Synthèse

Programmation Avancée

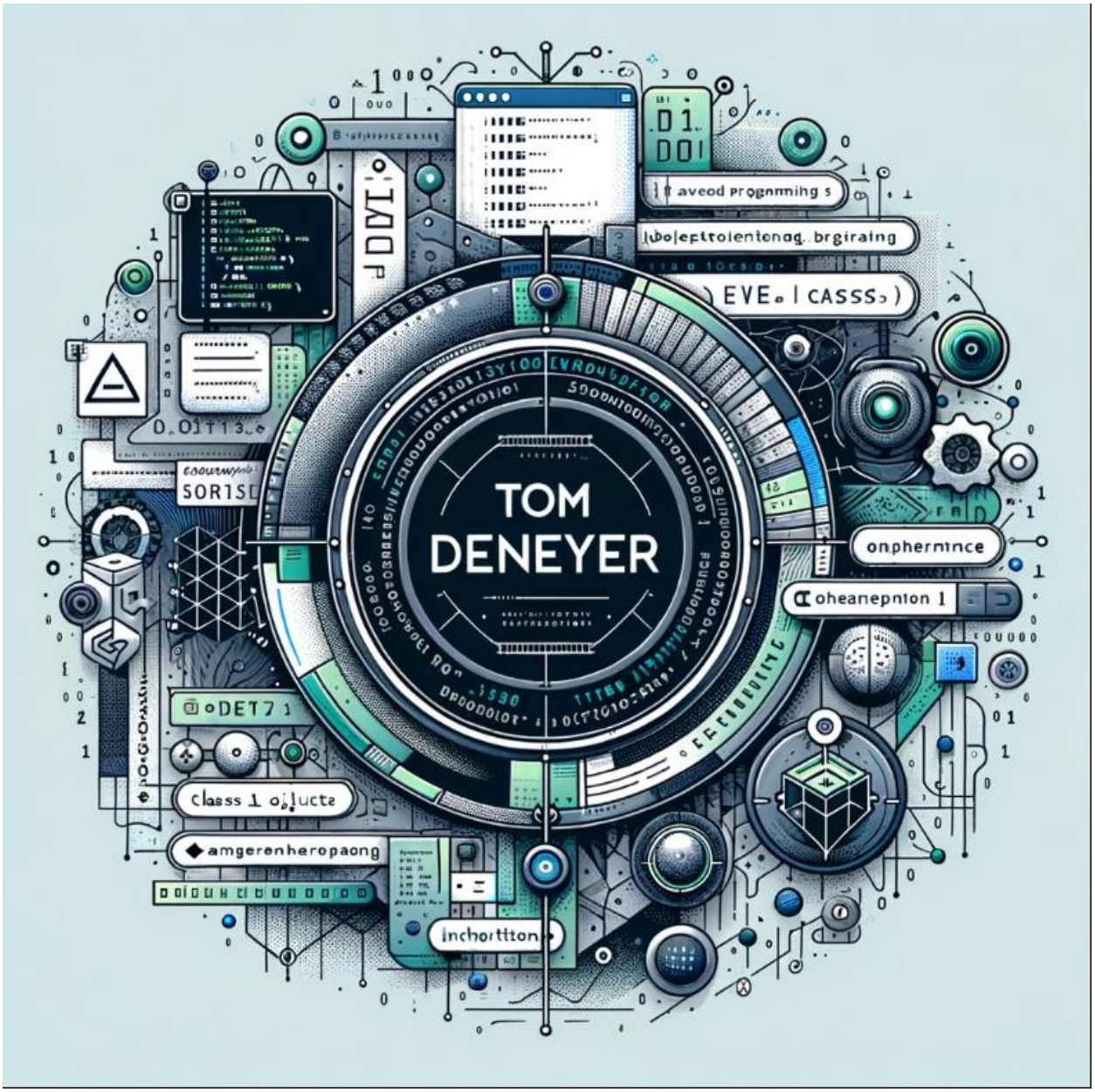

Table des matières

Généralités

- Paradigme informatiques
  - Facon de concevoir le code
  - Facon de penser, réfléchir à la résolution d'un problème
- Implémentation
  - Suite à la réflexion à un problème, concevoir et mettre en place la résolution

Paradigmes objet et procédural

- Procédural = Structure impérative, étape par étape, structuré

  Une procédure : *Fonction qui prend des arguments (données) qui utilisent ces données.*

- POO = Orienté objet, données de l'objet, et manipulation de ceux-ci

  Création d'objets via une classe

  Méthode sur ces objets : équivalent d'une fonction

  Il faut simplifier les objets lors de leurs conceptions ➔ conceptualisation simplifiée (modélisation de l'objet)

