

Le diagramme de classes

Table des matières

I. Contexte	3
II. Les classes et attributs	3
A. Les classes et attributs	3
B. Exercice : Quiz	6
III. Relations entre les classes	7
A. Relations entre les classes	7
B. Exercice : Quiz	12
IV. Essentiel	13
V. Auto-évaluation	13
A. Exercice	13
B. Test	14
Solutions des exercices	14

I. Contexte

Durée : 1 h

Environnement de travail : logiciel de modélisation UML (plusieurs possibles)

Contexte

Avant de construire une maison, il est important de dessiner les plans de la future habitation. Il en va de même pour la programmation orientée objet, la création d'une base de données ou la conception d'un projet informatique. Nous concevons d'abord « *un plan* » appelé diagramme de classes.

Le diagramme de classes est une méthode de modélisation qui permet de représenter graphiquement les entités d'un système informatique et les relations qui existent entre elles. Que vous vouliez devenir ou que vous soyez développeur, architecte logiciel, chef de projet ou encore analyste fonctionnel, le diagramme de classes est un outil essentiel pour concevoir, documenter et communiquer sur vos projets informatiques.

II. Les classes et attributs

A. Les classes et attributs

Définition Classe

Les classes dans un diagramme de classes permettent de modéliser la structure d'un système orienté objet et les relations entre les objets. Elles sont essentielles pour la conception et la planification d'un système, et peuvent être utilisées pour communiquer efficacement les idées entre les membres de l'équipe de développement.

Il existe aussi ce qu'on appelle une classe abstraite dans le cadre de l'utilisation de la notion d'héritage, notion que nous aborderons un peu plus loin. Dans la programmation objet, lorsque nous définissons une classe abstraite, nous ne pouvons pas l'instancier, c'est-à-dire, nous ne pouvons pas créer des objets de cette classe.

Définition Attributs et méthodes

Un attribut est une variable qui contient des données associées à un objet d'une classe. Les attributs décrivent les caractéristiques des objets de la classe et peuvent être de différents types tels que des entiers, des chaînes de caractères, des booléens, des tableaux ou des objets d'autres classes. Les attributs sont définis à l'intérieur de la classe et peuvent être accessibles et modifiables à partir d'autres parties du code, en utilisant des méthodes appropriées.

Une méthode est une fonction qui est définie à l'intérieur d'une classe et qui permet de décrire le comportement des objets de cette classe. Elle peut être appelée pour effectuer des opérations sur les attributs de la classe ou retourner des résultats. Les méthodes peuvent également prendre des arguments en entrée et retourner des valeurs en sortie.

Une classe

Voici comment se profile la modélisation d'une classe :

Nom de la classe
Les attributs
Les méthodes()

Le **nom** de la classe doit refléter la nature de l'objet ou du concept qu'il représente. Quant aux **attributs**, ceux-ci représentent les propriétés ou les caractéristiques de l'objet. Enfin, les **méthodes** vont contenir les différentes actions que l'objet peut effectuer.

Attention Syntaxe

La syntaxe d'une classe est importante, car elle définit la structure de la classe et permet aux développeurs de comprendre comment la classe fonctionne et comment elle peut être utilisée dans le système. Une syntaxe claire et cohérente peut améliorer la qualité du système, faciliter la maintenance et la documentation, et maintenir la compatibilité entre les différentes parties du système.

Conseil

Voici quelques conseils de syntaxe pour la modélisation d'une classe :

- Les noms de classe ne doivent pas contenir d'espaces, de signes de ponctuation ou même d'accents.
Pour cela, nous pouvons utiliser la méthode en utilisant la syntaxe « _ » ou en utilisant une majuscule au début de chaque mot.
Exemple : NomClasse ou nom_classe
Pour une classe abstraite, nous écrirons le nom de la classe en italique.
- Les noms des attributs doivent commencer par une minuscule, suivie de la syntaxe « : » et ensuite de son type qui peut être par exemple « *chaîne de caractère* » : (string), « *entier* » : (Int), etc.
- Les méthodes doivent être formatées d'une certaine manière, la méthode doit forcément commencer par une lettre minuscule, éviter les accents, puis finira par des parenthèses pour indiquer que plus tard il y aura une fonction à programmer.

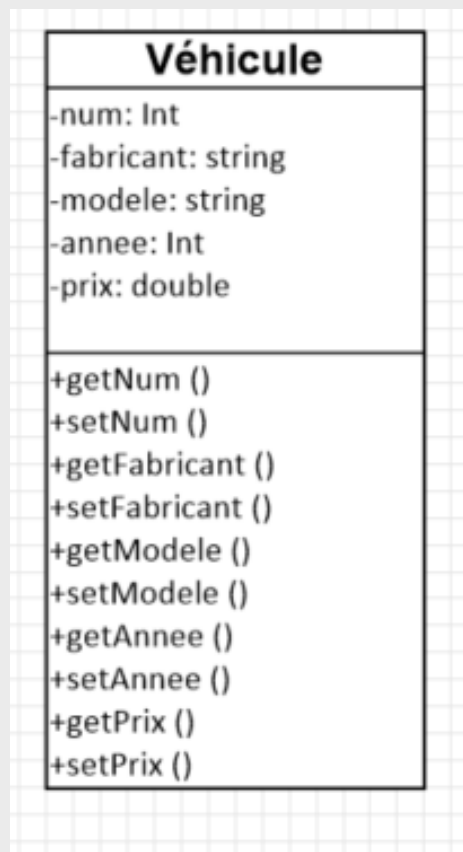
Visibilité d'un attribut et d'une méthode

La visibilité va permettre de définir l'accessibilité d'une méthode ou d'un attribut.

Cela va donc être notre rôle de définir quelle visibilité vont avoir nos méthodes et nos attributs.

Voici les différents types de visibilité :

- Le moins « - » qui va mettre nos méthodes et attributs en privé où toutes les autres classes ou sous-classes ne pourront pas accéder.
- Le plus « + » qui va mettre nos méthodes et attributs en public où toutes les autres classes et sous-classes pourront accéder.
- Le croisillon « # » qui va protéger nos méthodes et attributs qui donc ne pourront être accessibles que par la même classe ou ses sous-classes.
- Le tilde « ~ » qui va permettre que seules les classes ou sous-classes du même paquet aient accès à ces méthodes ou attributs.

