Université de Msila – Département d'Informatique

Génie logiciel (GL)

(3ème année SI&ISIL)

Dr BOUNIF M-E

- 6. Le diagramme Classe
- **Découvrir les objets du système par décomposition**

La décomposition des messages fait apparaître les objets du système, car elle conduit à des messages plus fins dont il convient de rechercher le destinataire.

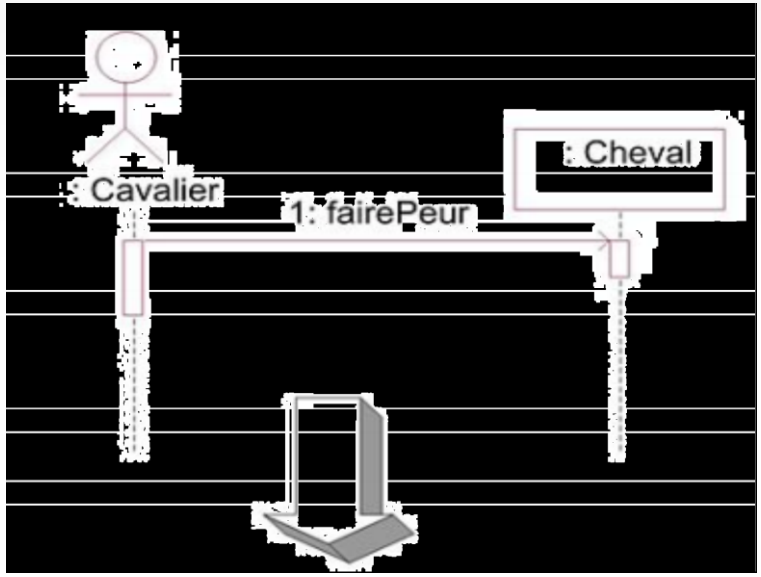
Une autre approche possible est la décomposition de l'information contenue dans un objet. Souvent, cette information est trop complexe pour n'être représentée que par la structure d'un seul objet. Elle doit parfois être également répartie entre plusieurs objets.

Exemple

La décomposition d'un cheval pour faire apparaître ses différents organes peut se faire soit par la décomposition d'un diagramme de séquence, soit par la décomposition guidée par les données.

La décomposition par le diagramme de séquence consiste à analyser différents envois de message: faire peur, courir, manger, dormir.

Ces derniers feront apparaître progressivement les différents organes du cheval. Elle est illustrée à la figure 6.1 pour le message fairePeur. Un cheval dilate ses naseaux pour marquer l'alerte, la surprise ou la peur. Il pince la bouche pour indiquer tension, peur ou colère. Enfin, la ruade est un mouvement défensif.



