Institut Polytechnique DEFITECH

SUPPORT DE COURS

INITIATION A LA PROGRAMMATION ORIENTEE OBJET

Niveau : Première année Licence Informatique

Option : Tronc Commun

Dispensé par : AMEVOR Kossi A.

Année académique 2023-2024

Chapitre 2 :

Encapsulation et Surcharge

I - L'ENCAPSULATION

➢ Définition

L'encapsulation est un mécanisme consistant à rassembler les données et les méthodes au sein d'une structure en cachant l'implémentation de l'objet, c'est-à-dire en empêchant l'accès aux données par un autre moyen que les services proposés.

I - L'ENCAPSULATION

➢ Définition

Cette liste de services exportables est appelée l'interface de la classe et elle est composée d'un ensemble des méthodes et d'attributs dits publics (**Public**).

Les méthodes et attributs réservés à l'implémentation des comportements internes à l'objet sont dits privés (**Private**). Leur utilisation est exclusivement réservée aux méthodes définies dans la classe courante.

I - L'ENCAPSULATION

➢ Implémentation

L'encapsulation est réalisée à l'aide de mots clés qui permettent de définir des niveaux de visibilité d'une classe et de ces éléments (attributs et méthodes).

Les attributs et les méthodes sont précédés lors de la déclaration par l'un des modificateurs de visibilité suivants :
« **public** », « **private** », « **protected** » et **Néant**.

I - L'ENCAPSULATION

➢Niveaux de visibilité

Ces modificateurs de visibilité définissent les droits d'accès aux données selon que l'on y accède par une méthode de la classe elle-même, d'une classe héritière, ou bien d'une classe quelconque.

Il existe quatre niveaux de visibilité:

I - L'ENCAPSULATION

➢Niveaux de visibilité

- Une méthode, classe ou attribut non précédés par un modificateur de visibilité explicite (**Néant**) ne vont être visibles qu'à l'intérieur de même package. C'est-à-dire seules les classes de même package peuvent accéder aux attributs et méthodes de classes « amies ».

Ce modificateur de visibilité est aussi appelé « modificateur de **package** » ou modificateur « **freindly** ».