Exemple

Dans cet exemple, la classe « *Vehicule* » représente un véhicule avec des attributs (tous en privé) tels que « *num* », « *fabricant* » qui est le fabricant du véhicule, « *modele* » qui est le modèle du véhicule, « *annee* » qui correspond à l'année de fabrication du véhicule, et le prix d'achat. Suivit des méthodes (en public) pour accéder et modifier ces attributs.

Les méthodes « *set* » et « *get* » sont des raccourcis de « *setters* » et « *getters* ».

La méthode « *set* » permet de modifier la valeur d'un attribut à partir d'un objet de la classe, dans cet exemple, la méthode « *+setModele()* » va nous permettre d'accéder à la valeur de l'attribut « *modele* » pour un objet de la classe « *Vehicule* ».

La méthode « *get* » permet d'accéder à la valeur d'un attribut à partir d'un objet de la classe, dans cet exemple, la méthode « *+getModele()* » va nous permettre d'accéder à la valeur de l'attribut « *modele* » pour un objet de la classe « *Vehicule* ».

Le « *num* » est un identifiant unique qui permet de distinguer chaque véhicule.

Choisir et définir une classe

Pour choisir les classes, les attributs et les méthodes dans un diagramme de classe, il est important d'identifier les entités ou les objets clés du système, de choisir des classes qui représentent ces objets, d'identifier les attributs qui décrivent les caractéristiques de l'objet, d'identifier les méthodes qui représentent les actions clés que les objets effectuent dans le système, et de réviser et d'affiner le diagramme de classe en fonction des besoins et des exigences du système.

Nous allons voir comment atteindre cet objectif.

Premièrement, il faut identifier les entités ou les objets du système : le premier pas dans la conception d'un diagramme de classe est d'identifier les entités ou les objets du système. Il est important de déterminer les objets clés qui interagissent dans le système et les relations entre ces objets.

Ensuite il nous faut choisir les classes : une fois que les entités ou les objets ont été identifiés, la prochaine étape consiste à choisir les classes qui représenteront ces objets. Nous nous rappelons que les classes sont des modèles qui représentent les caractéristiques et les comportements d'un objet. Il est important de choisir des classes qui représentent des concepts clés dans le système et qui sont suffisamment générales pour être utilisées dans plusieurs parties du système.

Prochaine étape, nous devons identifier les attributs : pour rappel, les attributs sont les propriétés d'une classe qui décrivent les caractéristiques de l'objet qu'elle représente. Il est important de choisir des attributs pertinents qui décrivent les caractéristiques importantes de l'objet. Les attributs peuvent être de différents types, tels que des attributs simples (comme un nom ou une date de naissance) ou des attributs complexes (comme une liste d'adresses ou de coordonnées).

Le point suivant est d'identifier les méthodes : nous nous rappelons que les méthodes sont les actions ou les comportements que les objets peuvent exécuter. Il est important de choisir des méthodes qui sont pertinentes pour les classes et qui représentent les actions clés que les objets effectuent dans le système. Les méthodes peuvent être des méthodes simples (comme une méthode de calcul de prix) ou des méthodes complexes (comme une méthode de génération de rapports).

Pour finir, réviser et affiner : une fois que les classes, les attributs et les méthodes ont été identifiés, il est important de réviser et d'affiner le diagramme de classe en fonction des besoins et des exigences du système. Il est également important de s'assurer que le diagramme de classe est cohérent et facile à comprendre.

B. Exercice : Quiz

[solution n°1 p.15]

Question 1

Les classes dans un diagramme de classes représentent des objets réels et des objets conceptuels.

- ☐ Vrai
- ☐ Faux

Question 2

Quelle est la bonne syntaxe pour écrire une méthode ?

- ☐ méthodeClasse()
- ☐ +methodeClasse()
- ☐ +méthodeClasse()

Question 3

Quel symbole permet d'attribuer une visibilité privée à un attribut ou une méthode ?

- ☐ Le symbole « + »
- ☐ Le symbole « - »
- ☐ Le symbole « # »
- ☐ Le symbole « ~ »

Question 4

Qu'est-ce qu'une méthode dans une classe ?

- ☐ Une propriété ou une caractéristique de l'objet
- ☐ Une action que l'objet peut effectuer
- ☐ Une instance de la classe

Question 5

Qu'est-ce qu'un attribut dans une classe ?

- ☐ Une propriété ou une caractéristique de l'objet
- ☐ Une action que l'objet peut effectuer
- ☐ Une instance de la classe

III. Relations entre les classes

A. Relations entre les classes

Les relations entre les classes sont une notion essentielle dans la conception de systèmes informatiques. Elles permettent de modéliser les liens et les interactions entre différentes entités, ce qui est crucial pour comprendre le fonctionnement d'un système et pour en assurer la maintenance et l'évolution. Maintenant nous allons voir les différentes relations qui peuvent exister entre les classes, ainsi que leurs implications et leurs avantages.

Définition Les associations

L'association est une relation entre deux classes qui permet de décrire une connexion sémantique entre ces dernières. Elle peut être unidirectionnelle (dans un sens), bidirectionnelle (dans les 2 sens) ou asymétrique (elle interdit une association). Les associations peuvent être qualifiées par des propriétés, appelées des rôles, qui précisent la nature de l'association. Les associations peuvent également avoir une cardinalité qui définit le nombre d'objets d'une classe qui peuvent être associés à un objet de l'autre classe. Les associations sont souvent utilisées pour modéliser des relations de dépendance, de composition ou d'agrégation entre les classes.

Définition Les cardinalités

Les cardinalités servent à décrire les relations entre les classes en indiquant le nombre d'instances d'une classe qui peuvent être associées à une autre classe.

