a ){
       this.j = j;this.m = m;this.a = a;
   }
  ...
}
```

Visibilité

- Les composants logiciels déclarés **`public`** sont utilisables par n'importe quelle classe.
- Les membres **`private`** ne sont accessibles que dans la classe qui les définit.
- Si aucun modificateur **`public`** ou **`private`** n'est spécifié, le membre d'un package (classe, méthode, variable) est accessible à toutes les méthodes du même package.

Variables `static`

Une variable déclarée `static` n'appartient pas aux objets.

- Elle appartient à la classe où elle est déclarée
- Elle est partagée par tous les objets

```
class Ratio{
    static int maxDen = 1000000;
...
}
println(Ratio.maxDen);
```

Méthodes `static`

- Une méthode `static` n'est pas associée à un objet
- Pour l'appeler, on utilise le nom de la classe

```
int    i = Integer.parseInt("123");
double x = Math.round(6.6);
double d = Math.pow(x,2);
```