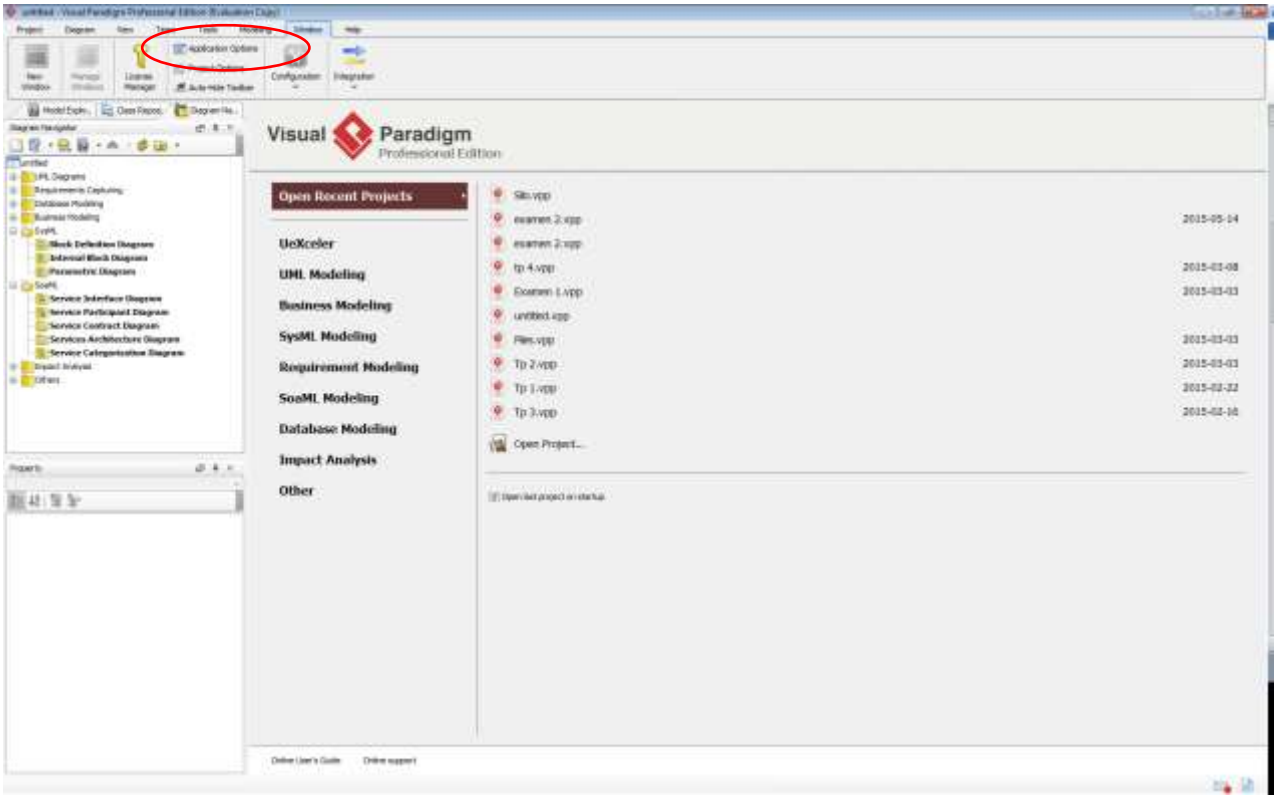
Table des matières

Diagramme de classe et d'objet UML

Environnement Visual Paradigm

Le logiciel VisualParadigm (**VP**) est un logiciel puissant pour créer des diagrammes UML, des diagrammes d'entité ERD et d'autres schémas d'analyse. Il est l'un des plus utilisés et il offre une version communautaire gratuite. C'est lui que nous allons utiliser dans le cours. Il est construit sur la plateforme Eclipse et il reprend donc son principe de fonctionnement avec un « workspace » et plusieurs projets.

