Recherche a faire en java :

execice :relation entre les classes en poo java

commentaire java docx

les modificateurs d'acces en java

relation entre les classes en poo java :

les différentes relations qu'il est possible de définir entre différentes classes en poo java, en commençant avec la composition et l'agrégation, deux relations qui vont créer une dépendances entre classes

la composition :

*On utilise donc un objet pour en définir un autre. Toutes les instances de la classe*A*sont*composées*de deux instances de la classe*B*. Il s'agit de la première relation entre objets que l'on découvre :*la relation de composition

Celle-ci s'appelle également la relation has-a. Il s'agit d'une relation qui lie deux classes. On dit qu'une classe AA est composée à partir d'une classe BB, ce qui signifie qu'une instance de la classe AA aura une variable d'instance de type BB.

Celle-ci est représentée par une flèche se terminant par un diamant et va de l'objet principal vers celui dont il est composé. On peut également indiquer à l'aide d'un entier le nombre de composants utilisés.

Exemple :

Supposons qu'on souhaite écrire un programme qui manipule des voitures, par exemple pour une application qui permette à des clients de construire la voiture de leur rêve via une site Internet en choisissant le type de roues, de volant et de sièges. On définirait par exemple les classes Car, Wheel, SteeringWheel et Seat. Une voiture étant composée de roues, de volant et de sièges, on aura les relations Car est composée à partir de Wheel, SteeringWheel et Seat.

NB :

Dans la relation de composition , le lien entre les classes liées entre elles est très fort. Cela signifie que lorsque l'instance de la classe composite disparait de la mémoire, les instances composées disparaissent également. Dans l’exemple donné si les roues ou le volant ou disparait la voiture n’existe plus.

l'agrégation :

Une relation plus générale que la composition est l'agrégation. Il s'agit donc d'une forme généralisée de la composition, mais sans l'appartenance. Des objets vont pouvoir être agrégés, mais sans devenir fortement liés à un autre objet.

Exemple :

Si on veut écrire un programme pour gérer les communes d'un pays, on pourrait avoir une classe City pour représenter une ville, celle-ci étant dirigée par un bourgmestre instance d'une classe Citizen représentant un citoyen

La différence entre composition et agrégation est donc que dans le deuxième cas, les deux objets sont moins liés, puisque la la suppression d'une instance de la classe City n'entrainera pas la suppression des objets agrégés.

cette relation d'agrégation, est représentée par un losange creux et vide au milieu.

Relation uses:

Enfin, une autre relation possible entre deux classes est la relation uses. Cette relation est très générale et indique simplement qu'une classe utiliseune autre classe. Elle peut simplement créer une instance d'une autre classe ou avoir une méthode qui renvoie un objet du type d'une autre classe ou encore avoir une méthode qui prend en paramètre un objet du type d'une autre classe. Les relations de composition et d'agrégation sont évidemment des cas particuliers de la relation uses

Contrairement à la composition et l'agrégation qui se traduisent par une relation entre les objets, la relation *uses* peut ne s'appliquer qu'entre deux classes.

NB :

La relation *uses* est représentée par une flèche normale qui part de la classe qui utilise vers la classe utilisée.

commentaire java docx :

Dans un programme, les commentaires contribuent à rendre le programme plus lisible

En Java, il existe trois types de commentaires :

1. Commentaires sur une seule ligne.
2. Commentaires multi-lignes.
3. Commentaires sur la documentation.

Commentaires sur une seule ligne:

Un programmeur de niveau débutant utilise principalement des commentaires sur une seule ligne pour décrire les fonctionnalités du code. Ce sont les commentaires les plus faciles à taper.  
Syntaxe:

//(comments here)

Commentaires multilignes :

Pour décrire une méthode complète dans un code ou un extrait complexe, des commentaires sur une seule ligne peuvent être fastidieux à écrire, car nous devons donner ‘//’ à chaque ligne. Donc, pour surmonter cela, des commentaires sur plusieurs lignes peuvent être utilisés.  
Syntaxe:

/\*comments starts

…continue

…

…continue

\*/comment ends

Nous pouvons également réaliser des commentaires sur une seule ligne en utilisant la syntaxe ci-dessus, comme indiqué ci-dessous :

/\*comment here\*/

Commentaires sur la documentation

Ce type de commentaires est généralement utilisé lors de l’écriture de code pour un projet/progiciel, car il permet de générer une page de documentation de référence, qui peut être utilisée pour obtenir des informations sur les méthodes présentes, ses paramètres, etc.  
Par exemple [http:// docs.oracle.com/javase/7/docs/api/java/util/Scanner.html](http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html) est une page de documentation générée automatiquement qui est générée à l’aide de commentaires de documentation et d’un outil javadoc pour traiter les commentaires.

