Algo et prog Q2

Créateur	Robin Gillard (el_pablo__)

```
question
30 min pour se préparer + 5 minute de conversation(aide)
si bien compris la question code pendant 2h
si pas compris la question c ciao
```

Paradigme : représentation du monde dans un sens large (le fait de changer sa vision d'une chose)

l'implémentation c'est l'action qui va suivre une réflexion pour la concrétiser

Paradigme de programmation → manière de concevoir le code et un moyen pour formuler le problème dans un code informatique/imaginer sa solution et l'implémenter

Programmation procédurale : programmation impérative (ordre) et structuré (l'execution dans un ordre précis)

- Suite d'instructions avec structures de contrôle
- appel de procédures réutilisable
- s'inspire de l'architecture des ordinateurs

objets 1

- Attributs
- méthodes

ageUtilisateur = 22 est un objet de type integer par exemple

Par exemple l'ordinateur est un objet, il a des propriétés, peut réaliser des actions, on peut avoir plusieurs ordinateurs qui auront des caractéristique différente

un objet serra divisé en 3

- Objet (un nom)
- se trouve dans un état (propriétés)
- répond à des messages (comportements)

Attribut → variable de classe (nom, prénom, age, taille, ...)

Propriété → manière de manipuler des attributs (lecture seule, accès non autorisée,...)

avantage de la POO

- Lisibilité et maintenance du code
- code source réduit, modulaire, évolutif

Inconvénient

- L'apprentissage est plus complexe que la procédural car beaucoup de concepts

une classe → un peu comme un plan de conception, genre (humain)

objet → instance de classe (ex Johan, Erwin, Joakim,...)

Points a retenir

- Un objet est une entité qui possède un état (définit par ses **attributs**) et des comportements (définis par ses **méthodes**)
- une classe est une **description** d'un ensemble d'objets qui partagent les **mêmes attributs** et **méthodes**

- un objet est une **instance** de sa classe
- une classe peut admettre plusieurs **conceptions**

Un point désigne un emplacement et sa seule caractéristique c'est sa position

Méthode ⟶ fonction d'une classe (procédure aussi)

- c'est une action
- ex : manger(TypeNourriture, quantité)

Méthode qui ne renvoie rien → procédure

Méthode de classe ⟶ fonction d'une classe avec une forte appartenance à une classe

Méthode statique ⟶ fonction d'une classe mais indépendante de la classe dans laquelle elle est appelé (on n'utilisera pas l'objet sur laquelle la méthode est appelée)

▼ **Encapsulation**

Encapsuler nos donnée dans notre objet et il ne serra donc plus visible de l'extérieur (on protège l'attribut et on ne pourra plus y accéder normalement)

Obtenir la valeur d'un attribut ⟶ Méthode d'accès "Getter"

Modifier la valeur d'un/plusieurs attributs ⟶ "Setter"

L'encapsulation va permettre de définir le niveau de visibilité d'éléments de la classe. On définit le droit d'accès au données.

Les attributs d'une classe sont privés et ses méthode publiques

3 niveau de protection

- Public → accès par tout

- Protected → attribut privé accessible par la classe et tout ses “enfants” aussi (faire ca par défaut)
- Private → attribut privé accessible uniquement depuis la classe elle même

choisir le bon niveau

- Protection maximum du code depuis l’extérieur
- Sensibilité de chaque élément
- Impacts des niveaux d’accès à un élément sur le reste de la classe
- Identification des éléments qui ont besoin d’accès pour fonctionner

▼ **Interface**

C’est le contrat d’une classe et la définition de ses méthodes

(une classe avec que des méthodes)

ex : classe humain, avec nom, prénom, age → on est tous humain ici et on a une interface professeur avec une méthode enseigner et une étudiant avec méthode étudier et dès que l’un des humains prend la classe professeur acquerra la méthode enseigner et si classe étudiant acquerra méthode étudier, mais ce n’est pas parce qu’il l’a reçu que son utilisation et obligatoire.

