CONCEPTION MODÉLISATION OBJETS

Walter Rudametkin
Maître de Conférences
Bureau F011
Walter.Rudametkin@polytech-lille.fr

Copyright notice

- Ce cours est très largement inspiré (i.e. copié) des cours de Bernard Carré et d'Anne Etien, et indirectement de Jean-Christophe Routier à Lille 1, entre autres.

Warning

- Je suis étranger…
  - et j'ai un **accent**
- Je me **trompe beaucoup** en français

  (et en info, et en math, et …)
  - N'hésitez pas à me corriger ou à me demander de répéter
- Je **commence** à enseigner
  - J'accepte des critiques (constructifs)
    - et surtout des **recommandations**
  - N'hésitez pas à poser des questions

Présentation

- Objectif : **Concevoir une solution objet de qualité et la réaliser en Java**
  - Présenter les concepts de base de l'approche objet
  - Adopter le "penser objet"
  - Connaître et savoir mettre en œuvre les concepts fondamentaux
  - Sensibiliser à la production d'un code de qualité en Java
- Organisation :
  - 12h de cours
  - 10h de TP
- Notation :
  - DS (2h)
  - TP/Projet

A l'issue de ce module vous devriez...

- **Connaître les éléments de base de la programmation objet**
  - maitriser le vocabulaire de la programmation objet :
    - classe, instance, méthode, interface, attribut, constructeur, encapsulation, polymorphisme
  - savoir décomposer un problème simple en classes et objets
  - savoir expliquer ce qui différencie la programmation objet des autres paradigmes
  - pouvoir identifier certaines situations de mauvaises conception objet et les corriger

A l'issue de ce module vous devriez...

- **Savoir spécifier et coder un problème objet dans le langage Java**
  - Connaître les principaux éléments de la syntaxe du langage java
  - Être en mesure d'écrire (et corriger) un programme dans le langage Java
  - Pouvoir expliquer clairement le rôle et la sémantique des éléments de langage suivants et savoir les utiliser :
    - new, public, private, enum this, static, final, package import, throws, throw
  - Comprendre le transtypage (upcast/downcast)
  - Être en mesure de choisir une structure de données appropriée et savoir utiliser les types java
    - List, Set, Map, Iterator …
  - Savoir gérer les exceptions et connaître la différence entre capture et levée d'exception
  - Savoir utiliser les "outils" liés à la plateforme java :
    - javac, java (et classpath), javadoc, jar

Bienfaits de l'abstraction

- Tous les langages de programmation fournissent des abstractions.
- Qu'est-ce qu'on tente d'abstraire ?
  - Le langage assembleur : petite abstraction de la machine sous-jacente.
  - Les langages « impératifs » : abstractions du langage assembleur.
    - nettes améliorations par rapport à l'assembleur,
    - Mais réflexion en termes de structure ordinateur
  - Alternative : modéliser le problème qu'on tente de résoudre.
    - LISP : une vue particulière du monde (« Tous les problèmes se ramènent à des listes »)
    - PROLOG convertit tous les problèmes en chaînes de décisions.
  - Adapté mais … pas adapté

Bienfaits de l'abstraction

- L'approche orientée objet va un cran plus loin
  - Description du problème avec ses termes plutôt qu'avec ceux de la machine
  - Représentation assez générale→ pas de restriction à un type particulier de problèmes.
  - Utilisation des « objets » dans l'espace problème et dans l'espace solution.
    - Plus autres objets qui n'ont pas leur analogue dans l'espace problème.
- Abstraction plus flexible et puissante que précédemment.
- Chaque objet ressemble à un mini-ordinateur ;
  - un état,
  - des opérations qu'on peut lui demander d'exécuter.
- Attention : la PPO n'est pas adéquate pour résoudre facilement tous les problèmes de programmation

Caractéristiques des langages purs objet

- Toute chose est un **objet**.
- Un programme est un ensemble d'objets se disant les uns aux autres quoi faire en s'envoyant des **messages**.
- Chaque objet est d'un **type** précis.
  - chaque objet est une *instance* d'une *classe*,
- Tous les objets d'un type particulier peuvent recevoir le même message.

