Méthodes et penser objet

Arnaud Labourel arnaud.labourel@univ-amu.fr

1er février 2022

Section 1

Appel de méthodes

Utilisation d'objet

On interagit avec un objet uniquement via :

- ses méthodes
- ses attributs

Appel/invocation de méthode

Appel/invocation de méthode = « envoi de message »
L'objet a le contrôle sur le message qui lui est envoyé :
$\Rightarrow$ seulement les méthodes définies dans la classe de l'objet sont autorisées.
Utilisation de la notation pointée :

```
aPerson.isAdult()
```

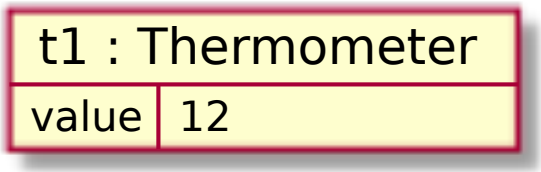
Exemple : classe `Thermometer`

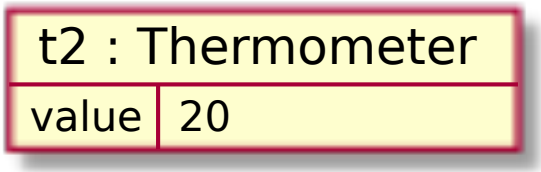
Classe `Thermometer`

- structure attributs : `value` de type `int`
- méthodes : `getValue()`, `toFahrenheit()`, `toString()`

`t1` et `t2` deux instances de la classe Thermometer

$\Rightarrow$ même structure d'attributs, mais valeurs différentes





Exemple de code : classe Thermometer

```
public class Thermometer {
  private float temperature; // attribut
  public Thermometer(float initialTemp) {
    // constructeur
    this.temperature = initialTemp;
  }
  public float temperatureInCelsius() {
    // méthode
    return this.temperature;
  }
  public void changeTemperature(float newTemp) {
    // méthode
    this.temperature = newTemp;
  }
 //...
```

Exemple de code : classe Thermometer

```
  // suite
  public String toString() { // méthode
    return "Current temperature is : "
      + this.temperature + "°C";
  }
  public float temperatureInFahrenheit() { // méthode
    return (9.0f/5.0f)*this.temperature+32;
  }
}
```

Exemple : classe `Person`

Classe Person :

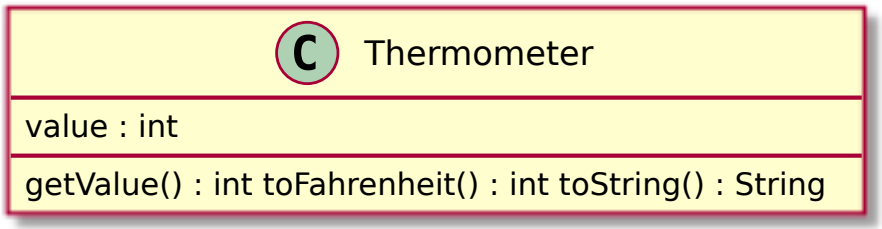
- structure attributs : `birthYear` de type `int`, `name` de type `String`
- méthodes : `isAdult()`, `isOlderThan(Person)`, `getBirthYear()`, `getName()`, `toString()`

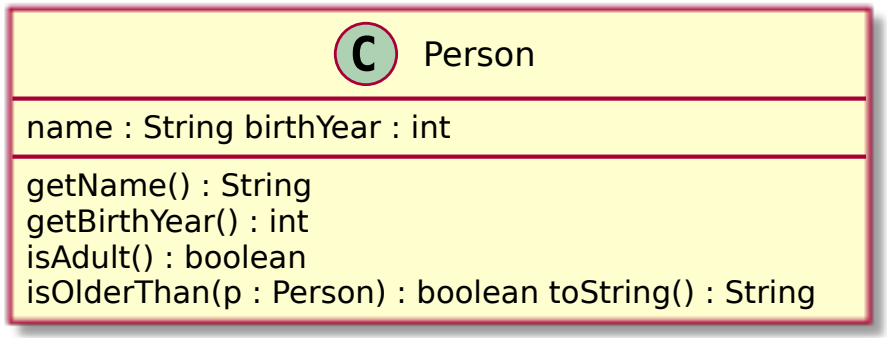
`alice` et `bob` deux instances de la classe `Person`

alice : Person	
name	"Alice"
birthYear	2001

bob : Person	
name	"Bob"
birthYear	2006

Exemples d'utilisations






```
t1.toFahrenheit() // -> 53.6
t2.toFahrenheit() // -> 68
t1.toString() // -> "Current temperature is : 18°C"
t1.isAdult() // impossible (erreur à la compilation)
alice.getBirthYear() // -> 19
alice.isAdult() // -> true
alice.toString() // -> "Alice is born in 2001"
alice.toFahrenheit()
// impossible (erreur à la compilation)
```