Constructeur est la première méthode de votre classe

- Il fonctionne comme une fonction
- Il peut prendre des paramètres

Un constructeur est une méthode particulière de la classe, on utilisera le nom de la classe comme nom du constructeur. Il est appelé au moment de la

création de l’objet

Il peut être de 3 types différents :

- Constructeur par défaut → pas de paramètre (obligatoire, doit toujours être la)
- constructeur par recopie/de copie → qu’1 paramètre, en gros une photocopie des attributs
- constructeur paramétrique → on peut avoir un deuxième constructeur avec 1 paramètre qui va affecter tout les attributs

---

▼ Rappel

Encapsulation : boite noire en gros imaginons que je sorte ou rentre les train d’atterrissages, je n’ai pas toute les étapes juste je dis de le faire

Classe → plan (de construction par ex)

Object → instance de la classe

On peut dire qu’une voiture possède

- Marque
- Couleur

Vu qu’on possède c’est un attribut

On peut dire qu’une voiture peut faire

- Rouler
- klaxonner

Vu qu’elle peut faire qq chose c’est une méthode

Le qualificateur Statique → pas compris revoir

Délégation → Lorsqu’un objet va déléguer une méthode a un autre objet mais qui va fournir le même résultat

Association : A est lié à B
Ex : Une université emploie un professeur

L'unif a besoin d'un professeur. On ne peut pas instancier uni sans prof.

Mais le prof peut donner cours à plusieurs uni

Agrégation : A est composé de B ⇒ Indépendant

Ex : Ordinateur est composé d'un disque dur

On peut changer le dd sans perturber l'ordi mais l'ordi existe tjs
Ça peut se changer

Composition : forte liaison ⇒ Si j'en tue un l'autre meurt

Ex : Un livre est composé de plusieurs pages
Le livre n'existe pas si il n'y a plus de pages

Impossible d'avoir double multiple

EXO →

Voiture a 4 roues :

Composition (si les roues ne sont pas détachables et ne sont pas réutilisables avec une autre voiture)

▼ Héritage

- Animal → classe mère
- Panda → classe fille
- Cheval → classe fille ...
- La fille est une spécialisation de la mère

- Véhicule
  - Bateau

- Véhicule a roue
  - Voiture
    - 3 Portes
    - 5 Portes
  - Vélo

Défini un lien de parenté (attribut commun (souvent))

Une classe fille héritera donc de tous les attribut et méthode de la classe mère, mais elle peut aussi en recevoir des unique a elle-même

→ Voiture a roue mais Moto et camion aussi

→ Voiture a coffre mais pas camion ni moto

→ Moto a un guidon mais pas voiture ni camion

Etc ...

Extent → défini qu'une classe est fille d'une autre classe

Super → Référence dans la classe fille d'un attribut/méthode dans la classe mère

L'héritage peut être infini

Si notre fille a hériter d'une même méthode de plusieurs mère

- Modifier le comportement de celle ci (redéfinition → overriding)
- Ajouter des instructions particulières (surcharge → overloading)

Overloading → déclarer dans une même classe ou classe hérité deux méthodes de même nom mais avec des paramètres différents

- Même nom de méthode
- Paramètre différents (soit sur le nom ou types)
- Le type de retour n'est pas pris en compte

Redéfinition → Réécrire dans une classe héritée une méthode qui a le même nom et les mêmes paramètres

- Même nom de méthode
- Même type de retour
- Même paramètre (nombre et type)

▼ Polymorphisme

un changement qui fait qu'on est soit un chat, vache, chien, ..

En POO c'est le fait d'avoir une même fonction, qui auront les méthodes de même nom mais qui auront des comportement différent (parler → aboyer, miauler, ..)

C'est la capacité du système a choisir dynamiquement l'opération effectuer (SePrésenter → même méthode mais réponse différente en gros)

Surcharge se fait a la compilation

Polymorphisme se fait pendant l'exécution

- Polymorphisme ad hoc → classe image, classe lien et classe complexe doit afficher des info, mais du coup peut importe le type de l'objet
- Polymorphisme d'héritage → en échec → on est des pièce et on possède tous la méthode mouvement qui grâce au polymorphisme d'héritage pourra nous permettre de réaliser le bon mouvement dans le bon cas
- Polymorphisme paramétrique → le système décidera selon les paramètre qu'on donne lors de l'appel la bonne méthode

▼ **Rappel**

Association : une classe qui en utilise une autre

Etudiant utilise prof pour apprendre et prof utilise élève pour apprendre a qqn

Composition : une classe qui en possède une autre (elles sont indépendantes)

Agrégation : Leur cycle de vie n'est pas lié (liaison faible)

Composition : Cycle de vie lié (Liaison forte)

Héritage : enfant de ..