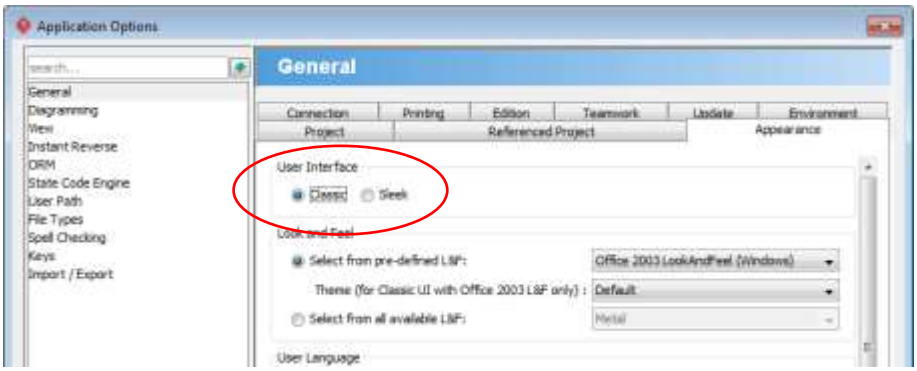
En démarrant VP, il est bien possible que vous soyez devant l'interface le thème « Sleek » de la nouvelle version de VP :



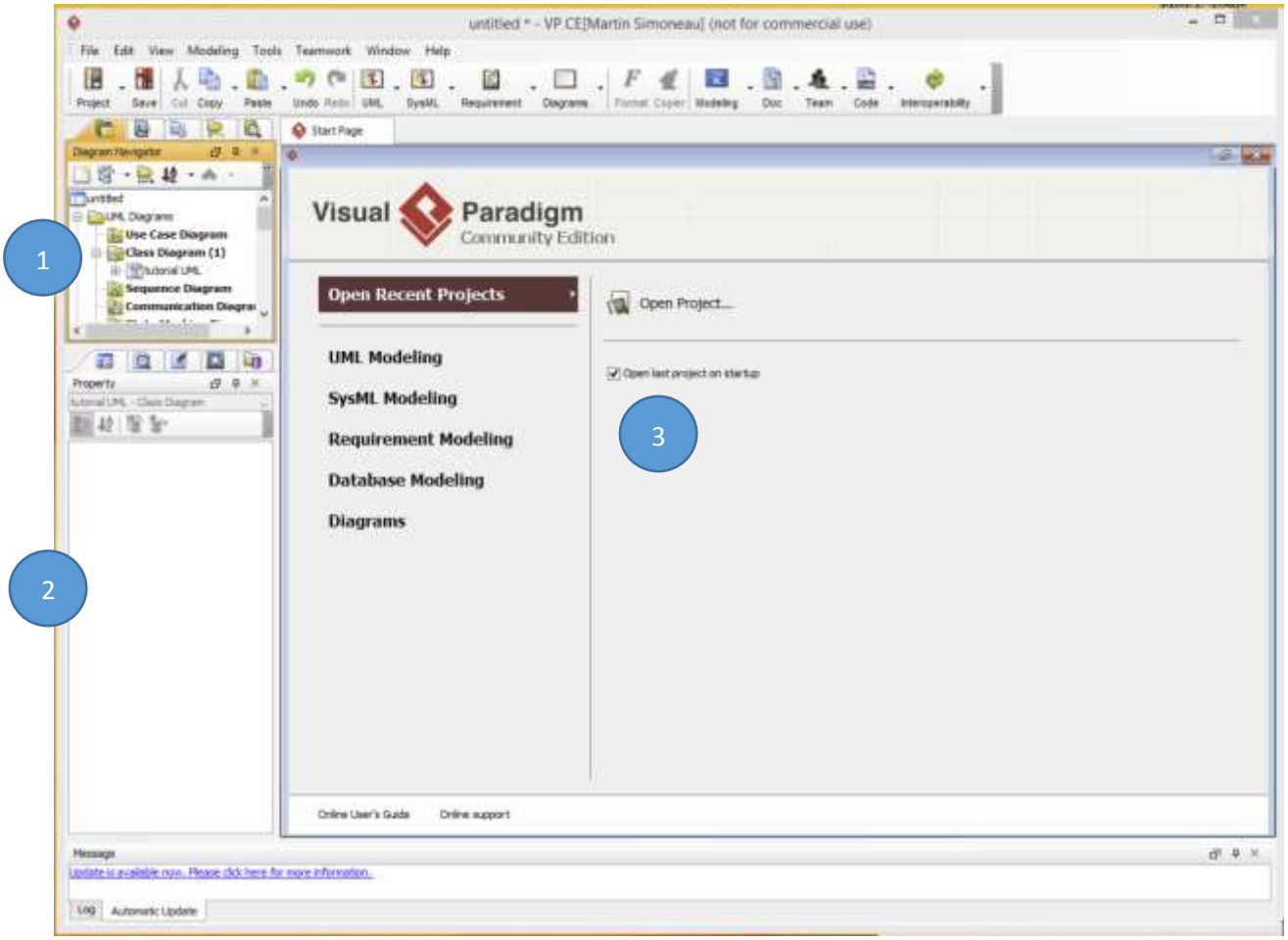
Comme l'ensemble des explications du présent document sont basées sur l'ancienne interface « Classic », nous allons modifier la configuration de VP pour l'adapter à cette réalité. À la suite de la lecture de ce document, vous pourrez revenir au nouveau style d'interface « Sleek ». Son apprentissage sera plus facile.