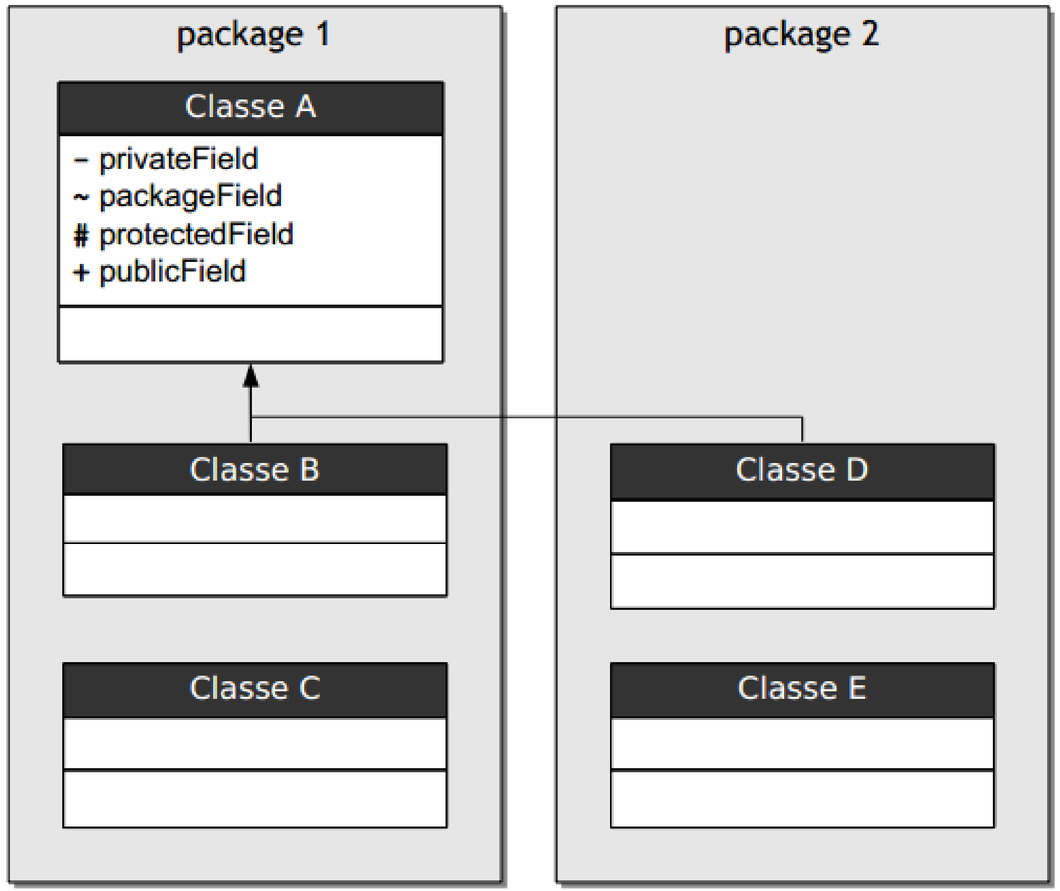
I - L'ENCAPSULATION

➢Niveaux de visibilité

- Une méthode, classe ou attribut sont déclarés comme publiques « **public** » s'ils doivent être visibles à l'intérieur et à l'extérieur quelque soit leur package.
- Une méthode ou attributs sont définis comme étant privés « **private** » s'ils sont accessibles uniquement par les méthodes de la classe en cours. Ils ne sont pas accessibles ailleurs.
- Une méthode ou attribut sont définis comme protégés « **protected** »s'ils ne peuvent être accessibles qu'à travers les classes dérivées ou les classes de même package.

I - L'ENCAPSULATION

➢Niveaux de visibilité



Visibilité des champs :

	Classe A	Classe B	Classe C	Classe D	Classe E
privateField	V				
packageField	V	V	V		
protectedField	V	V	V	V	
publicField	V	V	V	V	V

I - L'ENCAPSULATION

➢Tableau récapitulatifs des droits d'accès

- Modificateurs d'accès des classes

Modificateur	Signification pour une classe
Public (+)	Accès toujours possible.
Néant (~)	Accès possible depuis les classes du même paquetage.

- Modificateurs d'accès pour les membres et les classes internes

Modificateur	Signification pour une classe
Public (+)	Accès possible partout où la classe est accessible.
Néant (~)	Accès possible depuis toutes les classes du même paquetage.
Private (-)	Accès restreint à la classe où est faite la déclaration.
Protected (#)	Accès possible depuis toutes les classes de même paquetage ou depuis les classes dérivées.

I - L'ENCAPSULATION

➢ Avantages

L'encapsulation permet de :

- ❖ Garantir l'intégrité des données contenues dans l'objet;
- ❖ Modifier les structures de données internes sans modifier l'interface de celle-ci;
- ❖ Ajouter aisément des règles de validation;
- ❖ Offrir une interface orientée services et responsabilités;
- ❖ Simplifier la maintenance globale de l'application,

II – SURCHAGE DE METHODES

➢ Définition

La surcharge est la possibilité d'utiliser dans une même classe, le même nom pour déclarer des services différents mais qui ont la **même sémantique**.

Elle permet de définir plusieurs fois une même méthode/constructeur avec **des paramètres différents**.

II – SURCHAGE DE METHODES

➢Mise en œuvre

- Le mécanisme de surcharge permet de réutiliser le nom d'une méthode déjà définie, pour une autre méthode qui en différera par ses paramètres.

- En fonction du type et de nombre de paramètres lors de l'appel, la méthode correspondante sera choisie.

- La méthode surchargée doit conserver la même "intention sémantique".

II – SURCHAGE DE METHODES

➢Mise en œuvre

- La surcharge peut se faire sur une méthode définie localement ou héritée.

- La surcharge de méthodes peut être empêcher (dans les classes héritières) par le mot clé « **final** ».