  Les objets peuvent communiquer, interagir, avoir ses propres caractéristiques…

- Dans procédural ou POO, les données sont utilisées dans des fonctions (ou attributs dans des méthodes)…

Comparaison

Procédural :	POO :
A = 1	A = Point(0)
B = 2	B = Point(1)
distance (A,B)	B.distance(A)

POO

Objet

- Les objets possèdent des « propriétés » qui sont attribués pour chaque objet de la classe.
  Pour un humain, les propriétés sont le nom, prénom, âge, travail, etc…
- Les actions d'un objet (méthodes) impactent ses attributs ou les attributs d'un autre objet.
- Propriété = manière de manipuler un attribut (accès autorisée ou non…)
- Abstraction : Passer de plusieurs attributs, méthodes, etc… A une classe. Extraire les éléments abstraits vers une classe.

Classe

- Plan de conception.
- Instance de classe, élément qui sont créés, basées sur une classe (INSTANCIATION)
  Les instances de la classes (objets) partagent les même attributs (pas forcement les même valeurs…
- Le modèle de la classe, ses attributs, etc. Dépendent du contexte.
- Polymorphisme
  Possibilité de modifier le comportement d'une méthode dans différentes classes
- Spécification initiale
  Rendre du concret en informatique, genèse de l'application
- Modèles
  - De domaine : Descriptions objets du monde réels (partie de conception de l'informatique, l'utilisateur s'en fout)
  - D'application : Parties visible par l'utilisateur
- Architecture du système
  - Etablissement des stratégies de conceptions générales + prévision des ressources

Méthode

- Méthode
  Fonction de la classe, service de celle-ci, ce que l'objet peut faire.
- Méthode statique
  Qui n'utilise pas l'objet lui-même.
  Classe **utilitaire**
  Aucune instance de la classe, exemple une console (classe console) dans laquelle on print.
- Méthode de classe
  Fonction avec fort liens à une classe

Encapsulation

- Encapsuler les attributs = Protéger l'attribut, en définissant des méthodes pour modifier les attributs.
- Obtenir la valeur d'une « capsule » = **Getter**
- Modifier la valeur d'une « capsule » avec des sécurités, exemple if…= **Setter**

- Permet d'attribuer les droits d'accès et visibilités d'attributs.
- Par défaut, les attributs sont privés et ses méthodes publiques.
- 3 niveaux d'accès
    - Publique (+)
      Accessible par l'utilisateur et par la classe
    - Protégé (#)
      Uniquement accessible par la classe elle-même et ses classes héritées (enfants)
    - Privé (-)
      Méthodes et attributs avec ce mot clé sont accessibles uniquement depuis la classe elle-même.
    - En général, protégé pour les attributs, publique pour les méthodes.
    - Le niveau de protection impacte le reste de la classe

Interface

- C'est une classe appelée interface dans laquelle on ne définit que des méthodes qui sont vides.
  Une interface donne l'accès aux classes qui utilisent cette interface à utiliser les méthodes définies dans la classe non instanciée (l'interface elle-même.) Cette notion de lien est appelée le contrat qui donne la validité réponse/retour.

Constructeur

- Fonctionne comme une fonction (il peut prendre des paramètres) définie en première dans la classe. Méthode obligatoire pour pouvoir instancier la classe.
- Nom du constructeur = nom de la classe.
- 3 types différents :
    - Par défaut
      Pas de paramètre
    - Par recopie
      L'unique argument du constructeur c'est une instance de la classe pour le copier
    - Paramétrique (après le constructeur par défaut)
      Un constructeur qui prend en paramètre des arguments pour modifier des attributs de la classe.

Conceptualisation POO

- Abstraction ➔ Passer d'objet réels à un concept (exemple mathématique) ➔ Classe concept, abstraction en code.
- UML, conceptualisation du problème.

Association

- Un objet A est lié à un objet B
- Une classe A est définie par rapport à B
- Association symétrique ➔ A a besoin de B et B a besoin de A

Agrégation

- L'objet A est composté de l'objet B, Mais les objets sont indépendant. Donc si la classe A n'est pas valable, l'objet B l'est toujours. Dans les deux sens.