Quand vous êtes en mode d'interface « Sleek » et que vous voulez passer vers le mode d'interface « Classic ». Choisissez dans l'onglet « Windows » l'option « Application Options ».

Dans l'onglet « Appearance », de la boîte de dialogue, sélectionnez l'option « Classic » dans la section « User Interface ». Confirmez votre choix à l'aide du bouton « Apply » suivi de « OK ».



Par la suite il faudra redémarrer VP, pour constater le changement.

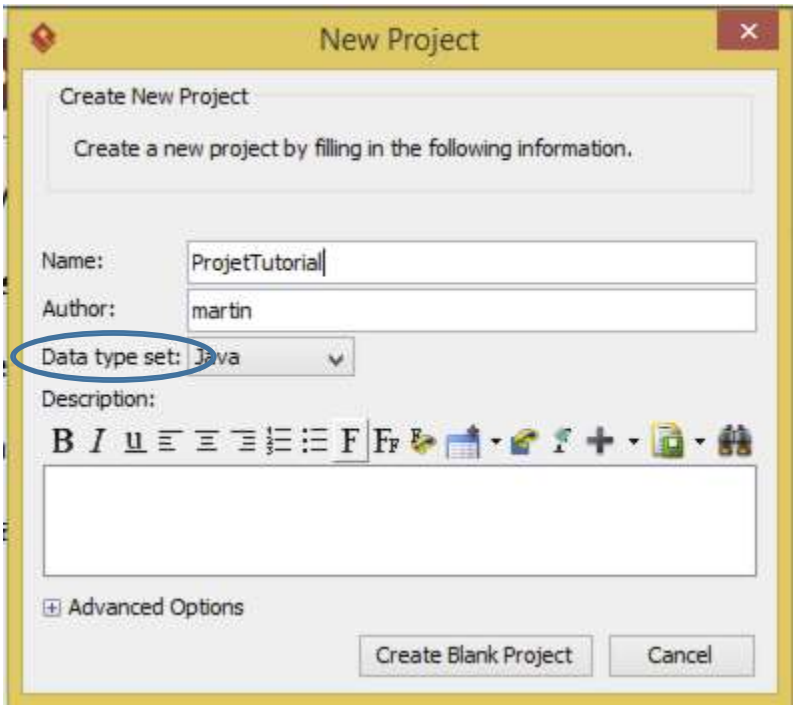


- Zone 1 : Dans cette zone vous trouverez
  - **L'explorateur de modèle** : vous permet de consulter tous les éléments du modèle en cours
  - **Le navigateur de diagramme** : Cette vue retrouve tous les diagrammes dans votre projet et elle vous les présente par type de diagramme
  - **Le répertoire de classe** : montre les classes qui sont associées au modèle lorsqu'on fait de la « rétroingénérie » et lorsqu'on fait créer le code du modèle.
  - Les autres vues ne seront pas utilisées dans ce cours.
- Zone 2 : Zone de propriétés. Elle contient :
  - **Propriété** : montre toutes les propriétés de l'élément sélectionné dans la zone 3
