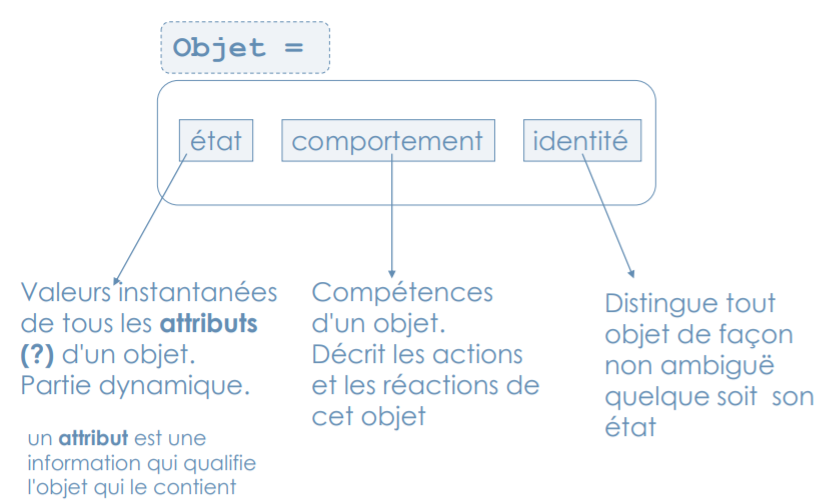
**Modélisation objet/ Programmation Orientée Objet**

1. Concept :

* Objets
* Message (action d’un objet sur un autre)
* Classe (détails/attributs d’un objet)
* Heritage ?
* Polymorphysme ?

1. Objets :



Etat/attribut : couleur, taille, etc.

Comportement/méthode : son action

Identité : ex à qui il appartient, sorte de donnée unique, nom de l’objet

1. Les messages

Représentent le comportement ou les services d’un objet.

Composés de :

* Un nom,
* Une liste de paramètres d’entrées
* Une liste de paramètres de sortie
* La signature d’un objet.

Réception d’un message implique un traitement.

Il existe **5 catégories de messages** :

– les constructeurs, qui créent les objets et peuvent initialiser les attributs (sont toujours présent dans une classe et portent le même nom que celle-ci) : sont appelés constructeurs au lieu de méthode à cause de cette caractéristique d’unicité. (Ex : Cercle() = constructeur vide et Cercle (…) = constructeurs avec arguments)

– les destructeurs, qui détruisent les objets

– les sélecteurs qui renvoient tout ou partie de l'état d'un objet

– les modificateurs qui changent tout ou partie de l'état d'un objet

– les itérateurs qui visitent l'état d'un objet ou le contenu d'une structure de données qui contient plusieurs objets.

Chaque **objet** a des attributs et des méthodes (fonction de traitement/comportement de l’objet (ex : avancer pour un vélo)).

Argument : dans l’exemple, il s’agit de rayon : Integer et couleur : String. Ex : Cercle (rayon : Integer, couleur : String)

**2 types de comportement** pour l’objet qui le reçoit :

– Statique si l’état de l’objet n’intervient pas dans le traitement invoqué ;

– Dynamique si le comportement de l’action invoquée dépend de l’état de l’objet

Les valeurs de retour se trouvent après les méthodes. Ex : Surface () : Double et Circonf() : Double. Ici, les valeurs de retour sont les Double.

1. Les classes

Saladier = classe : endroit où l’on dépose tous les objets pour créer un nouvel objet (endroit où l’on regroupe tous les ingrédients pour faire un gâteau)

On y regroupe les objets qui partagent des propriétés et des comportements communs.

+ : signe des classes publiques (*public*)

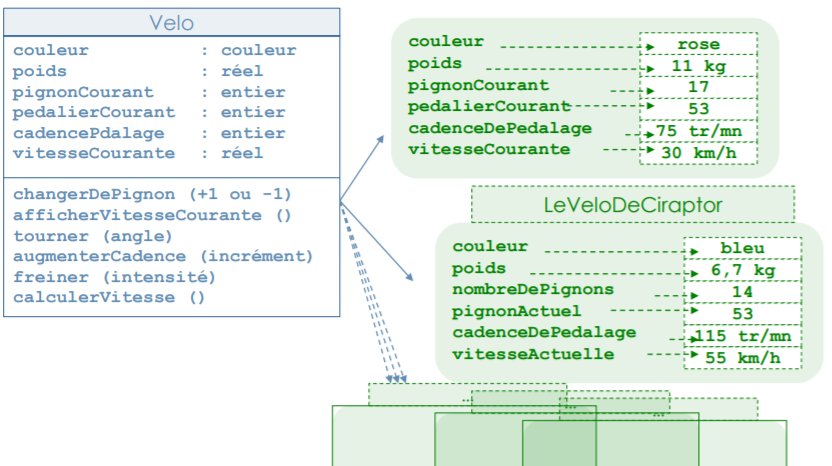
- :signe des classes privées (*private*)

# : signe pour la classe protégée (*protected*). Peut être utilisé uniquement par sa propre classe mais ne peut être modifié par aucune classe (même la sienne)

**CLASSE : FABRIQUE DES OBJETS**

**OBJET : ELEMENT D’UNE CLASSE**

Les objets sont construits à partir de classe par **INSTANCIATION.**



**ORIGINES DE JAVA**

Invention de java car le langage C n’était pas adapté. Il fallait que les logiciels puissent s’installer partout (imprimante comme réfrigérateur) sur un réseau.