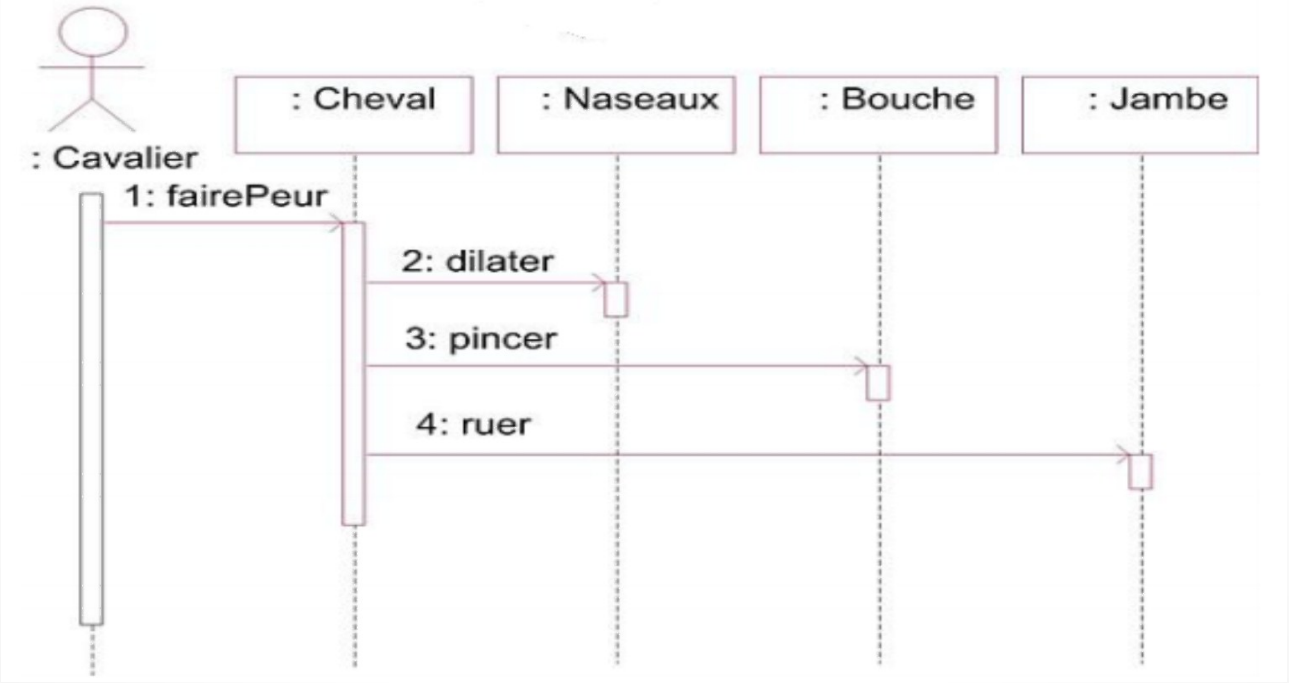


Figure 6.1 Découverte des objets par enrichissement du diagramme de séquence

La décomposition guidée par les données consiste à étudier directement les différents organes d'un cheval et à les prendre en compte dans le diagramme des classes. La figure 6.2 illustre un cheval composé de ses différents organes.

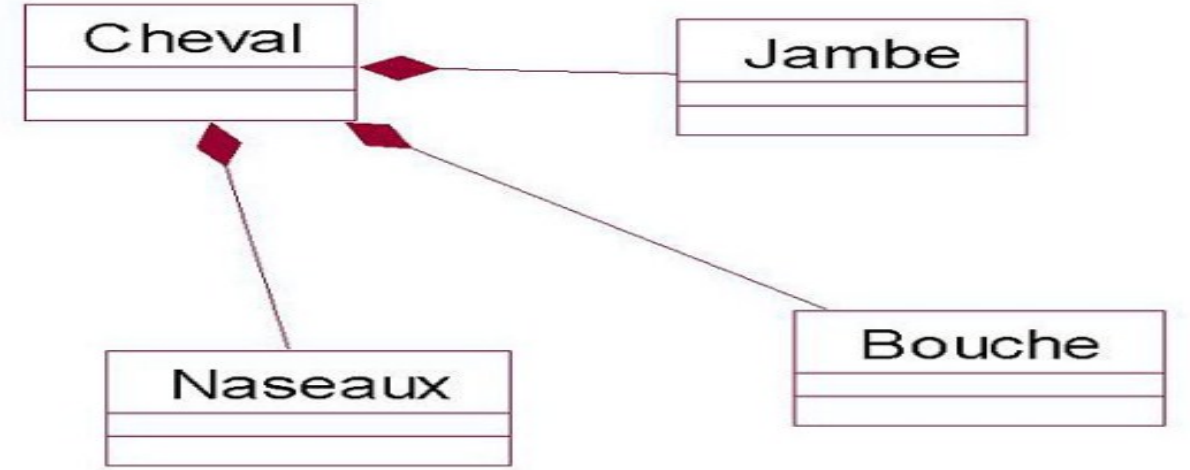


Figure 6.2 Composition de Cheval

La décomposition par le diagramme de séquence, comme la décomposition guidée par les données apparaissent naturelles pour découvrir les objets. Ce résultat est normal car un objet est l'assemblage d'une structure et d'un comportement. Il convient enfin de remarquer que ces deux approches ne sont pas incompatibles. La décomposition guidée par les données est plus efficace quand la personne chargée de la modélisation connaît bien le domaine.

- **La représentation des classes**

1. La forme simplifiée de représentation des classes

Les objets du système sont décrits par des classes dont une forme simplifiée de la représentation en UML est donnée à la figure 6.3. Cette représentation est constituée de trois parties.