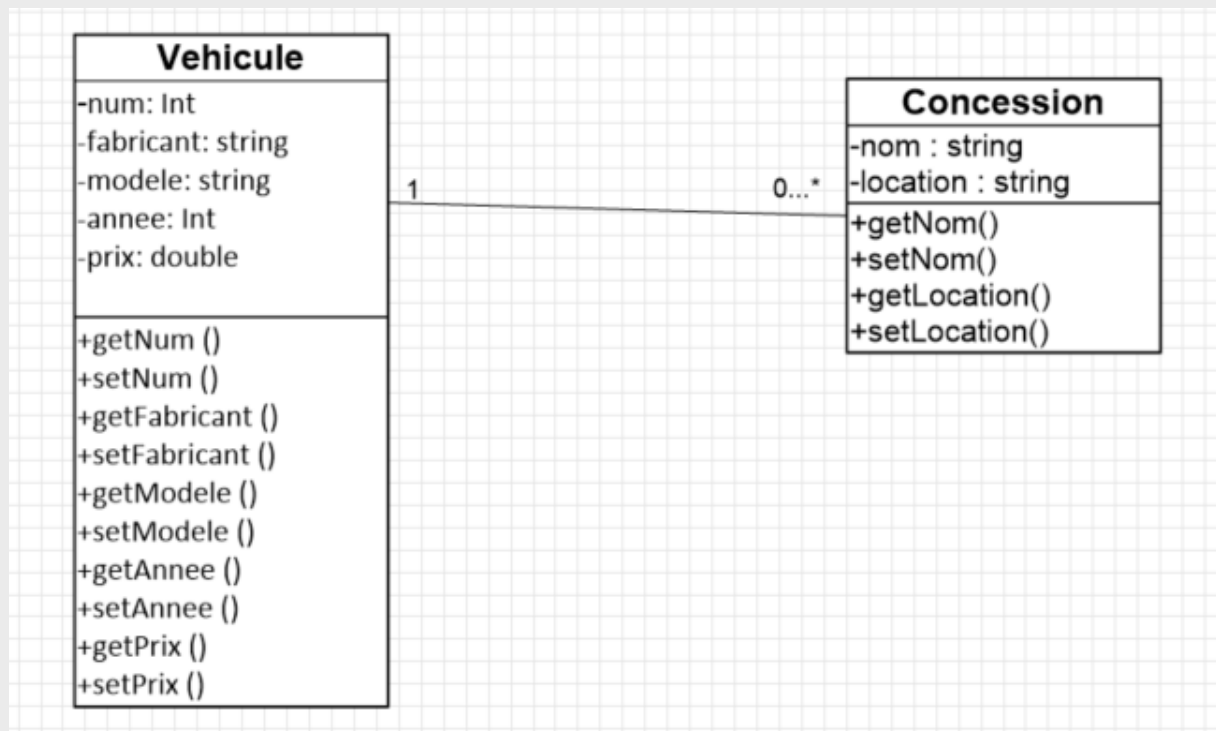
Dans un diagramme de classes, les cardinalités sont représentées par des nombres ou des symboles spéciaux placés près des extrémités des lignes qui relient les classes. Les cardinalités sont souvent utilisées pour décrire les relations d'association, de composition ou d'agrégation entre les classes.

À noter qu'on n'en met pas dans les relations d'héritage.

Voici les différentes cardinalités :

- « 0...1 » : relation optionnelle,
- « n » qui va représenter un nombre bien spécifique selon l'utilisation demandée,
- « 0...* » : de minimum 0 à plusieurs,
- « 1...* » : de minimum 1 à plusieurs,
- « m...n » qui va représenter une plage d'un numéro bien spécifique à un autre.

Exemple Une association basique



La relation entre les classes est définie par une association basique entre les classes « *vehicule* » et « *concession* ». Cette association permet de modéliser le fait qu'un véhicule peut être vendu par une concession automobile. La classe « *concession* » peut stocker une liste de véhicules à vendre, tandis que chaque véhicule peut être associé à une concession automobile qui le vend. Dans cet exemple, un véhicule peut être vendu par une et une seule concession, et une concession peut vendre zéro ou plusieurs véhicules.

Associations n-aire

Il est possible de créer une association entre plus de 2 classes appelées association n-aire, elle sera représentée par un losange rejoint par un lien venant de chaque classe reliée. Dans ce cas-là, il est obligatoire qu'il y ait toutes les cardinalités pour chaque association.

Une association n-aire est une association entre trois classes ou plus dans un diagramme de classes. Elle représente une relation complexe entre ces classes, où chaque classe est liée à plusieurs autres classes par une relation de type « *plusieurs à plusieurs* ». Une association n-aire est souvent utilisée pour modéliser des relations complexes entre des classes qui ont des interactions et des dépendances mutuelles.

Elle est souvent utilisée dans des domaines tels que la finance, la logistique ou la gestion de projet, où il est important de modéliser des relations complexes entre plusieurs entités.