  - Survol du diagramme (**diagram overview**) : Montre une vue globale du diagramme et permet de se déplacer rapidement sur ce dernier.
  - Les autres vues ne seront pas utilisées dans ce cours.
- Zone 3 : Zone de travail. C'est dans cette zone que vous travaillerez sur vos diagrammes UML. Comme Eclipse cette zone peut contenir plusieurs diagrammes ouverts simultanément.

Avant de commencer

Il faut d'abord créer un projet.

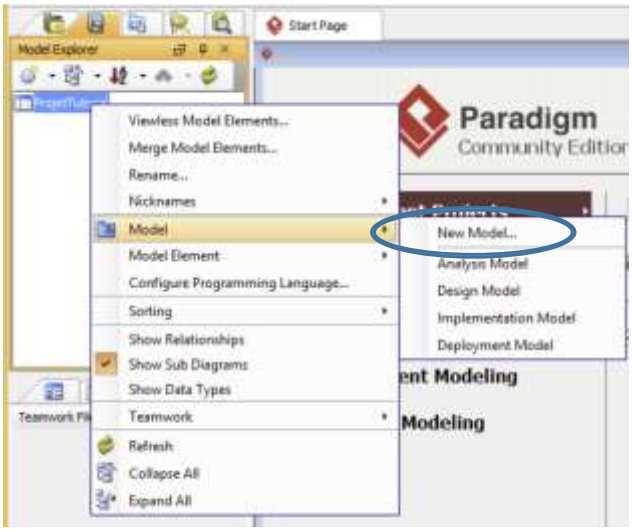
- Appuyer sur le bouton *Project* puis sur le sous-menu *New*
- Entrez le nom de votre projet puis ***n'oubliez pas de sélectionner JAVA*** **comme Data type set**. Cela va assurer que les types utilisés sont ceux du Java.



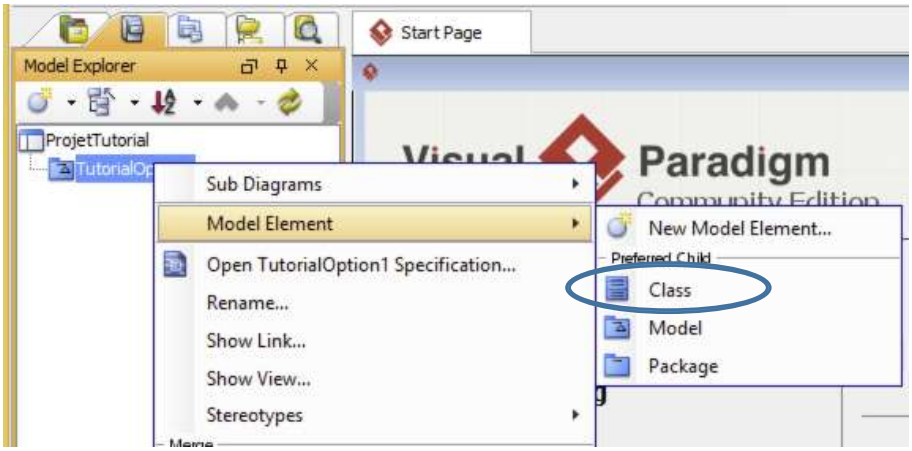
Un workspace VP peut contenir plusieurs projets, mais, contrairement à Eclipse, un seul peut être ouvert à la fois. Pour travailler avec simultanément avec plusieurs idées (alternatives), il faut utiliser les modèles. Un modèle est un ensemble d'éléments qui fonctionnent ensemble. Il est possible d'avoir plusieurs modèles dans un projet VP et il est possible de dupliquer des modèles pour concevoir plusieurs alternatives simultanément.