Figure 6.3 Représentation simplifiée d'une classe en UML

La première partie contient le nom de la classe.
La deuxième partie contient les attributs. Ceux-ci contiennent l'information portée par un objet. L'ensemble des attributs forme la structure de l'objet.
La troisième partie contient les méthodes. Celles-ci correspondent aux services offerts par l'objet. Elles peuvent modifier la valeur des attributs. L'ensemble des méthodes forme le comportement de l'objet.
Exemple

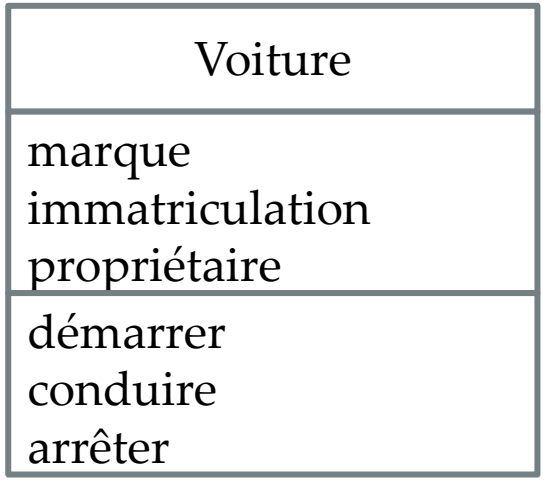


Figure 6.4 La classe Voiture

Cette forme de représentation des classes est la plus simple car elle ne fait pas apparaître les caractéristiques des attributs et des méthodes en dehors de leur nom. Elle est très souvent utilisée lors d'une première phase de modélisation.
Avant d'examiner les représentations plus complètes, nous devons aborder les notions essentielles d'encapsulation, de type et de signature des méthodes.

2. L'encapsulation

L'encapsulation a été introduite dans le chapitre Les concepts de l'approche par objets. Certains attributs et méthodes ne sont pas exposés à l'extérieur de l'objet. Ils sont encapsulés et appelés attributs et méthodes privés de l'objet. UML, comme la plupart des langages à objets modernes, introduit trois possibilités d'encapsulation :

L'attribut ou la méthode privée : la propriété n'est pas exposée en dehors de la classe, y compris au sein de ses sous-classes.

L'attribut ou la méthode protégée : la propriété n'est exposée qu'aux instances de la classe et de ses sous-classes.

L'encapsulation de paquetage : la propriété n'est exposée qu'aux instances des classes de même paquetage.

La notion de propriété privée est rarement utilisée car elle amène à instaurer une différence entre les instances d'une classe et les instances de ses sous-classes. Quant à l'encapsulation de paquetage, elle provient du langage Java. Elle est réservée à l'écriture de diagrammes destinés aux développeurs.
L'encapsulation est représentée par un signe plus, un signe moins, un dièse, ou un tilde précédant le nom de l'attribut. Le tableau suivant détaille la signification de ces signes.

public	+	élément non encapsulé visible par tous
protégé	#	élément encapsulé visible dans les sous-classes de la classe
privé	-	élément encapsulé visible seulement dans la classe
paquetage	~	élément encapsulé visible seulement dans les classes du même paquetage

Exemple

La figure 6.5 montre la classe Voiture en faisant ressortir les caractéristiques d'encapsulation.

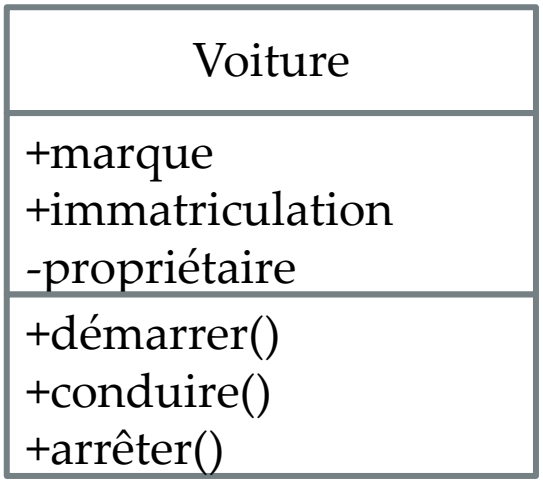


Figure 6.5 La classe Voiture avec les caractéristiques d'encapsulation

3. La notion de type

Nous appelons ici variable tout attribut, paramètre et valeur de retour d'une méthode. D'une façon générale, nous appelons variable tout élément pouvant prendre une valeur.

