Programmation orientée objet

Programmation procédurale

- Centrée sur les procédures
- Décomposition des fonctionnalités en procédures
- Procédures s'exécutent séquentiellement
- Les données sont indépendantes des procédures
- Les données à traiter sont passées en arguments aux procédures

Critique de l'approche procédurale

- Ressemble peu à notre schème de penser
- Vue algorithmique d'un programme donc très près du langage de la machine
- Le programmeur doit faire un effort cognitif pour interpréter ce qu'il veut modéliser pour être "compris" par la machine
- Peu intuitif lors de l'analyse d'un code existant

Critique suite…

- Tend à générer du code "Spaghetti"
- La maintenance et l'ajout de nouvelles fonctionnalités demandent de modifier ou d'insérer des séquences dans ce qui existe déjà
- Peut devenir complexe très rapidement
- Modularité et abstraction absente (ou presque)
- Réutilisation ardue => "Couper-coller" = DANGER!
- Travail d'équipe difficile (peu modulaire), donc la qualité du code en souffre

Programmation orientée objet

- Centrée sur les données
- Tout tourne autour des "objets" qui sont de petits ensembles de données représentants leurs attributs
- Ex.: Un joueur a un **nom,** un **score,** une **adresse courriel**, etc
- Ex.: Un compte de banque a un **propriéaire**, un **solde**, un **historique**, etc.

Programmation orientée objet

- Ces objets, en plus des données ont aussi des comportements: les **méthodes**
- Ex.: Un joueur peut **jouer son tour**, **écrire** un message, etc
- Ex.: Un compte de banque peut **s'ouvrir**, **traiter une transaction** (dépôt, retrait), **se fermer**, etc.

Classe et Objet

La classe == le moule

L'objet == la figurine

Encapsulation des attributs et méthodes

Attributs

- Aspect statique de l'objet
  - Couleur
  - Taille
  - Poids
  - Nom
  - Numéro,
  - Etc.

Méthodes

- Aspect dynamique de l'objet
  - S'afficher
  - S'initialiser,
  - Se modifier,
  - S'envoler,
  - Exploser,
  - Etc.

Niveaux d'encapsulation (public et privé)

- Public : accessible de partout dans le programme
- Privé: accessible seulement dans la portée (scope) de la classe
- En résumant grossièrement, les attributs sont privés et les méthodes publiques

Deux motifs de l'encapsulation privée

- Empêcher le programmeur-client de la classe d'avoir accès à ce qu'il n'a pas besoin.
  C'est un service qu'on lui rend: il peut ignorer les détails et se concentrer sur les services de la classe
- Le développeur de la classe peut modifier ce qui est privé sans déranger les programmeurs-clients de la classe