On vous recommande donc de toujours créer un modèle avant de fabriquer vos diagrammes. Pour créer un modèle,

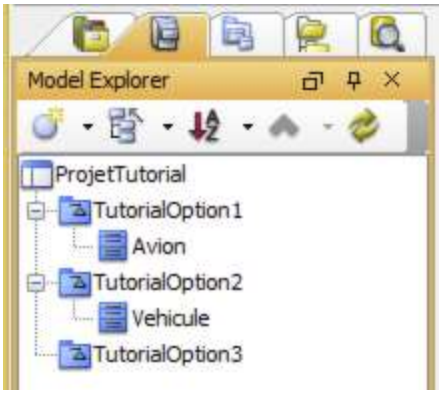
- allez dans l'*explorateur de modèle* et à l'aide du menu contextuel sur votre projet sélectionner le menu **Model/New Model**.



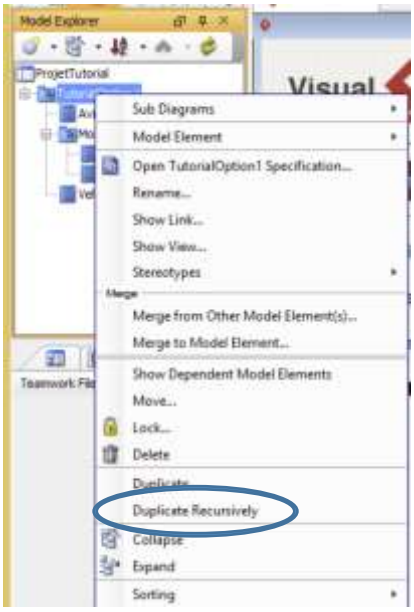
- Entrez ensuite le nom que vous voulez donner à ce modèle.
- Il est possible d'imbriquer des modèles les uns dans les autres (comme les dossiers d'un système d'exploitation). On peut alors parler de sous modèles.
- Vous pourrez ensuite créer des classes à l'aide du menu contextuel sur le nouveau modèle. Model Element/ Class



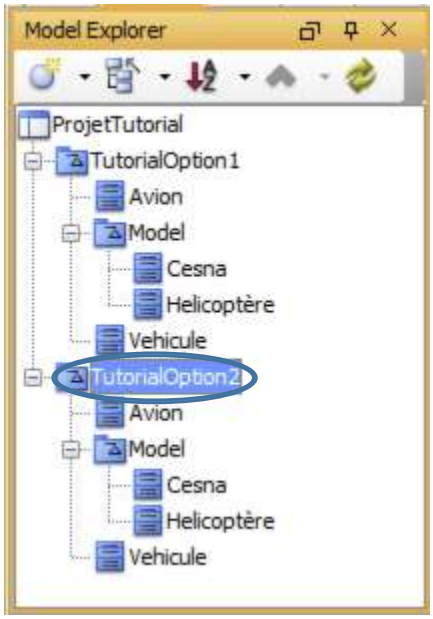
Vous pouvez créer plusieurs modèles dans votre projet en répétant les étapes précédentes



Très souvent on désire dupliquer un modèle existant pour esquisser une nouvelle hypothèse (pour proposer à des collègues par exemple). VP offre 2 options pour faire cela simplement : **Duplicate** (seulement le modèle sélectionné) et **Duplicate recursively** (tous les éléments et sous éléments du modèle sélectionné incluant les sous-modèles).



En faisant *Duplicate recursively* on crée une copie complètement indépendante, mais conforme de notre modèle. On pourra la modifier à notre guise sans abimer le modèle original. On peut donc facilement créer plusieurs versions de modèles afin d'étudier et comparer plusieurs pistes de design. Ici on a fait une copie récursive du modèle afin de créer une seconde option de design.

