

P00 : PRÉSENTATION



QU'EST-CE QUE LA POO ?

- Programmation Orientée Objet
- Consiste à définir des *objets* et à les faire interagir

UN PEU D'HISTOIRE

- Apparu pour la première fois dans les années 70
- Simula 67 suivi de Smalltalk étaient les premiers langages de POO

- La POO a connu une percée majeure avec le **C++** dans les années 80,
- D'autres langages sont apparus en 90 comme Java,
- Suivis du .NET et C# dans les années 2000

LES 5 PILIERS

On distingue 5 principes fondamentaux en POO :

- Objet et classe,
- Encapsulation
- Héritage
- Abstraction
- Polymorphisme

PREMIER PILIER : LES OBJETS ET CLASSES



LES OBJETS

- Un objet est une entité qui représente un élément,
- Il dispose de ses attributs et de ses méthodes,
- **Un attribut** : représente ce qui constitue l'objet,
- **Une méthode** : représente ce que peut faire l'objet.

EXEMPLE

Un marteau est caractérisé par :

- sa marque,
- son poids,
- son type (menuisier, electricien ...)

*Ce sont donc les attributs de notre
marteau*

EXEMPLE

Un marteau peut peut :

- Frapper,
- Déclouer






*Il disposera donc d'une méthode **frapper**
et une méthode **déclouer**.*

LES CLASSES

- Pour créer facilement différents objets, nous utilisons une classe,
- Une classe est comme un modèle qui nous permet de créer des objets,

- Tous les objets créés à partir d'une même classe auront les mêmes attributs,
- Cependant, la valeur de ses attributs ne seront pas obligatoirement identiques d'un objet à l'autre
- Les méthodes elles, seront identiques

EXEMPLE

Marteau	
	Marteau
	marque:
	poids:
	type:
	frapper()
	declouer()

- Ici, notre classe serait les plans de nos marteaux,
- Tous les marteaux construits à l'aide de cette classe disposeront des mêmes attributs.

A partir du même modèle, je suis donc capable de construire différents marteaux:



Chaque objet créé est une '*instance*' de la classe *Marteau*

EN RÉSUMÉ :

- Une classe est un type abstrait (ici, un marteau)
- Un objet est un type concret (le marteau de marque Stanley)

SECOND PILIER : L'ENCAPSULATION



DÉFINITION

- L'encapsulation stipule qu'il est préférable d'accéder/modifier un objet à l'extérieur de la classe
- On utilisera des méthodes spéciales (getter/setter) pour accéder aux attributs d'un objet
- Pour empêcher l'accès autrement que par les getter/setter, on utilisera la notion de *privacy*

PRIVACY

On distingue trois niveaux de protection des attributs :

- Public
- Protected
- Private

PORTÉE : PUBLIC

- Niveau de protection le plus bas
- Les attributs sont accessibles par l'objet en question
- Ils sont aussi accessibles par les autres objets

PORTÉE : PROTECTED

- Les attributs seront accessibles par la classe et les classes enfants
- Elles ne seront pas accessibles par les autres objets

PORTÉE : PRIVATE

- Niveau de protection le plus haut
- Les attributs ne sont accessibles que par les méthodes de l'objet
- Elle ne seront pas transmises via héritage

GETTER/SETTERS

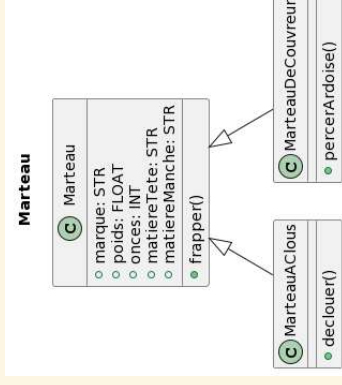
- Contrairement à un attribut qui est une simple valeur, un getter nous permet d'y intégrer une logique de traitement
- Grâce à cela, peu importe le contenu de notre objet, on peut afficher l'information attendue

Voir chapitre associé pour plus de détail !

TROISIÈME PILIER : L'HÉRITAGE

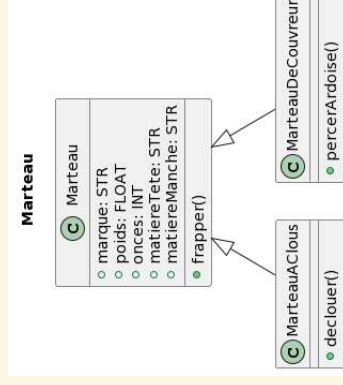


DÉFINITION



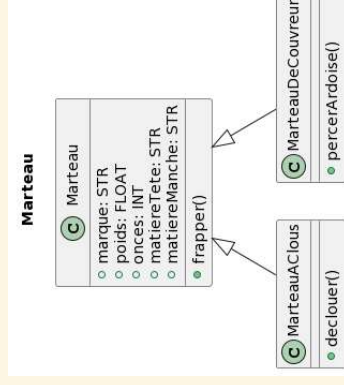
- Est un élément central de la POO
- Il nous évite de coder plusieurs classes similaires qui ont en commun attributs et méthodes

EXEMPLE



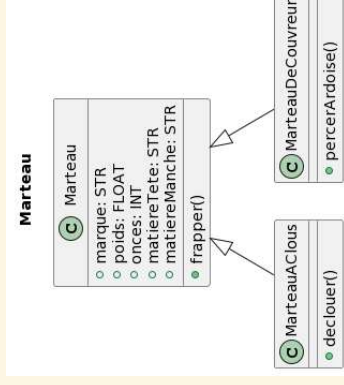
- Ici, un marteau a clous héritera de tous les attributs du marteau
- Il disposera cependant de sa propre méthode (déclouer)

SUPER CLASSE



*Super Classe désigne la classe parent
(ici Marteau)*

SOUS CLASSE



*Sous **Classe** désigne une classe enfant
(ici, MarteauAClous et
MarteauDeCouvreur)*

QUATRIÈME PILIER : L'ABSTRACTION



DÉFINITION

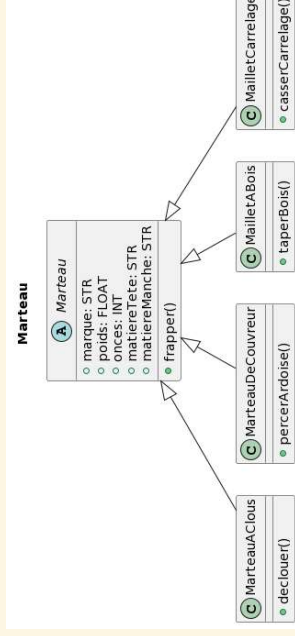
- Une classe abstraite est une classe qui ne peut pas avoir d'instance,
- Son unique but est de transmettre par héritage des attributs/méthodes communs à ses classes enfants

DÉFINITION

- On y représentera les attributs que les sous classes auront en commun
- Une classe qui n'est pas abstraite est appelée :

classe concrète

EXEMPLE



- Ici, **Marteau** est notre classe abstraite
- Signalé par un **A** sur le diagramme
- Il nous est impossible de créer un "Marteau" à proprement parler

CINQUIÈME PILIER : LE POLYMORPHISME



DÉFINITION

- Permet de surcharger/redéfinir une fonction héritée ou non
- En d'autres termes, la fonction aura le même nom mais ne fera pas la même chose

DÉFINITION

- Procéder ainsi nous évite de multiplier les noms de fonctions
- L'idée étant que les méthodes polymorphées fassent la même chose, mais pas de la même manière

DON'T PANIC

*Tout deviendra plus clair lorsque le sujet
sera abordé en détail.*

LA SUITE

Par ici !