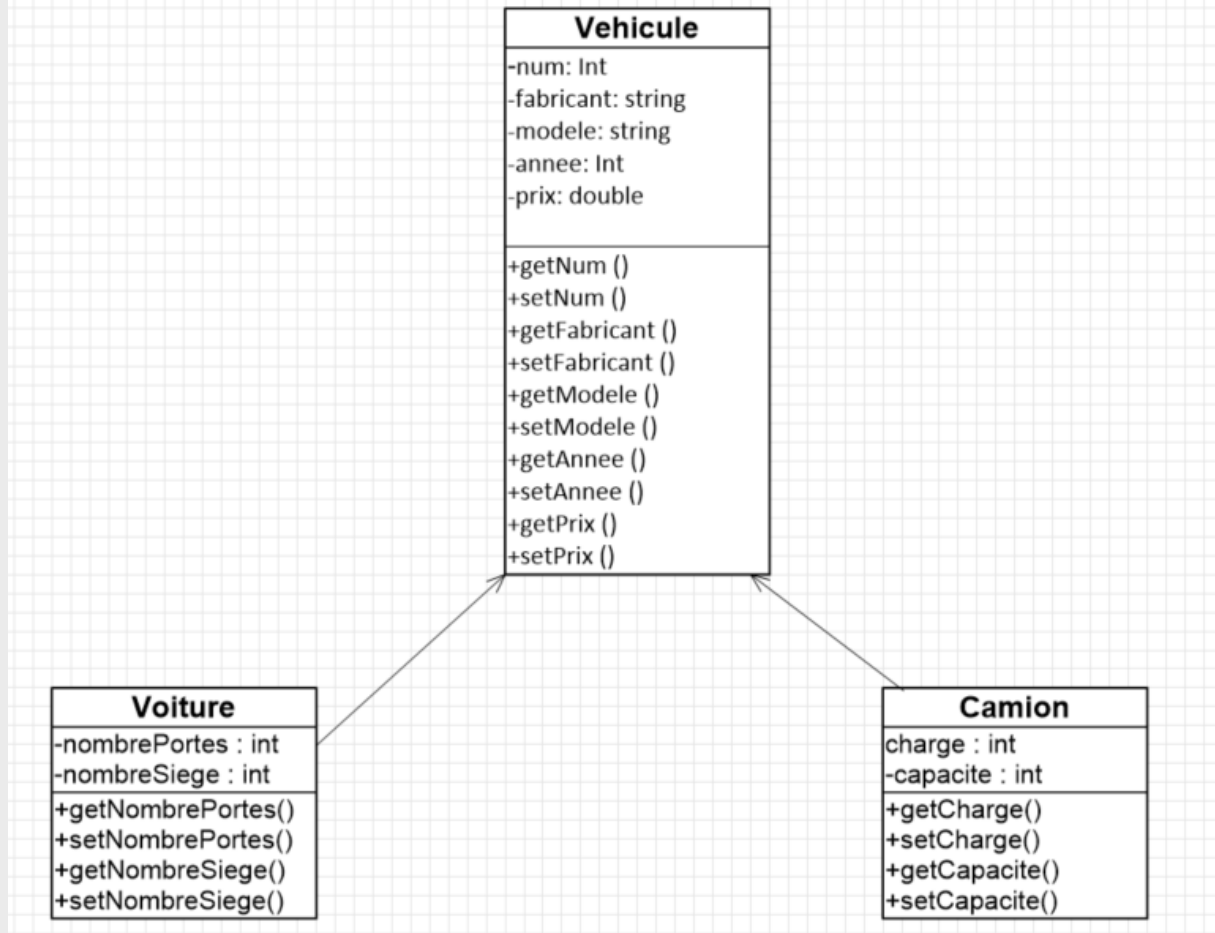
En utilisant une association n-aire, il est possible de décrire les relations et les interactions entre plusieurs classes de manière plus précise et plus détaillée qu'avec une simple association binaire. Elle peut également faciliter la compréhension du modèle de données en permettant de visualiser clairement les relations entre les différentes classes.

L'héritage

L'héritage est une relation entre deux classes qui permet à une classe (la classe fille) d'hériter des propriétés et des méthodes d'une autre classe (la classe mère). L'héritage permet la réutilisation du code et d'éviter la redondance de la définition de propriétés et de méthodes communes à plusieurs classes, mais aussi de simplifier la conception et

d'améliorer la lisibilité du code. Il existe différents types d'héritage, tels que l'héritage simple, multiple, hiérarchique, etc. Cependant, l'utilisation excessive de l'héritage peut entraîner des problèmes de maintenance et de complexité du code.

Exemple Un héritage



Dans cet exemple, la classe véhicule représente une classe de base (classe mère) pour tous les véhicules. Les classe voitures et camion sont des classes dérivées (classe fille) de la classe véhicule, qui héritent des attributs et des méthodes spécifiques à leur type de véhicule.

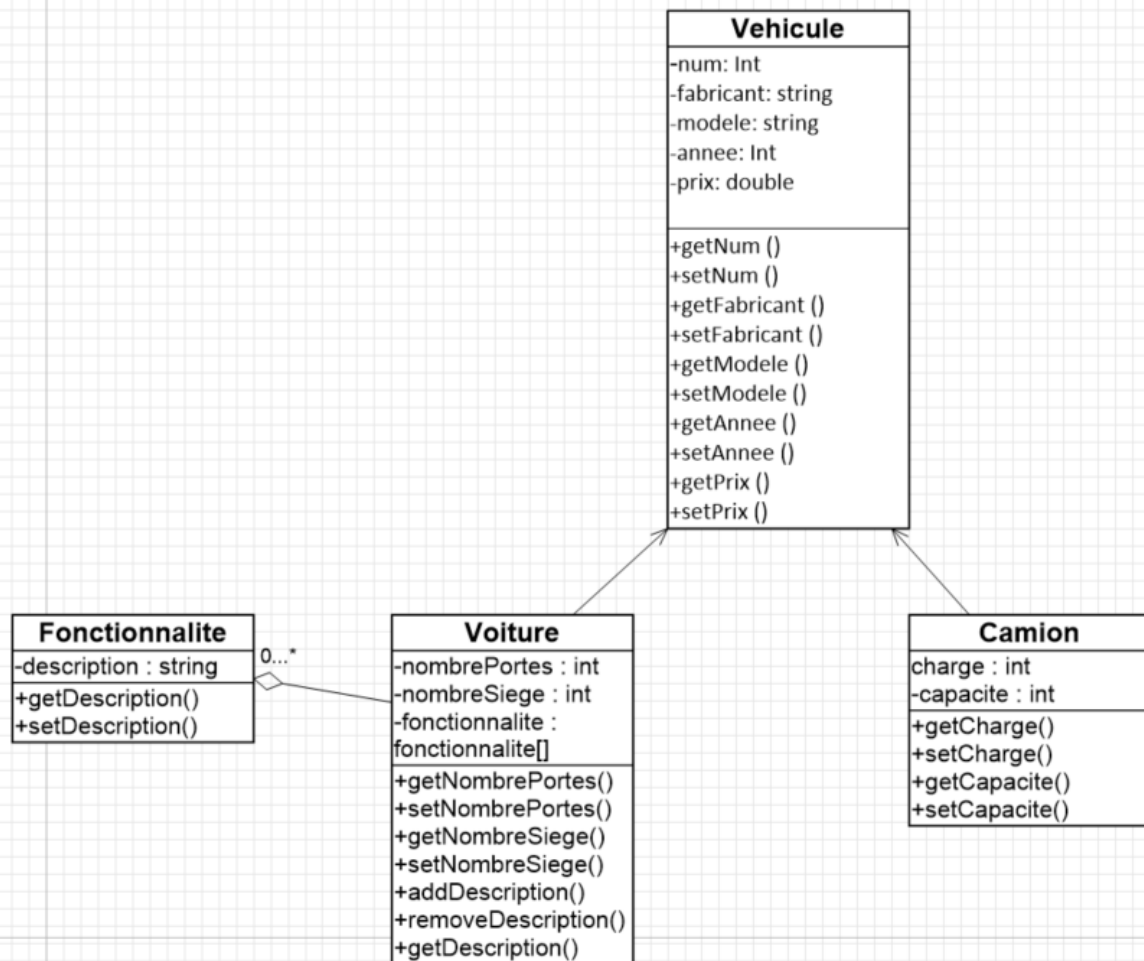
La classe voiture ajoute des attributs tels que le nombre de portes et le nombre de sièges, tandis que la classe camion ajoute des attributs tels que la capacité de charge utile et la capacité de remorquage.