Envoi de message

Conformément à sa classe (son type), l'objet « contrôle » le message :

- sa légalité
- pas d'ambiguïté pour deux messages (méthodes) de même nom dans des classes différentes (cf. `toString()`)

Règle d'appel de méthode

- Une méthode ne peut pas être utilisée autrement qu'en étant appelée/invoquée sur un objet via un envoi de message à cet objet
- La validité du message pour le type de la référence est vérifié à la compilation

Receveur

L'objet invoquant = le **receveur** du message

Il fait partie du contexte d'exécution de la méthode

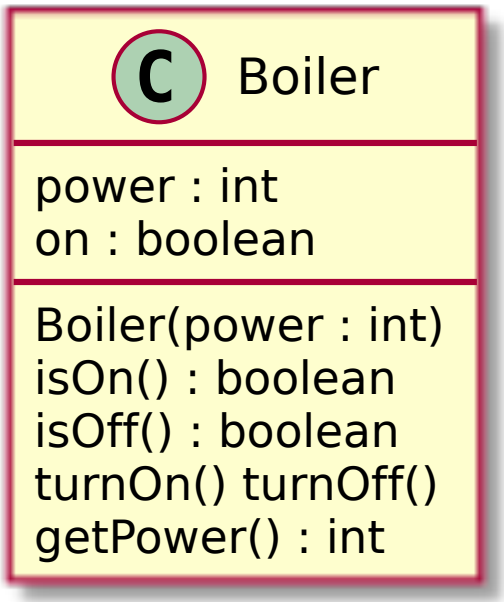
$\Rightarrow$ dans le code d'une méthode, on peut :

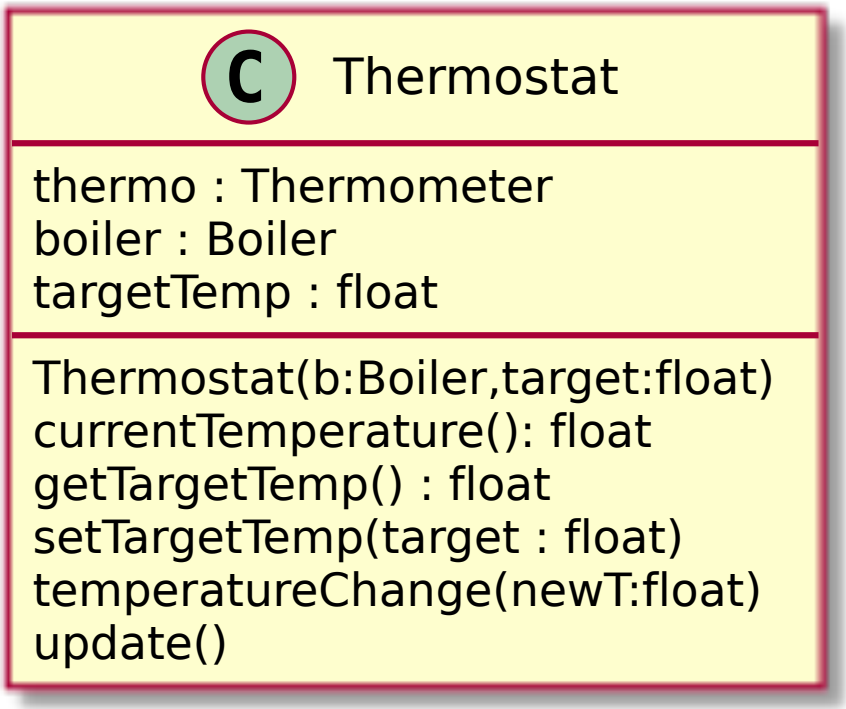
- appeler une méthode sur le receveur
- accéder aux attributs du receveur

Messages en cascade

Les attributs d'un objet étant eux-mêmes des objet : il peut y avoir des messages en cascade.
$\Rightarrow$ Appel d'une méthode d'un receveur peut entraîner l'appel d'une méthode sur un de ses attributs.

Exemple thermomètres, chaudières et thermostats






```
Boiler b = new Boiler(1000);
Thermostat t = new Thermostat(b, 20);
t.temperatureChange(10);
```

$\Rightarrow$ Envoi de messages en cascade `t` $\rightarrow$ `b`

Classe et instance

Classe = définition/modèle

- décrit la structure de l'état des objets : les attributs et leurs types
- définit les envois de messages possibles : les méthodes
  $\Rightarrow$ **interface** d'une classe

Abstrait

Instance = objet conforme au modèle de la classe qui l'a créé

- son état correspond à la structure définie par sa classe
  $\rightarrow$ association de valeurs aux attributs
- n'accepte que les messages définis par la classe
  = n'exécute que les méthodes définies par sa classe

Concret

Section 2

Le penser objet

Analyse d'un problème en programmation objet

Penser objet : décomposer le programme en objets

- Quels sont les objets nécessaires à la résolution du problème ?

