Introduction aux objets

Cours 2 / GL / LP DA2I

Cédric Lhoussaine

2019-2020

Notion d'objet

Données et traitements

- programmes impératifs "orientés données": on opère sur les variables par étapes successives
- programmes fonctionnels "orientés traitements": on applique aux données d'entrée une composition de fonctions

La programmation "orientée objet" traite

- simultanément données et traitements,
- en les regroupant par entités autonomes

Exemple: les dates

- On veut représenter plusieurs dates dans un programme.
  - tableaux d'entiers `int[] jour; int[] mois; int[] annees`
  - tableau d'entiers à plusieurs dimensions `int[][] dates ;`
  - tableau de chaînes de caractères `String[] dates;`
  - etc.

  Que choisir ?

Exemple: les dates

- On veut représenter plusieurs dates dans un programme.
  - tableaux d'entiers `int[] jour; int[] mois; int[] annees`
  - tableau d'entiers à plusieurs dimensions `int[][] dates ;`
  - tableau de chaînes de caractères `String[] dates;`
  - etc.

  Que choisir ?

- Les dates sont liées à des traitements spécifiques:
  - calcul de la date du lendemain
  - temps écoulé entre deux dates
  - validité d'un jour ou d'un mois
  - affichage avec nom du mois...

Données et traitements

Les objets permettent de regrouper:

- des données (types primitifs ou autres objets) pour former une *structure considérée comme un tout* $\rightarrow$ attributs
- les traitements associés à ces structures $\rightarrow$ méthodes

Données et traitements

Les objets permettent de regrouper:

- des données (types primitifs ou autres objets) pour former une *structure considérée comme un tout* → attributs
- les traitements associés à ces structures → méthodes

object	=	un état	(attributs)
	+	un comportement	(méthodes)
	+	une identité	(référence)

Classes et instances

Classes et instances

- la classe définit :
  - la structure (= attributs)
  - le comportement (= méthodes)

  d'une *catégorie* d'objets $\rightarrow$ c'est une sorte de "moule"

Classes et instances

- la classe définit :
  - la structure (= attributs)
  - le comportement (= méthodes)

  d'une *catégorie* d'objets $\rightarrow$ c'est une sorte de "moule"

- l'instance est un *objet concret* obtenu en donnant des valeurs particulières aux attributs d'une classe.

Objets en Java

- certaines classes sont définies par le langage : `Object`, `String`, `Math`...

Objets en Java

- certaines classes sont définies par le langage : `Object`, `String`, `Math`...
- la plupart sont écrites par le développeur (vous !)

Objets en Java

- certaines classes sont définies par le langage : `Object`, `String`, `Math`...
- la plupart sont écrites par le développeur (vous !)
- l'instanciation s'effectue par la commande `new`

Objets en Java

- certaines classes sont définies par le langage : `Object`, `String`, `Math`...
- la plupart sont écrites par le développeur (vous !)
- l'instanciation s'effectue par la commande `new`
- toute classe définit un type:

```
String x;
```

déclaration d'une variable de type `String` $\rightarrow$ `x` peut *référencer* une instance de `String`

Objets en Java

- certaines classes sont définies par le langage : `Object`, `String`, `Math`...
- la plupart sont écrites par le développeur (vous !)
- l'instanciation s'effectue par la commande `new`
- toute classe définit un type:

```
String x;
```

déclaration d'une variable de type `String` → `x` peut *référencer* une instance de `String`

- exemple

```
// déclaration d'une variable de type String
String s ;
// création d'une instance de la classe String
// référencée par la variable s
s = new String("Hello !") ;
```

→ la chaîne `Hello!` est une instance de la classe `String`

Contenu d'une classe en Java

Le constructeur

c'est une méthode spéciale qui définit ce qui doit être fait pour créer une instance à partir d'une classe donnée

- exemple

```
class Personne {
    String nom, prenom;
    int age;
    // attributs
    Personne(String n, String p) { // constructeur
        nom = n ; prenom = p ; age = 1 ;
    }
}
```

Le constructeur

utilisation:

```
Personne p ;
// instanciation
p = new Personne("César", "Jules") ;
```

- le constructeur diffère des autres méthodes:
  - il porte le nom de la classe
  - il ne comporte pas de type de retour
  - il est appelé par le mot clef new