Ces objets qui nous entourent

- des voitures, des livres, des portes, des chaises, des ordinateurs, des thermomètres, des téléviseurs,...
- des chats, des personnes, des facteurs, des éléphants,...
- des comptes en banque, des dossiers étudiant, des connexions réseau,...
- Ces objets
  - ont des **caractéristiques** : la couleur d'une voiture, l'âge ou le nom d'une personne, le solde d'un compte en banque,...
  - ont un **comportement** : ouvrir la porte, le chat miaule, créditer le compte en banque,...

Exemples

- Un thermomètre mesure une température (un nombre)
- Cette température mesurée est une donnée, ou caractéristique, de ce thermomètre . Elle définit son **état**.
- Sur ce thermomètre, on peut envisager certaines manipulations ou
- Opérations = le **comportement** du thermomètre :
  - obtenir la température en degrés Celsius ou en Fahrenheit
  - modifier la température mesurée
  - associer à la température une couleur (bleu, vert, rouge, ...) ou un mot (froid, normal, chaud,...)
- A chaque opération correspond un **traitement**.

Un premier thermomètre

- Un objet thermomètre → soit thermo1 son identité.
- Son état est défini par une température mesurée temp (22.5◦C).
- On peut exploiter son comportement :

  temperatureEnCelsius, temperatureEnFahrenheit, modifierTemperature, couleurTemperature

  thermo1.temperatureEnCelsius() → 22.5

  thermo1.temperatureFarenheit() → 72.5

  thermo1.modifierTemperature(25.8) → -

  thermo1.temperatureEnCelsius() → 25.8

  thermo1.couleurTemperature() → Couleur.VERT

Le comportement dépend de l'état et agit sur l'état.

Java

- langage orienté objet (pas 100% objet), langage de classes
- langage compilé, fortement typé
- indépendance OS/architecture : multi plate-forme
  - utilisation d'une machine virtuelle (la JVM) - bytecode Java
  - "compile once, run everywhere"
- gestion dynamique de la mémoire
  - utilisation d'un GC (garbage collector = ramasse-miettes)
- gestion des erreurs par exceptions
- nombreuses bibliothèques/API (gratuites) (réseau, RMI, JDBC, etc.)
- existe depuis 1995, libre depuis ~ 2007... – JDK, JRE, SDK

En Java

```
public class Thermometre {
  private float temp;
  public Thermometre(float tempInit) {
     this.temp = tempInit;
  }
  public float temperatureEnCelsius() {
     return this.temp;
  }
  public float temperatureEnFahrenheit() {
     return (9.0/5.0)*this.temp+32;
  }
  public void modifierTemperature(float
              nouvelleTemp) {
     this.temp = nouvelleTemp;
  }
```

```
…
public Color couleurTemperature() {
    if (this.temp < 0) {
       return Color.BLUE;
    }
    else if (this.temp < 30) {
       return Color.GREEN;
    }
    else return Color.RED;
  }
}
```

Des questions ?

Matérielle additionnelle à continuation

Objet

Objet = identité + état (attributs) + comportement (méthodes)

avec

L'**identité** permet d'exploiter le **comportement** d'un objet. Le **comportement** agit sur l'**état** et l'**état** influence le **comportement**

Une identité

- Une identité permet de s'adresser à l'objet
- chaque identité est unique
  - deux objets différents ont des identités différentes
- on peut faire référence à l'objet (à son identité), la nommer
- on peut avoir plusieurs références pour une seule identité (un seul
- objet)

Un état

- ensemble de propriétés ou caractéristiques définies par des valeurs
- valeurs propres (personnelles) à chaque objet
- l'état d'un objet (les valeurs des propriétés) peut évoluer dans le temps

Un comportement

- ensemble des traitements que peut accomplir un objet (ou que l'on peut lui faire accomplir)
- On dit que l'on **invoque une méthode sur un objet**.
- On ne peut utiliser une méthode qu'en l'invoquant sur un objet.