L'héritage permet de réutiliser le code de la classe mère dans les classes filles, ce qui permet de réduire la duplication de code et d'améliorer la maintenabilité du code. Les classes filles peuvent ajouter des fonctionnalités spécifiques à leur type de véhicule, tout en bénéficiant des fonctionnalités de base de la classe mère.

L'agrégation

L'agrégation est une relation entre deux classes qui permet à une classe de contenir des instances d'une autre classe. Les instances de la classe contenue (classe agrégée) peuvent exister indépendamment de la classe conteneur (classe agrégative). L'agrégation est souvent utilisée pour modéliser des relations de composition ou de tout/partie entre les classes. L'agrégation permet de structurer les objets en composants réutilisables, mais elle peut aussi rendre la maintenance plus difficile en cas de changements dans la structure des classes.

Exemple Une agrégation



La classe fonctionnalité représente une caractéristique optionnelle d'un véhicule, avec un attribut description et des méthodes pour accéder et modifier cet attribut. La classe voiture possède des attributs spécifiques tels que le nombre de portes et le nombre de sièges et une liste d'objets fonctionnalité, qui représentent les caractéristiques optionnelles de la voiture.

L'association entre les classes voiture et fonctionnalité est une association d'agrégation, qui modélise le fait qu'une voiture peut avoir plusieurs caractéristiques optionnelles. La classe « *voiture* » agrège une liste d'objets fonctionnalité, qui peuvent être ajoutés ou supprimés à partir de la liste à l'aide des méthodes `addDescription()` et `removeDescription()`.

Dans cet exemple, une fonctionnalité peut être présente dans zéro ou plusieurs voitures.

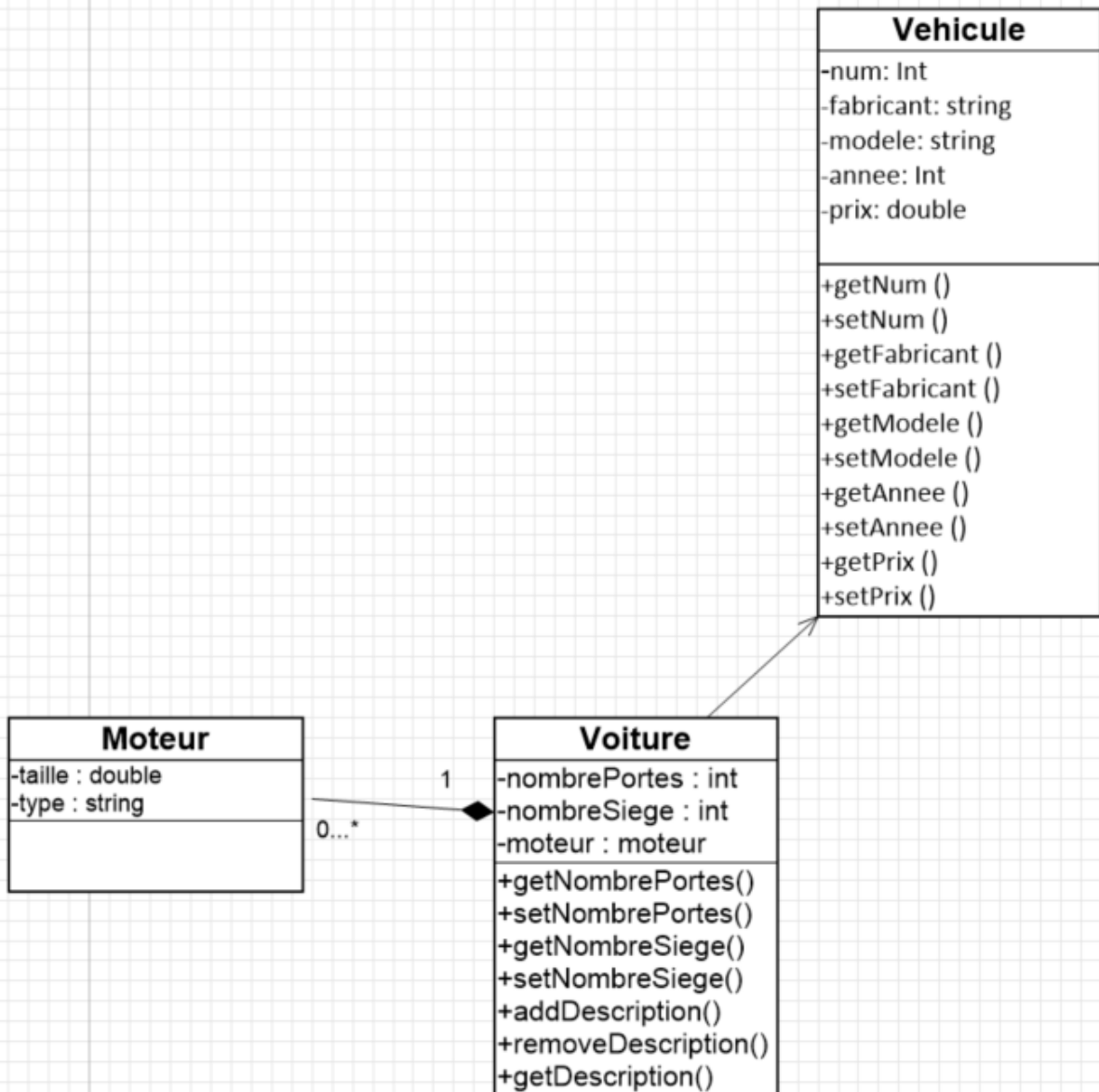
L'association d'agrégation est représentée par une ligne avec un losange blanc qui est dirigé vers la classe dépendante.