Le type est une contrainte appliquée à une variable. Il consiste à fixer l'ensemble des valeurs possibles que peut prendre cette variable. Cet ensemble peut être une classe, auquel cas la variable doit contenir une référence vers une instance de cette classe. Il peut être standard comme l'ensemble des entiers, des chaînes de caractères, des booléens ou des réels. Dans ces derniers cas, la valeur de la variable doit être respectivement un entier, une chaîne de caractères, une valeur booléenne et un réel.

Les types standard sont désignés ainsi :

- Integer pour le type des entiers ;
- String pour le type des chaînes de caractères ;
- Boolean pour le type des booléens ;
- Real pour le type des réels.

Le type d'un attribut, d'un paramètre et de la valeur de retour d'une méthode est précisé au niveau de la représentation de la classe.

Exemple

La figure 6.6 montre la classe Voiture pour laquelle le type de tous les attributs a été fixé.

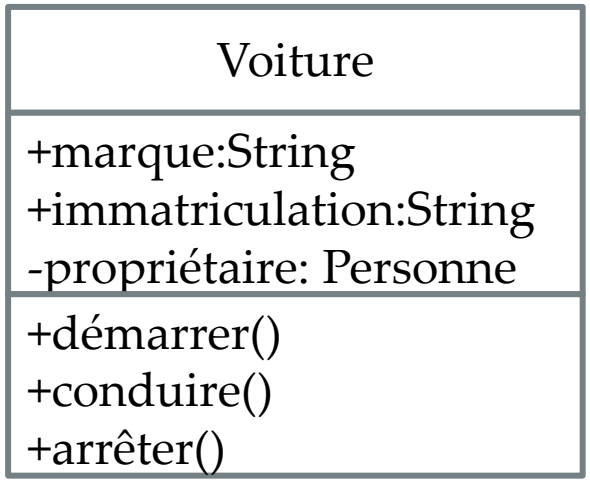


Figure 6.6 La classe Voiture avec attributs typés

4. La signature des méthodes

Une méthode d'une classe peut prendre des paramètres et renvoyer un résultat. Les paramètres sont des valeurs transmises :

- à l'aller, lors de l'envoi d'un message appelant la méthode ;
- ou au retour d'appel de la méthode.

Le résultat est une valeur transmise à l'objet appelant lors du retour d'appel.

Comme nous l'avons vu précédemment, ces paramètres ainsi que le résultat peuvent être typés. L'ensemble constitué du nom de la méthode, des paramètres avec leur nom et leur type ainsi que le type du résultat s'appelle la signature de la méthode. Une signature prend la forme suivante :

<nom Méthode> (<nom Paramètre> :<type>,…):<typeRésultat>

Rappelons que le nombre de paramètres peut être nul et que le type du résultat est optionnel.

Exemple

La figure 6.7 montre la classe Voiture dont la méthode *conduire* a été munie de deux paramètres ces deux valeurs sont de types Lieu.

Quant aux autres méthodes, elles ne prennent pas de paramètres et ne renvoient pas de résultat.

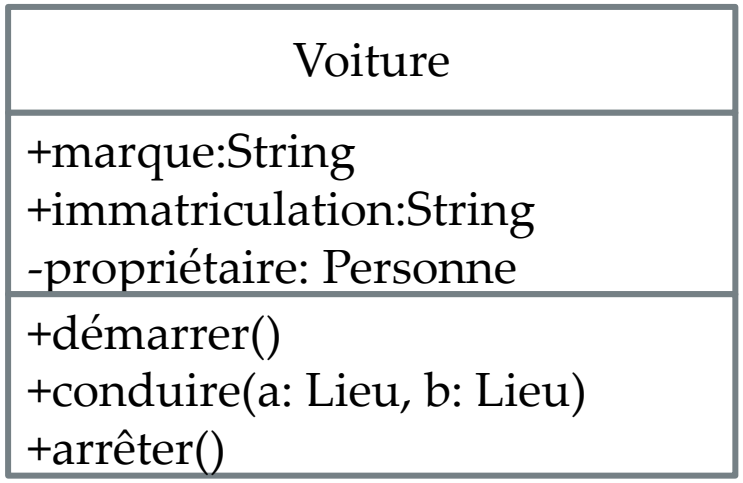


Figure 6.7 La classe Voiture avec la signature des méthodes

5. La forme complète de représentation des classes

La représentation complète des classes fait apparaître les attributs avec leur caractéristique d'encapsulation, leur type et les méthodes avec leur signature complète.