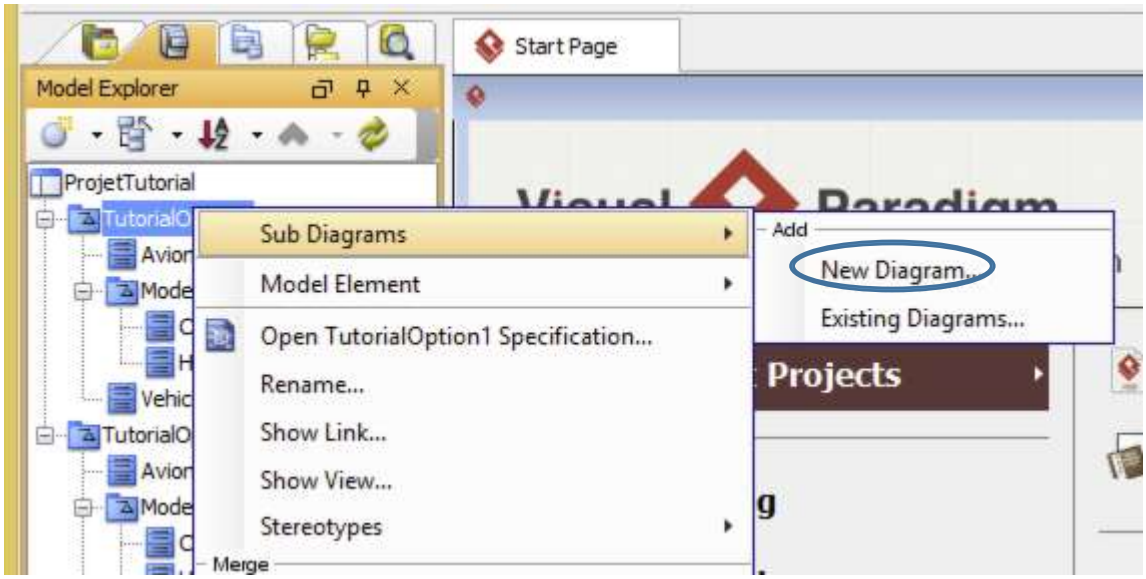


Une fois le projet bien préparé avec un modèle bien défini, on peut s'attaquer aux différents diagrammes.

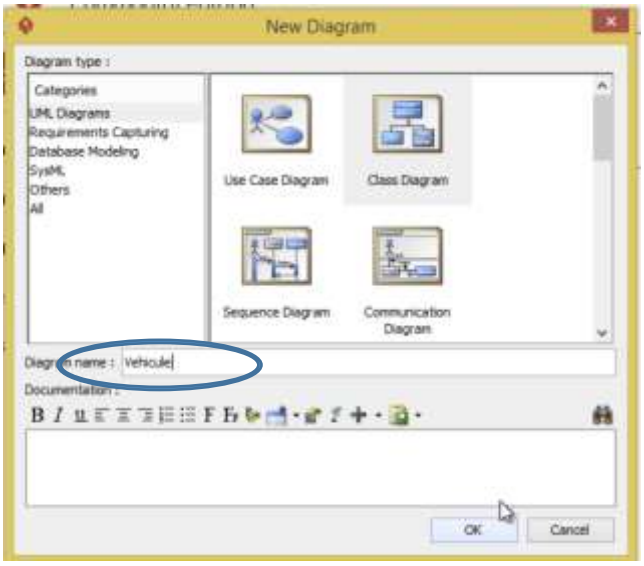
La classe

Le diagramme de classe est un diagramme conçu pour montrer rapidement l'organisation des différentes classes qui constituent un programme. Pour créer un diagramme de classe