  $\Rightarrow$ décomposition du problème en objets

- À quels modèles des objets correspondent-il ?

  $\Rightarrow$ Quelles sont les classes ?

- Quels sont les fonctionnalités/opérations dont on doit/veut pouvoir disposer sur ces objets ?

  $\Rightarrow$ Quelles sont les méthodes des classes ?

- Quelle est la structure des données de l'objet ?

  $\Rightarrow$ Quelles sont les attributs des classes ?

Exemple de problème

- un catalogue regroupe des articles, il permet de trouver un article à partir de sa référence.
- un article est caractérisé par un prix et une référence que l'on peut obtenir. On veux aussi pouvoir déterminer si un article est plus cher qu'un autre
- une commande est créée pour un client et un catalogue donnés, on peut ajouter des articles à une commande, accéder à la liste des articles commandés ainsi que prix total des articles et le montant des frais de port de la commande.
- un client peut créer une commande pour un catalogue et commander dans cette commande des articles à partir de leur références.

Classes du problèmes

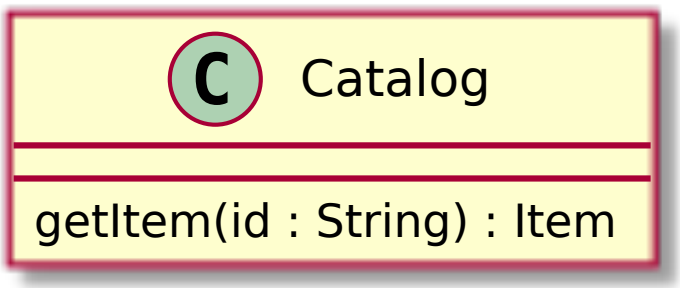
- un **catalogue** regroupe des **articles**, il permet de trouver un article à partir de sa **référence**.
- un **article** est caractérisé par un prix et une référence que l'on peut obtenir. On veux aussi pouvoir déterminer si un article est plus cher qu'un autre
- une **commande** est créée pour un **client** et un **catalogue** donnés, on peut ajouter des **articles** à une commande, accéder à la liste des articles commandés ainsi que prix total des articles et le montant des frais de port de la commande.
- un **client** peut créer une **commande** pour un catalogue et commander dans cette commande des **articles** à partir de leur références.

Méthodes

- un catalogue regroupe des articles, il permet de **trouver** un article à partir de sa référence.
- un article est caractérisé par un prix et une référence que l'on **peut obtenir**. On veux aussi pouvoir **déterminer** si un article est plus cher qu'un autre
- une commande est créée pour un client et un catalogue donnés, on peut **ajouter** des articles à une commande, **accéder** à la liste des articles commandés ainsi que prix total des articles et le montant des frais de port de la commande.
- un client peut **créer** une commande pour un catalogue et **commander** dans cette commande des articles à partir de leur références.

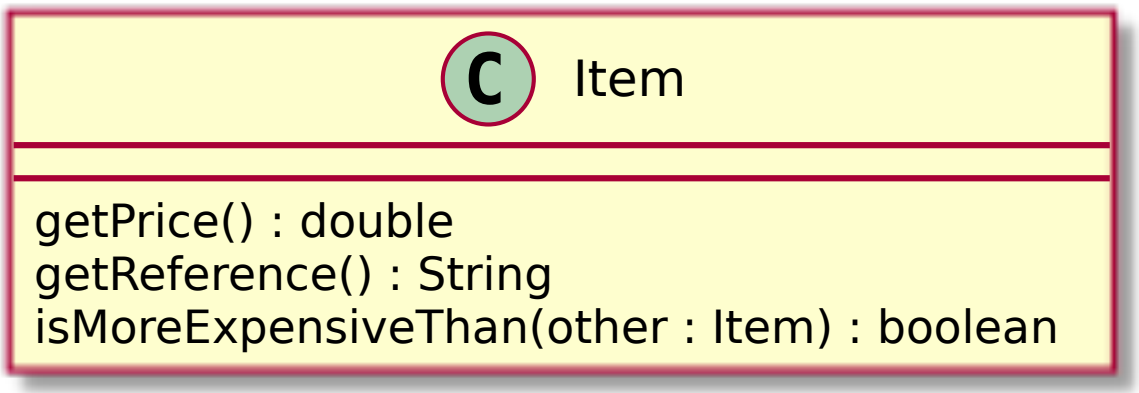
Description d'un catalogue

un catalogue regroupe des articles, il permet de **trouver** un article à partir de sa référence.