/\*comments starts

\* tags are used in order to specify a paramemeter

\*or method of heaving

\*html tags can be used

\*/comment ends

Il y’a beaucoup de balise disponible mais nous allons en citer:

|  |  |  |
| --- | --- | --- |
| **Étiqueter** | **La description** | **Syntaxe** |
| @auteur | Ajoute l’auteur d’une classe. | @author name-text |
| {@code} | Affiche le texte dans la police du code sans interpréter le texte comme un balisage HTML ou des balises javadoc imbriquées. | {@code texte} |

Disponible sur [Java](https://fr.acervolima.com/category/java/), [java-basics](https://fr.acervolima.com/category/java-basics/), [School Programming](https://fr.acervolima.com/category/school-programming/) / Par [Acervo Lima](https://fr.acervolima.com/author/jack_sparrow/" \o "Voir toutes les publications de Acervo Lima)

les modificateurs d'acces en java :

Le modificateur d'accès par défaut signifie que nous ne déclarons pas explicitement un modificateur d'accès pour une classe, un champ, une méthode, etc.  
Une variable ou une méthode déclarée sans modificateur de contrôle d'accès est disponible pour toute autre classe du même package. Les champs d'une interface sont implicitement public static final et les méthodes d'une interface sont par défaut public

Il y a deux types modifier en java: access modifiers et non-access modifiers.  
Les access modifiers dans java déterminent l'accessibilité (portée) d'un membre de données, d'une méthode, d'un constructeur ou d'une classe.  
Il y a quatre types de  java access modifiers :

1. private
2. (Défaut)
3. protected
4. public

Le tableau ci-dessous vous donne un aperçu de la façon d'utiliser les access modifier.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modificateur d'accès | **Accès en classe** | **Accès dans package** | **Accéder à l'extérieur de package par sous-classe seulement** | **Accéder à l'extérieur du colis et non dans la sous-classe.** |
| Privé | Y |  |  |  |
| Défaut | Y | Y |  |  |
| Protégé | Y | Y | Y |  |
| Public | Y | Y | Y | Y |

#### Modificateur d'accès privé

Les méthodes, les variables et les constructeurs déclarés privés ne sont accessibles que dans la classe déclarée elle-même.

Le modificateur d'accès privé est le niveau d'accès le plus restrictif. La classe et les interfaces ne peuvent pas être privées.

Les variables déclarées privées sont accessibles en dehors de la classe si des méthodes de lecture publiques sont présentes dans la classe. (**getters setters** )

NB:

L'utilisation du modificateur privé est la moyenne principale par laquelle un objet s'encapsule et cache les données du monde extérieur.

#### Modificateur d'accès publique – public

Une classe, une méthode, un constructeur, une interface, etc. déclarée publique peut être accessible à partir de n'importe quelle autre classe. Par conséquent, les champs, les méthodes et les blocs déclarés dans une classe publique sont accessibles à partir de toute classe appartenant à l'univers Java.

NB:

Toutefois, si la classe publique à laquelle vous essayez d'accéder se trouve dans un package différent, elle doit toujours être importée.  
En raison de l'héritage, toutes les méthodes et variables publiques d'une classe sont héritées par ses sous-classes.

#### Modificateur d'accès protégé – protected :

Les variables, méthodes et constructeurs déclarés protégés dans une superclasse sont accessibles uniquement aux sous-classes d'un autre package ou de toute classe du package de la classe des membres protégés.

Le modificateur d'accès protégé ne peut pas être appliqué à la classe et aux interfaces. Les méthodes et les champs peuvent être déclarés protégés. Toutefois, les méthodes et les champs d'une interface ne peuvent pas être déclarés protégées

NB :

L'accès protégé donne à la sous-classe une chance d'utiliser la méthode d'assistance(Helper) ou la variable, tout en empêchant une classe non liée d'essayer de l'utiliser.

Les règles suivantes pour les méthodes héritées sont appliquées

* Les méthodes déclarées publiques dans une super-classe doivent également être publiques dans toutes les sous-classes.
* Les méthodes déclarées protégées dans une superclasse doivent être soit protégées, soit publiques dans les sous-classes; elles ne peuvent pas être privées.
* Les méthodes déclarées privées ne sont pas du tout héritées, il n'y a donc pas de règle pour elles

https://developpement-informatique.com/article/239/modificateurs-dacces-java---public-private-protected-et-