Il est également possible de donner des valeurs par défaut aux attributs et aux paramètres d'une méthode. La valeur par défaut d'un attribut est celle qui lui est attribuée dès la création d'un nouvel objet. La valeur par défaut d'un paramètre est utilisée lorsque l'appelant d'une méthode ne fournit pas explicitement la valeur de ce paramètre au moment de l'appel.

La figure 6.8 illustre la représentation complète d'une classe. Il est bien sûr possible de choisir une représentation intermédiaire entre la représentation simplifiée et la représentation complète.

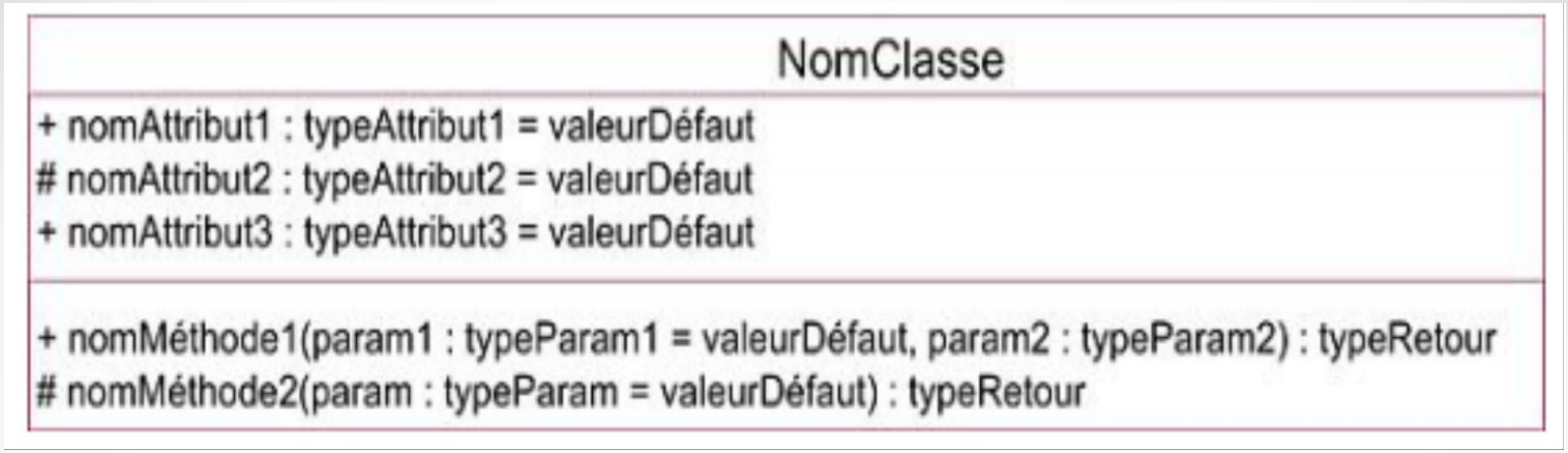


Figure 6.8 Représentation complète d'une classe en UML

Cette représentation complète intéresse surtout le développeur chargé de la réalisation du logiciel en aval de la modélisation. Il dispose ainsi d'une description de la classe très proche de celle qu'il doit utiliser dans son langage de programmation.

6. Les attributs et les méthodes de classe

Chaque instance d'une classe contient une valeur spécifique pour chacun de ses attributs. Cette valeur n'est donc pas partagée par l'ensemble des instances. Dans certains cas, il est nécessaire d'utiliser des attributs dont la valeur est commune à l'ensemble des objets d'une classe. Un tel attribut voit sa valeur partagée au même titre que son nom, son type et sa valeur par défaut. Un tel attribut est appelé attribut de classe car il est lié à la classe.