Premier nom : OAK

HOTJava : premier navigateur supportant les applets

Applet : programme écrit en JAVA qui peut être embarqué dans une autre application

Avantage principal : développement de logiciels uniquement avec la modélisation orienté objet. Condition sine qua none pour utiliser JAVA

Pas de pointeur en JAVA.

Les classes sont organisées en paquetages (packages) :

* Avec paquetage java.net aussi facile d’ouvrir un fichier local que distant
* Avec paquetage java.rmi, on peut faire communiquer deux classes situées sur deux machines distantes

En opposition au C, C++… qui sont des langages compilé, JAVA n’a pas besoin d’être compilé puis interprété.

JAVA est très robuste :

– Java élimine certains type d’erreur de programmation,

– est fortement typé (pas de conversion automatique risquant une perte de données),

– est fiable (gestion des pointeurs et de la mémoire),

– supporte la gestion des exceptions (récupération erreurs).

INCONVENIENTs :

Performant … pour un langage interprété (jusqu’à 20x plus lent que du C)

Suffisant pour applications interactives, avec IG, et basées sur le réseau

Pas toujours suffisant tout de même (Systèmes critiques) : dans ce cas langage de bas niveau (C, C++, Fortran)

Pour rendre Java plus rapide : – Compilateurs (perte de la portabilité)

Cours > orli > logiciels pedagogique > java

* Installé dans C > Program files

Encapsulation : Cache des attributs dans un objet, pour éviter que les autres le modifie ou y accede

Le mot public signifie que tout le monde peut acceder à l’objet. S’utilise pour les methodes, les attributs, les classes. Choses associés à *public* sont accessibles dans toutes les classes.

Le mot privé s’utilise pour que personne ne puisse accéder à l’attribut ou la méthode. En général, toutes les classes sont publiques. Choses associés à *private* accessibles seulement dans la même classe.

Protégée (*protected*) ne sera pas utilisée pour l’instant.

Tout les accesseurs et le modifieurs sont codés de la même manière

public class Voiture {

private double vitesse;

[...]

public double getVitesse() {

return vitesse;

}

public void setVitesse(double vitesse) {

this.vitesse = vitesse;

}

}

Et on peut alors l’utiliser de cette manière :

public class UneAutreClasse {

[...]

Voiture ma\_voiture;

ma\_voiture = new Voiture();

~~ma\_voiture.vitesse = 50;~~ On utilise la ligne du dessous car vitesse est signalée comme privée !

ma\_voiture.setVitesse(50);

}

Relations entre classes

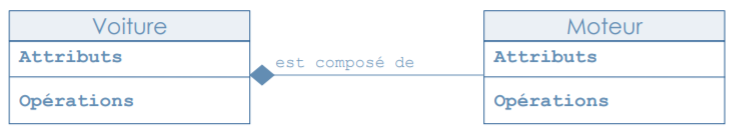
Association : Connexion sémantique bidirectionnelle entre classes



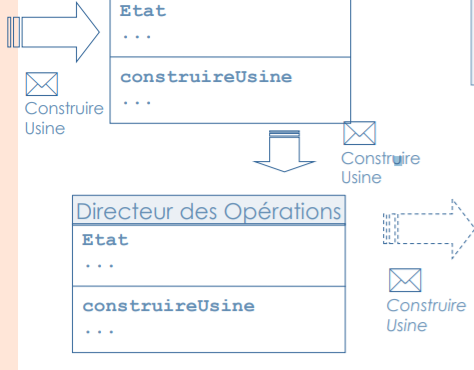
Aggrégation : Couplage fort. Une classe joue un rôle plus fort (relations maîtreesclaves).



Composition : Sorte d’agrégation dans laquelle les objets sont liés par leur cycle de vie : la disparition de l’un entraîne celle de l’autre.



Délégation: Un client communique avec une interface qui propage les questions à un ou plusieurs délégués



Multiplicité : précise le nombre d’instances qui participent à la relation.

Valeurs les plus courantes : 1 : un et un seul

0..1 : zéro ou un

M..N : de M à N (entiers naturels)

\* : de zéro à plusieurs

1..\* : plusieurs