Composition

- l'objet A a besoin de B pour exister, à l'opposé de l'agrégation

Héritage

- Classe fille qui hérite d'une classe mère. Les spécificité de la mère sont valables dans les filles, et on ajoute des détails dans les filles (attributs spécifiques, méthodes…)
- La classe héritée, hérite de tous les attributs et méthodes de la classe mère
- Mots clefs
  - Extends :
    Permet de féfinir une classe comme fille (Voiture extends Vehicule)
  - Super :
    Permet d'accéder aux attributs et méthodes de la classe mère.
- Overloading
  - Appelée « surcharge », dans une même classe, il existe deux méthode de même nom avec des sémantiques différentes
  - Les paramètres des même nom de méthodes on des paramètres différents, donc en fonction des paramètre le programme choisis la bonne méthode
- La redéfinition
  - Appelé « Overriding », Consiste à définir le comportement d'une méthode selon le type de l'objet qui utilise la méthode.
  - Même nom de méthode et même paramètre mais change en fonction de la classe
  - ➔ redéfinir ce qui a déjà été définit

Polymorphisme

- Lié à l'héritage et la surcharge. Les méthodes de classes différentes mais de même noms on un comportement différent en fonction de la classe héritée.
- Polymorphisme ad hoc
  Fonctions de même nom, avec même fonctionnalité, dans des classes sans liens entre elles.
- Polymorphisme d'héritage
  Spécialisation de la méthode du même nom.
  exemple : pièces d'échec.
- Polymorphisme paramétrique
  L'appel à la méthode utilise dynamiquement la bonne méthode en fonction des paramètres.

UML

Généralités

- Standard de modélisation objet (visualisation via dessins)
- Depuis 2017 : UML 2.5.1
- Lien direct avec la POO
- Abstraction et Encapsulation (Langage + méthodologie holistique, visualiser l'ensemble)

- On commence les diagrammes, dès l'idée de la réalisation
  Et l'adapter au fur et à mesure
- Si UML mal réfléchis, deviens très complexe
- Premier diagramme à faire : Diagramme de cas (fonctionnalités, actions, pov des utilisateurs.) et montrer au client

Types de diagrammes

- Diagramme de classe :
  Représentation des classes / attributs / méthodes…
  Souligne la structure et la conception du système
- Diagramme d'objet :
  Montre des instances de classe à un moment donné.
- Diagramme de cas :
  Représentation des services, actions fournies. POV utilisateurs
  Séquence d'action du système
  Habituellement décrit par un verbe + nom
- Diagramme de Paquet :
  Représente les fonctionnalités du système et leurs interactions avec utilisateurs + système
- Diagramme de composant :
  ??
- Diagramme de séquence :
  Quelle action à quel moment ?
- Classement de ces diagrammes
  - Deux grands types : Structurel et comportemental
    (voir schema cours à placer ici slide 22-24)

Etapes modélisation

- Définir périmètres du système
- Apports extérieurs **ACTEUR**
- Acteur = type stéréotypé qui représente un rôle joué par une personne ou une chose qui interagit avec le système
- Une personne physique peut avoir plusieurs rôles
  - Acteur = utilisateurs ou entités **externes** qui interagissent
  - Acteurs principaux (catégorie…), secondaire (ceux qui effectuent des taches secondaires), matériel externe, systèmes.
  - Représentation visuel par un dessin (simple)
- Ecrire un cas d'utilisation
  - Ovale comportant le verbe + nom qui décrivent l'action, lité à un acteur
- On regroupe le ou les cas d'utilisation dans un grand rectangle avec un titre (le système)
- Relation multiplicité (ex : un ou plusieurs acheteurs qui achètent un ou plusieurs articles.
- Cardinalité :
  - 0..1 : 0 ou 1 fois
  - 1 : 1 et une seule fois
  - * : zéro ou plusieurs
  - 1..* : un ou plusieurs

  - M..N : entre M et N fois
  - N : N fois
- Extension
  - Enrichir un cas (optionel)
  - Via le mot « extend »
  - Equivalent d'un « if » car via une condition
- Obligation : "include"

Réaliser un exercice

1) Identifier Acteurs
2) Définir cas d'utilisation
3) Etablir les relations
4) Vérifier et valider
5) Conseils
   a. Simplicité
   b. Focaliser sur l'utilisateur
   c. Collaboration
   d. Grain le plus important, visualiser l'ensemble
   e. Ne pas réfléchir « comment » réaliser.

Modélisation d'un SI

- Analyse des processus métier
  (diagramme séquence)
- Architecture fonctionnelle
  comment le système fonctionne, verbalisation des actions.
- Architecture applicative
  Diagramme de classe, structure de l'application
- Architecture technique
  Technique de ce qu'il se passe, au niveau technique, avec un grain plus fin.

Outils GL (génie logiciel)

Outils de générateur de code à partir d'un diagramme
Il écrit les classes, etc...