La composition

La composition est une relation entre deux classes où l'une des classes est composée d'une ou plusieurs instances de l'autre classe. Cette relation est souvent utilisée pour modéliser des relations « *partie-tout* », où une classe est composée d'autres classes qui ne peuvent pas exister indépendamment de la classe principale. La composition est une relation forte et non symétrique où la classe fille ne peut exister indépendamment de la classe mère.

Une relation de composition est utile pour modéliser des relations de composition fortes entre des classes dans un système et d'assurer que les objets ou les entités composants sont toujours disponibles pour la classe composite. Elle est souvent utilisée pour modéliser des relations fortes entre des classes qui ont une interdépendance ou une dépendance fonctionnelle.

Exemple Une composition



Dans cet exemple, la classe « *voiture* » est composée d'un moteur qui est représenté par la classe *moteur*. Un moteur peut appartenir à zéro ou plusieurs voitures et une voiture peut avoir un seul et unique moteur.

La composition est représentée par une flèche avec un losange noir dirigé vers la classe principale.

Si nous supprimons la classe « *voiture* », le moteur n'a plus de raison d'exister non plus. En revanche, la composition n'est pas symétrique : la classe « *moteur* » peut exister indépendamment de la classe « *voiture* », même si elle est utilisée dans la composition de cette dernière.

Attention Impacts sur la maintenance et l'évolution du système

Les relations entre les classes peuvent avoir des impacts significatifs sur la maintenance et l'évolution du système. Tout changement dans la structure d'une classe peut affecter toutes les classes qui sont associées à elle, ce qui peut rendre la maintenance plus difficile et plus coûteuse. Les contraintes de maintenabilité liées aux relations entre les classes doivent être prises en compte lors de la conception du système.

Définition Définir les relations

Le choix des relations dans un diagramme de classe est une étape importante dans la conception d'un système orienté objet. Les relations permettent de modéliser les interactions entre les classes et les objets du système. Voici quelques conseils pour choisir les relations dans un diagramme de classe.

Premièrement, identifier les relations clés : le premier pas dans le choix des relations est d'identifier les relations clés entre les classes et les objets du système. Il est important de déterminer comment les classes et les objets interagissent les uns avec les autres et de déterminer les types de relations qui existent entre eux.

Ensuite, il nous faut choisir les types de relations : les types de relations courants dans un diagramme de classe sont l'association, l'héritage et la composition. Il est important de choisir les types de relations appropriés pour modéliser les interactions entre les classes et les objets.

Prochaine étape, déterminer les cardinalités : les cardinalités sont des indicateurs qui décrivent le nombre d'instances d'une classe qui peuvent être associées à une ou plusieurs instances d'une autre classe. Il est important de déterminer les cardinalités appropriées pour chaque relation. Par exemple, une relation « *un à un* » signifie qu'une instance d'une classe est associée à une et une seule instance d'une autre classe, tandis qu'une relation « *un à plusieurs* » signifie qu'une instance d'une classe peut être associée à plusieurs instances d'une autre classe.

Ensuite, il nous faut identifier et modéliser les relations complexes : les relations complexes peuvent impliquer plusieurs classes et types de relations différents. Il est important de modéliser ces relations de manière claire et précise. Cela peut impliquer l'utilisation de relations n-aires, d'associations réflexives, ou de sous-classes et de super-classes.

Pour finir, réviser et affiner : une fois que les relations ont été identifiées et modélisées, il est important de réviser et d'affiner le diagramme de classe pour s'assurer qu'il est cohérent et facile à comprendre.

B. Exercice : Quiz

[solution n°2 p.16]

Question 1

Dans un diagramme de classes, que représente une association ternaire ?

- ☐ Une association entre trois classes
- ☐ Une association entre deux classes avec une contrainte supplémentaire
- ☐ Une agrégation entre trois classes

Question 2

Qu'est-ce que la cardinalité d'une association dans un diagramme de classes ?

- ☐ La capacité d'une classe à contenir un nombre donné d'objets d'une autre classe
- ☐ La capacité d'une association à contenir un nombre donné de classes
- ☐ Le nombre d'objets d'une classe qui peuvent être associés à un nombre donné d'objets d'une autre classe

Question 3

Une agrégation représente une relation de tout à partie.

- ☐ Vrai
- ☐ Faux

Question 4

Une composition représente une relation d'association faible entre deux classes.

- ☐ Vrai
- ☐ Faux

Question 5

Dans la programmation orientée objet, qu'est-ce que l'héritage ?

- ☐ Un mécanisme permettant d'étendre les fonctionnalités d'une classe existante en créant une nouvelle classe qui en hérite
- ☐ Un mécanisme permettant d'instancier une classe en créant des objets qui en héritent
- ☐ Un mécanisme permettant de modifier les fonctionnalités d'une classe existante en la modifiant directement

IV. Essentiel

Le diagramme de classes est un outil de modélisation utilisé en génie logiciel pour représenter les classes et les relations entre elles. Il est souvent utilisé pour concevoir des systèmes orientés objet, où les classes représentent des entités du monde réel, telles que des personnes, des lieux, des objets, etc.