La fille serra une spécialisation de la mère

La mère serra une généralisation de la fille

La flèche va de la fille vers la mère /!\

Composition = best

- Renseigne immédiatement sur les intentions du programmeur
- N'induit pas de couplage fort
- Favorise un design souple et maintenable

POWERPOINT, PAGE 170 A 176 TRES IMPORTANT (héritage, etc..)

Classe abstraite → classe qu'on n'instancie jamais

Elle possède des attributs ou méthode abstraite

- Réduire complexité
- Évite la duplication de code
- Aide a renforcer la sécurité d'une application ou d'un programme car seuls les détails importants sont fournis

Une méthode abstraite est une méthode qui est déclarée dans la classe abstraite, mais elle doit être implémentée par les classes dérivées. Les méthodes abstraites n'ont pas de corps ; elles se contentent de définir la signature de la méthode

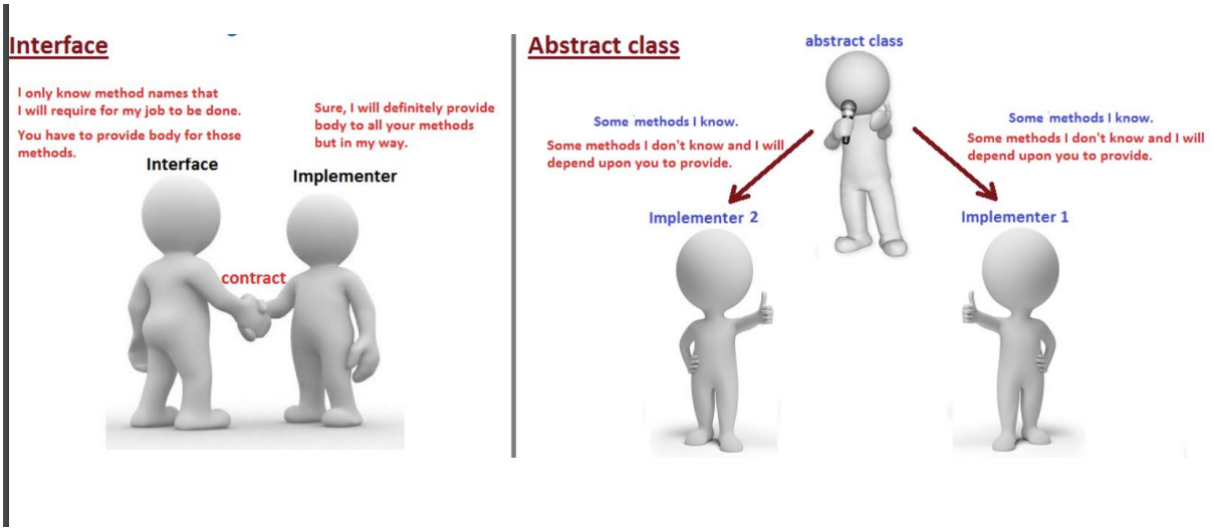
Les classes abstraites sont utilisées comme une base pour d'autres classes. Elles permettent de définir un modèle (ou contrat) que toutes les sous-classes doivent suivre

REVOIR LES SLIDE

Interface → contrat d'actions mais seulement abstraite (méthode abstraite), Pas d'attribut

Lorsqu'une classe implémente une interface, elle indique ainsi qu'elle s'engage a fournir une implémentation(un corp) pour chacune des méthode abstraite

Interface : je sais seulement le nom des méthode que j'aurai besoin pour que le boulot soit fait, toi tu dois le mettre en application (lui donner un corp)



Contrat commun → Les interfaces établissent un contrat que différentes classes peuvent suivre, permettant à ces classes d'interagir avec le reste du

système de manière prévisible.