Un attribut de classe est représenté par un nom souligné. Il peut être encapsulé et posséder un type. Il est vivement recommandé de lui donner une valeur par défaut.

Exemple

La figure 6.9 introduit une classe qui décrit une pièce de rechange d'une voiture. Lorsque ce produit est vendu, il est soumis à une TVA dont le montant est le même pour tous les pièces.

Cet attribut est souligné et protégé car il sert à calculer le prix avec TVA incluse. Il est exprimé en pourcentage d'où son type Integer. La valeur par défaut est 30 le taux de TVA des produits non-alimentaires.

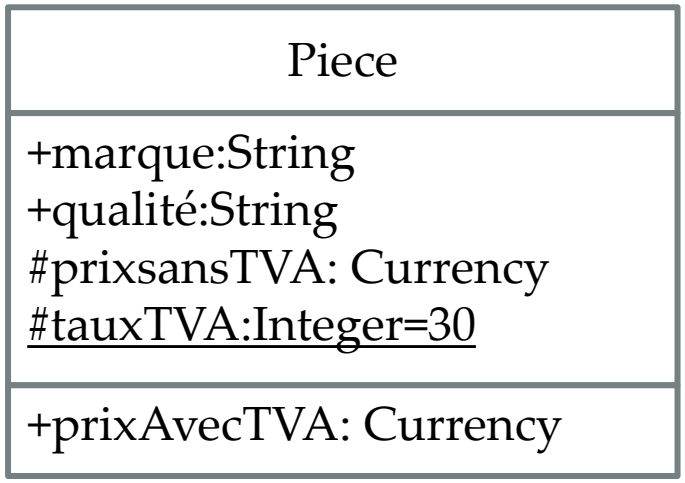


Figure 6.9 Attribut de classe

Le type utilisé pour l'attribut prixSansTVA et le résultat de la Méthode prixAvecTVA est Currency. Il indique que les valeurs sont des montants monétaires.

Au sein d'une classe, il peut également exister une ou plusieurs méthodes de classe. Celles-ci sont liées à la classe. Pour appeler une méthode de classe, il faut envoyer un message à la classe elle-même et non à l'une de ses instances. Une telle méthode ne manipule que les attributs de classe.

Exemple

La figure 6.10 ajoute une méthode de classe à la classe Piece qui sert à fixer le taux de TVA.

En effet, celui-ci est fixé par la loi et cette dernière peut être modifiée.