Les classes sont représentées sous forme de rectangles, avec des attributs et des méthodes associés à chaque classe. Les relations entre les classes sont représentées sous forme de lignes, qui peuvent être des associations, des agrégations, des compositions ou des héritages.

Une association représente une relation sémantique entre deux classes, tandis qu'une agrégation représente une relation de tout à partie et une composition représente une relation de tout à partie avec une propriété supplémentaire. L'héritage est une relation entre deux classes, où une classe fille hérite des propriétés et des méthodes de la classe mère.

Le diagramme de classes peut être utilisé pour décrire le comportement d'un système, pour concevoir des systèmes complexes et pour communiquer des idées de manière visuelle.

La création d'un diagramme de classes efficace nécessite une compréhension approfondie du système que vous modélisez, ainsi que des compétences en conception orientée objet. La création d'un modèle bien conçu peut aider à éviter les erreurs de conception coûteuses et à assurer le succès de votre projet.

V. Auto-évaluation

A. Exercice

Vous êtes développeur web travaillant pour un zoo et vous devez développer un site web permettant aux visiteurs de consulter les informations sur les animaux, les expositions et les activités disponibles dans le zoo. Le site devra également permettre aux administrateurs du zoo de gérer les informations sur les animaux, les expositions et les activités.

Question 1

[solution n°3 p.17]

Identifiez trois classes importantes dans le diagramme de classes d'un zoo.

Question 2

[solution n°4 p.17]

Créez des associations entre ces classes, puis rajouter la relation d'héritage pour la classe « *Animal* ».

B. Test

Exercice 1 : Quiz

[solution n°5 p.17]

Question 1

Dans un diagramme de classes, qu'est-ce que la cardinalité d'une association et comment est-elle représentée ?

- ☐ La cardinalité décrit le nombre d'objets d'une classe qui peuvent être associés à un nombre donné d'objets d'une autre classe et elle est représentée par des symboles sur les lignes d'association
- ☐ La cardinalité décrit le nombre de classes qui peuvent être associées à une classe donnée et elle est représentée par des symboles sur les rectangles de classe
- ☐ La cardinalité décrit le nombre de méthodes d'une classe qui peuvent être appelées à partir d'une autre classe et elle est représentée par des symboles sur les lignes d'association

Question 2

Qu'est-ce qu'un héritage dans un diagramme de classes ?

- ☐ Une relation de tout à partie
- ☐ Une relation de sous-classe à super-classe
- ☐ Une relation sémantique entre deux classes

Question 3

Comment sont représentées les classes dans un diagramme de classes ?

- ☐ Sous forme de cercles
- ☐ Sous forme de rectangles
- ☐ Sous forme de lignes

Question 4

Qu'est-ce qu'une association dans un diagramme de classes ?

- ☐ Une relation de tout à partie
- ☐ Une relation de sous-classe à super-classe
- ☐ Une relation sémantique entre deux classes

Question 5

Quel est l'avantage de l'utilisation d'un diagramme de classes dans la conception d'un système orienté objet ?

- ☐ Il permet de représenter visuellement les relations entre les classes
- ☐ Il permet de spécifier les attributs et les méthodes de chaque classe
- ☐ Il permet de modéliser le comportement du système


Solutions des exercices

Exercice p. 6 Solution n°1**Question 1**

Les classes dans un diagramme de classes représentent des objets réels et des objets conceptuels.

☒ Vrai

☐ Faux

 Les classes dans un diagramme de classes représentent des objets réels ou conceptuels, tels que des personnes, des voitures ou des transactions financières.


Question 2

Quelle est la bonne syntaxe pour écrire une méthode ?

☐ méthodeClasse()

☒ +methodeClasse()

☐ +méthodeClasse()

 +methodeClasse() est la bonne syntaxe, car il y a la visibilité de la méthode suivie du nom de la méthode sans avoir d'accent et avec les parenthèses à la fin pour montrer qu'il y aura une fonction à écrire.

Question 3


Quel symbole permet d'attribuer une visibilité privée à un attribut ou une méthode ?

☐ Le symbole « + »

☒ Le symbole « - »

☐ Le symbole « # »

☐ Le symbole « ~ »

 Le symbole « - » permet de mettre les attributs ou méthodes en privé, alors que le « + » les met en public, « # » met une visibilité protégée, et « ~ » en met une perceptible dans le même paquet.


Question 4

Qu'est-ce qu'une méthode dans une classe ?

☐ Une propriété ou une caractéristique de l'objet

☒ Une action que l'objet peut effectuer

☐ Une instance de la classe

 Une méthode dans une classe représente une action que l'objet peut effectuer, telle que calculer un prix, afficher une liste d'articles ou envoyer un e-mail.

Question 5

Qu'est-ce qu'un attribut dans une classe ?

- ☒ Une propriété ou une caractéristique de l'objet
- ☐ Une action que l'objet peut effectuer
- ☐ Une instance de la classe
- ☒ Un attribut dans une classe représente une propriété ou une caractéristique de l'objet, telle que son nom, son âge ou son adresse.

Exercice p. 12 Solution n°2

Question 1

Dans un diagramme de classes, que représente une association ternaire ?

- ☒ Une association entre trois classes
- ☐ Une association entre deux classes avec une contrainte supplémentaire
- ☐ Une agrégation entre trois classes
- ☒ Une association entre trois classes. Une association ternaire représente une relation sémantique entre les objets de trois classes. Elle peut être utilisée pour modéliser des relations complexes entre des objets.

Question 2

Qu'est-ce que la cardinalité d'une association dans un diagramme de classes ?

- ☐ La capacité d'une classe à contenir un nombre donné d'objets d'une autre classe
- ☐ La capacité d'une association à contenir un nombre donné de classes
- ☒ Le nombre d'objets d'une classe qui peuvent être associés à un nombre donné d'objets d'une autre classe
- ☒ Le nombre d'objets d'une classe qui peuvent être associés à un nombre donné d'objets d'une autre classe. La cardinalité est utilisée pour décrire les contraintes sur le nombre d'objets d'une classe qui peuvent être associés à un nombre donné d'objets d'une autre classe.

Question 3

Une agrégation représente une relation de tout à partie.