Envoi de message

- on s'adresse à l'objet par envoi de messages
  - on "demande" à l'objet de faire quelque chose

    envoi de message = invocation de méthode

- le comportement définit l'ensemble des messages qu'un objet peut recevoir

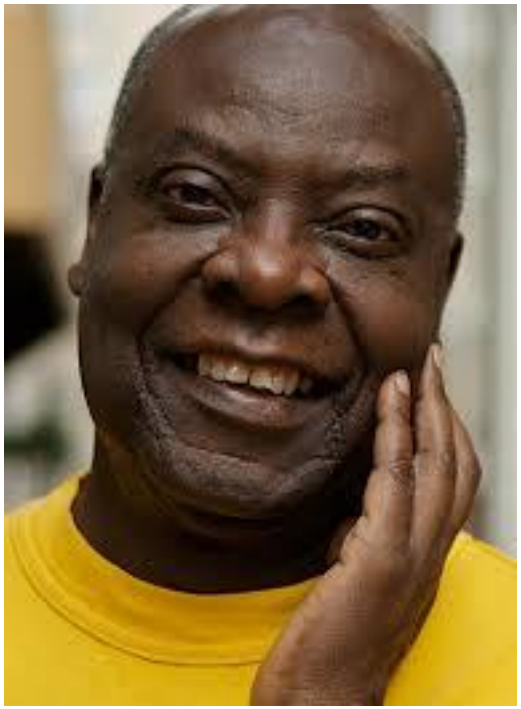
- interface de l'objet
  - "ensemble" des manières que l'on a pour interagir avec l'objet
  - ensemble des messages reconnus par l'objet
  - "interface de comportement"

4 images, 1 concept

- Quelles caractéristiques ? Quels comportements ?

4 images, 1 concept

JRR Tolkien



E Dongala

JK Rowling

EE Schmitt

- Quelles caractéristiques ? Quels comportements ?

4 images, 1 concept

- Quelles caractéristiques ? Quels comportements ?

- certains objets présentent les mêmes caractéristiques :
  - identités différentes mais
    - états définis par les mêmes attributs
    - même interface de comportement
- exemples :
  - les thermomètres thermo1 et thermo2
  - des livres “Le Seigneur des Anneaux” de John Ronald Reuel Tolkien paru en 1954 et “Dune” de Frank Herbert paru en 1965

  sont caractérisés par les mêmes attributs
  - auteur, titre, année, texte

  et ont la même interface de comportement
  - on peut leur faire accomplir les mêmes actions :
  - on peut les lire, les imprimer, etc.

il en serait de même pour (~) tous les livres

Tous les livres obéissent à un même schéma

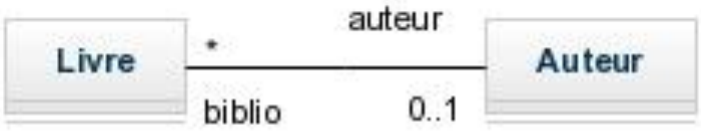
→ on peut en abstraire un “moule”, un patron, un modèle, etc

- Le moule définit
  - les attributs qui caractérisent l’état
  - l’interface et sa réaction = le comportement → les méthodes de tous les moulages qui en seront issus

“moulages = objets”


Livre
titre: String
annee: Integer
texte: String
adaptation: Boolean
auteur: Auteur
lire(): void
afficher(): void
adapter(): void


Auteur
nom: String
prenom: String
naissance: Date
deces: Date
nationalite: String
biblio: Livre
ajouteBiblio(): void
afficher(): void
fixerDeces(): void


Livre
auteur
*
Auteur
biblio
0..1

Classes et instances

Un moule est appelé **classe**. Une classe est un **type**.

Les moulages sont les **objets** appelés **instances** de la classe

- Une fois qu'on a la classe, on peut potentiellement créer autant d'objets / instances conformes à la classe que l'on veut :
  - Ils ont des identités différentes
  - Ils ont des états définis par les mêmes attributs, mais avec des valeurs différentes
  - Ils auront le même comportement (mêmes méthodes)

- programmation définition des classes → abstraction
- à l'exécution travail sur des objets/instances → concrétisation

- La classe définit le comportement de toutes ses instances
- Les instances ont des identités différentes et des valeurs d'attribut différentes.

Interface d'une classe
= ensemble des messages acceptés par les instances de la classe
≈ ensemble des signatures des méthodes publiques (généralement)