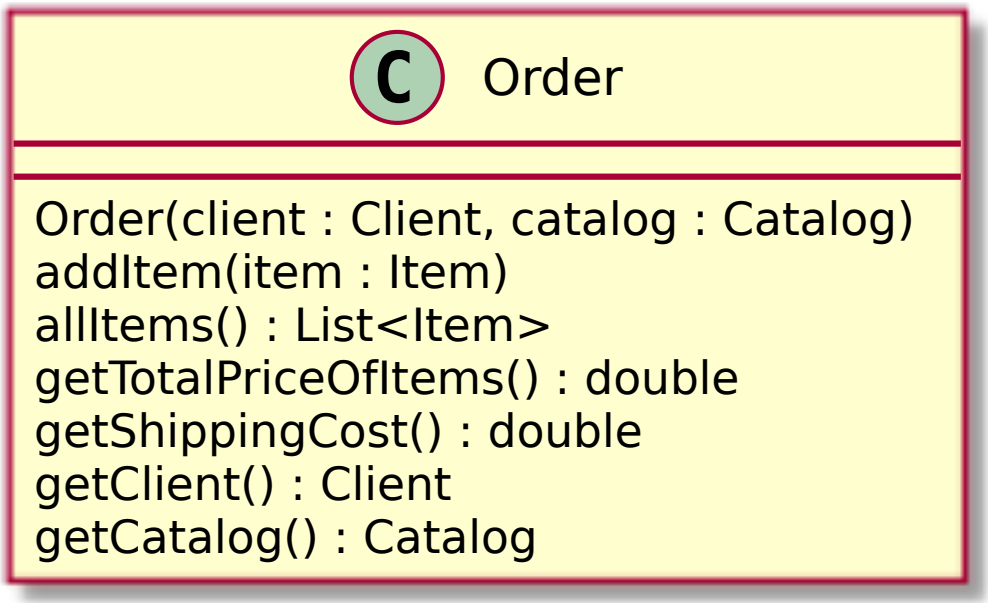
Description d'un article

un article est caractérisé par un prix et une référence que l'on **peut obtenir**. On veux aussi pouvoir **déterminer** si un article est plus cher qu'un autre



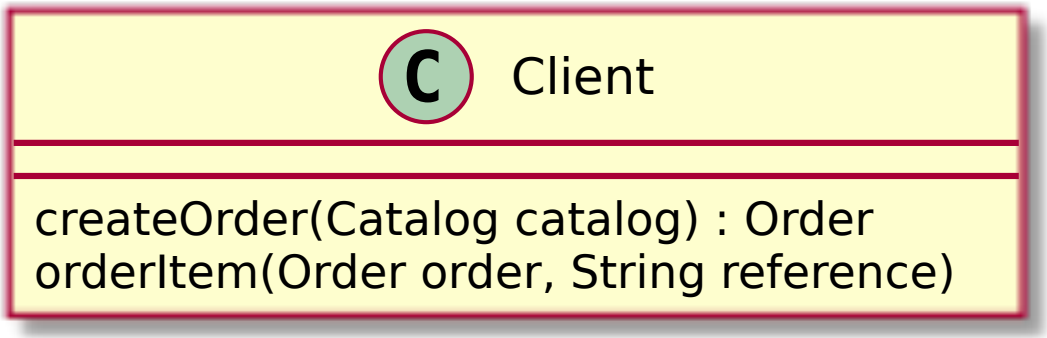
Description d'une commande

une commande est **créée** pour un client et un catalogue donnés, on peut **ajouter** des articles à une commande, **accéder** à la liste des articles commandés ainsi que prix total des articles et le montant des frais de port de la commande.



Description d'un client

un client peut **créer** une commande pour un catalogue et **commander** dans cette commande des articles à partir de leur références.



Utilisation des classe

Créer une commande pour un client, faire commander 2 articles par le client, obtenir le prix total des articles de la commande

on suppose les références disponibles et initialisées :

```
Client client = new Client(...)
Catalogue cata = new Catalogue(...)

Order order = client.createOrder(cata);
client.orderItem(order,"A0527");
client.orderItem(order,"B3879");
float price = order.getTotalPrice();
```

Code des méthodes

La méthode `orderItem` de `Client` permet d'ajouter un article à une commande à partir de son identifiant quel code pour cette méthode ?

```
public void orderItem(Order order, String id) {
  // récupérer le catalogue de la commande
  Catalog cata = order.getCatalog();
  // récupérer l'article dans le catalogue
  // à partir de la référence
  Item item = cata.getItem(id);
  // ajouter l'article à la commande
  order.addItem(item);
}
```