- ☒ Vrai
- ☐ Faux
- ☒ L'agrégation est une relation de tout à partie, où un objet de la classe tout contient des objets de la classe partie. Par exemple, un ordinateur peut contenir plusieurs composants tels que le clavier, la souris, l'écran, etc.

Question 4

Une composition représente une relation d'association faible entre deux classes.

- ☐ Vrai
- ☒ Faux
- ☒ La composition représente une relation de tout à partie comme l'agrégation, mais avec une propriété supplémentaire selon laquelle les objets de la classe partie ne peuvent exister sans l'objet de la classe tout. Par exemple, un ordinateur portable ne peut exister sans ses composants tels que le clavier, la souris, l'écran, etc.

Question 5

Dans la programmation orientée objet, qu'est-ce que l'héritage ?

- ☒ Un mécanisme permettant d'étendre les fonctionnalités d'une classe existante en créant une nouvelle classe qui en hérite
- ☐ Un mécanisme permettant d'instancier une classe en créant des objets qui en héritent
- ☐ Un mécanisme permettant de modifier les fonctionnalités d'une classe existante en la modifiant directement
- ☒ Un mécanisme permettant d'étendre les fonctionnalités d'une classe existante en créant une nouvelle classe qui en hérite. L'héritage permet à une classe abstraite d'hériter des attributs et des méthodes de la classe de base. La classe dérivée peut ensuite ajouter des fonctionnalités supplémentaires ou modifier les fonctionnalités héritées pour répondre à ses besoins spécifiques.

p. 13 Solution n°3

Les trois classes importantes dans le diagramme de classes d'un zoo peuvent être « *Animal* », « *Enclos* » et « *Personnel* ». Ces trois classes sont centrales pour la modélisation d'un zoo. La classe « *Animal* » représente les différents types d'animaux qui peuvent être hébergés dans le zoo. La classe « *Enclos* » représente les différents types d'enclos dans lesquels les animaux sont logés. La classe « *Personnel* » représente les différentes personnes impliquées dans la gestion du zoo, comme les gardiens, les vétérinaires, etc.

p. 13 Solution n°4

Ajoutez une association entre la classe « *Animal* » et la classe « *Enclos* » avec une cardinalité de « 1 à plusieurs » pour préciser que chaque enclos peut héberger plusieurs animaux et que chaque animal peut être hébergé dans un seul enclos. Ajoutez une association entre la classe « *Animal* » et la classe « *Personnel* » avec une cardinalité de « plusieurs à plusieurs » pour préciser que plusieurs membres du personnel peuvent être responsables de plusieurs animaux et que chaque animal peut avoir plusieurs membres du personnel responsables de lui. Ajoutez une association entre la classe « *Enclos* » et la classe « *Personnel* » avec une cardinalité de « 1 à plusieurs » pour préciser qu'un seul membre du personnel peut être responsable de la gestion de plusieurs enclos, mais chaque enclos ne peut être géré que par un seul membre du personnel.

Ajoutez une relation d'héritage à la classe « *Animal* » avec par exemple la classe « *Lion* », « *Girafe* » et « *Singe* », ils hériteront des attributs et des méthodes de la classe « *Animal* » et vous pouvez rajouter des attributs et des méthodes pour chaque sous-classe comme longueur du cou pour une girafe, taille de la crinière pour un lion, etc.

Exercice p. 14 Solution n°5**Question 1**

Dans un diagramme de classes, qu'est-ce que la cardinalité d'une association et comment est-elle représentée ?

- ☒ La cardinalité décrit le nombre d'objets d'une classe qui peuvent être associés à un nombre donné d'objets d'une autre classe et elle est représentée par des symboles sur les lignes d'association
- ☐ La cardinalité décrit le nombre de classes qui peuvent être associées à une classe donnée et elle est représentée par des symboles sur les rectangles de classe
- ☐ La cardinalité décrit le nombre de méthodes d'une classe qui peuvent être appelées à partir d'une autre classe et elle est représentée par des symboles sur les lignes d'association

- Q La cardinalité permet de spécifier les contraintes sur le nombre d'objets d'une classe qui peuvent être associés à un nombre donné d'objets d'une autre classe. Elle est représentée par des symboles tels que 1, 0..1, 1.., etc., qui sont placés sur les lignes d'association.

Question 2

Qu'est-ce qu'un héritage dans un diagramme de classes ?

- ☐ Une relation de tout à partie
- ☒ Une relation de sous-classe à super-classe
- ☐ Une relation sémantique entre deux classes

- Q Un héritage est une relation de sous-classe à super-classe dans laquelle une classe fille hérite des propriétés et des méthodes de la classe mère. Par exemple, une classe de base Animal peut être dérivée en classes dérivées telles que Chien, Chat, Oiseau, etc.

Question 3

Comment sont représentées les classes dans un diagramme de classes ?

- ☐ Sous forme de cercles
- ☒ Sous forme de rectangles
- ☐ Sous forme de lignes

- Q Les classes sont représentées sous forme de rectangles dans un diagramme de classes, avec des attributs et des méthodes associés à chaque classe.

Question 4

Qu'est-ce qu'une association dans un diagramme de classes ?

- ☐ Une relation de tout à partie
- ☐ Une relation de sous-classe à super-classe
- ☒ Une relation sémantique entre deux classes

- Q Une association est une relation sémantique entre deux classes dans laquelle un objet de la classe A est associé à un ou plusieurs objets de la classe B. Par exemple, une classe A Personne peut être associée à une classe B Adresse.

Question 5

Quel est l'avantage de l'utilisation d'un diagramme de classes dans la conception d'un système orienté objet ?

- ☒ Il permet de représenter visuellement les relations entre les classes
- ☐ Il permet de spécifier les attributs et les méthodes de chaque classe
- ☐ Il permet de modéliser le comportement du système

- Q L'utilisation d'un diagramme de classes dans la conception d'un système orienté objet permet de représenter visuellement les relations entre les classes. Cela peut aider à mieux comprendre les interactions entre les classes et faciliter la communication entre les membres de l'équipe de développement.