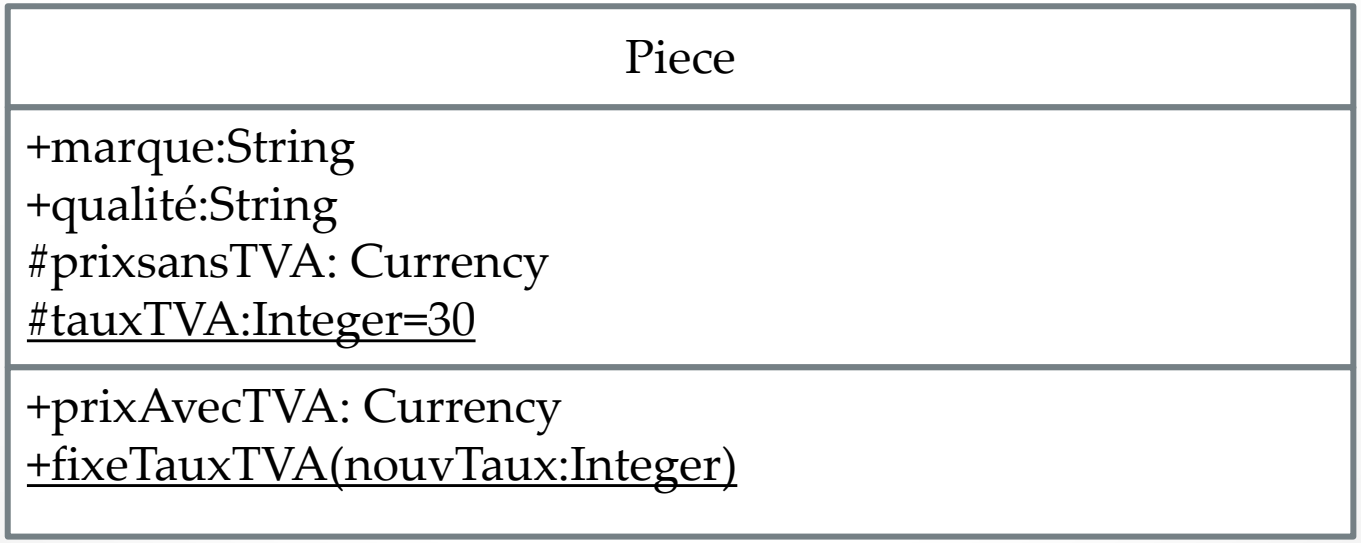


Figure 6.10 Méthode de classe

Un attribut ou une méthode de classe ne sont pas hérités. En effet, l'héritage s'applique à la description des instances qui est calculée par l'union de la structure et du comportement de la classe et de ses surclasses. Une sous-classe peut accéder à un attribut ou à une méthode de classe de l'une de ses surclasses mais n'en hérite pas. Rappelons aussi qu'une sous-classe peut accéder à un attribut ou à une méthode de classe de l'une de ses surclasses, à la condition que l'encapsulation privée n'ait pas été utilisée.

7. Les attributs calculés

UML introduit la notion d'attribut calculé dont la valeur est donnée par une fonction basée sur la valeur d'autres attributs. Un tel attribut possède un nom précédé du signe / et il est suivi d'une contrainte donnant le moyen de calculer sa valeur.

Exemple

Nous reprenons l'exemple de la figure 6.10. La méthode prixAvecTVA est remplacée par l'attribut calculé /prixAvecTVA comme l'illustre la figure 6.11.

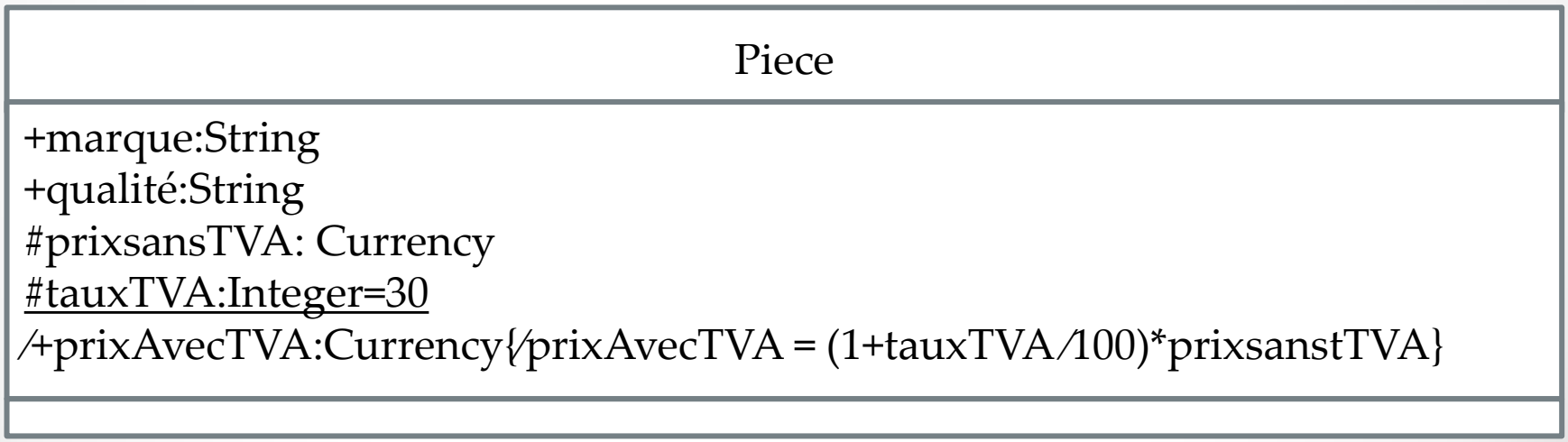


Figure 6.11 Attribut calculé

- Toute contrainte exprimée dans un diagramme UML est écrite entre accolades.