Variables d'instance ou de classe

Les valeurs des attributs peuvent être:

- propres à chaque instance → *variables d'instance*
  - le poids d'un être humain, son âge...
  - le jour, le mois, l'année constituant une date
- ou partagées par toutes les instances d'une même classe → *variables de classe*
  - le nombre de jambes d'un être humain
  - le nom des mois

Les variables de classe sont "partagées" par les instances

Méthodes d'instance ou de classe

Les traitements associés à une classe peuvent être:

- propres à chaque instance → *méthodes d'instance*
  - les mouvements d'un individu
  - le calcul du lendemain d'une date spécifiée
- partagés par toutes les instances d'une même classe → *méthodes de classe*
  - test d'année bissextile, etc.

Déclaration de variables/méthodes de classe

Les variables (ou méthodes) de classe sont précédées du mot-clef static.

- exemple

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

Une variable ou méthode d'instance peut utiliser une variable ou méthode de classe, mais pas l'inverse!

La classe Date

```
public class Date {
    // ATTRIBUTS
    // variables d'instance
    int jour, mois, annee ;
    // variables de classe
    static String[] MOIS = { "Janvier", "Février", ... } ;
    // Constructeur
    public Date(int j, int m, int a) {
        jour = j ;
        mois = m ;
        annee = a ;
    }
    // une méthode d'instance
    public String toString(){
        return jour + " " + MOIS[mois] + " " + annee ;
    }
    // une méthode de classe
    public static boolean estBissextile(int a){
        return
            (a % 400 == 0)
            ||((a % 4 == 0) && (a % 100 != 0)) ;
    }
} // fin de la classe
```

Utilisation des objets

Manipulation d'objets

- on "utilise" un objet en lui envoyant des messages
- un message = une méthode et ses arguments
- syntaxe en Java : `unObjet.unMessage`
- exemple: méthode donnant la longueur d'une chaîne

```
String s = "Hello !" ;
int l = s.length() ;
```

Manipulation d'objets

```
public class MontrerDate {
    // classe servant à lancer l'application
    public static void main(String [] args) {
        // méthode exécutable directement
        // déclaration d'une variable de type Date
        Date d ;
        // instanciation d'une Date
        d = new Date(8, 4, 2006) ;
        // affichage sur la sortie standard
        System.out.println(d.toString()) ;
        System.out.println(Date.estBissextile(2010)) ;
    }
}
```

Arborescence des fichiers

État initial

Les fichiers sources sont rassemblés dans un répertoire

```
essai/
essai/Date.java
essai/MontrerDate.java
```

Compilation

On peut alors compiler le fichier principal:

```
essai$ javac MontrerDAte.java
```

le compilateur cherche un fichier source `Date.java` définissant la classe `Date`, et le compile.

Résultat de compilation

- *Fichiers obtenus*

```
essai/
essai/Date.java
essai/Date.class
essai/MontrerDate.class
essai/MontrerDate.java
```

- *Exécution* de la classe contenant la méthode `main`:

```
essai$ java MontrerDate
```

la machine virtuelle cherche un fichier `MontrerDate.class` et exécute sa méthode `main`.

- *Résultat*

```
essai$ java MontrerDate
8 avril 2006
false
```

Résultat de compilation

À noter

- la méthode `toString` de Date s'est exécutée dans le contexte de l'instance `d`, i.e. pour les valeurs des attributs de l'instance `d`.
- la méthode `estBissextile` de `Date` s'est exécutée dans le contexte de la classe `Date`.

Organisation du code

Séparer les sources du bytecode

```
essai/
essai/sources/
essai/sources/Date.java
essai/sourcesMontrerDate.java
essai/classes/
essai/classes/Date.class
essai/classes/MontrerDate.class
essai/classes/doc/
```

Deux concepts clefs

- `sourcepath`: chemin contenant les sources
- `classpath`: chemin contenant les classes (le bytecode)

Compiler et exécuter avec des options

Exemple

```
essai$ javac -sourcepath sources -d classes sources/MontrerDate.java
essai$ java -classpath classes MontrerDate
8 Avril 2006
false
```

Fonctionnement

- le compilateur détermine les dépendances
- il cherche des sources dans les répertoires du sourcepath
- la machine virtuelle recherche toutes les classes dans les répertoires du classpath

Emacs 26.2 (Org mode 9.1.9)