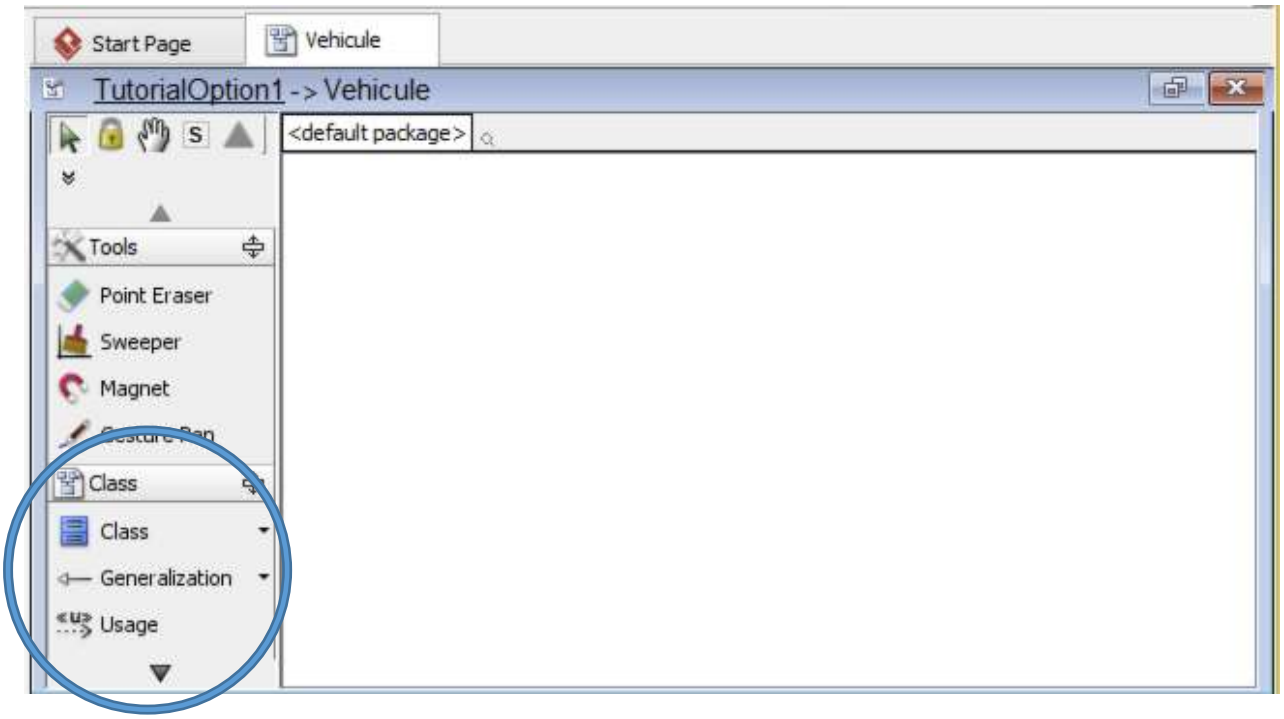
- Ouvrez le menu contextuel sur votre modèle (Sub Diagrams / New Diagram)



- Choisissez le Diagramme de classe et donner lui un nom (ici Vehicule)



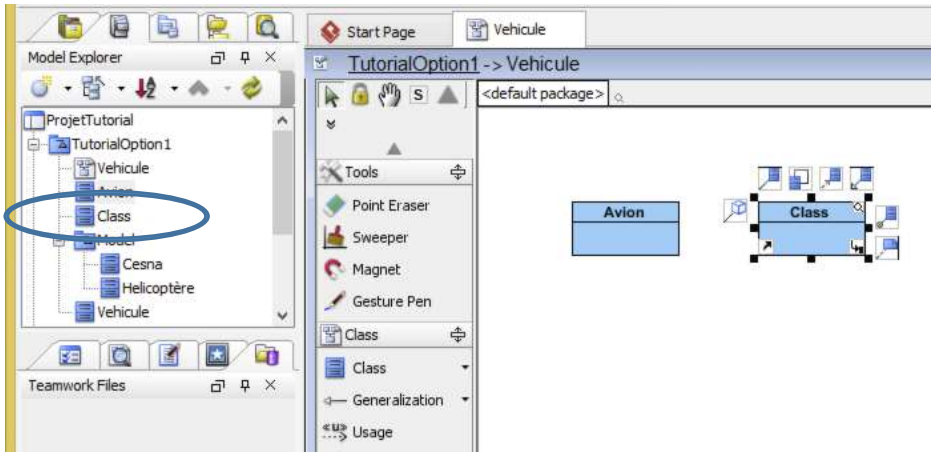
Le diagramme sera ajouté dans votre modèle ainsi que dans le navigateur de diagramme. Il sera également ouvert automatiquement. Vous pouvez alors commencer à créer votre modèle. La zone de travail ressemble alors à :



Ajouter une classe sur un diagramme.

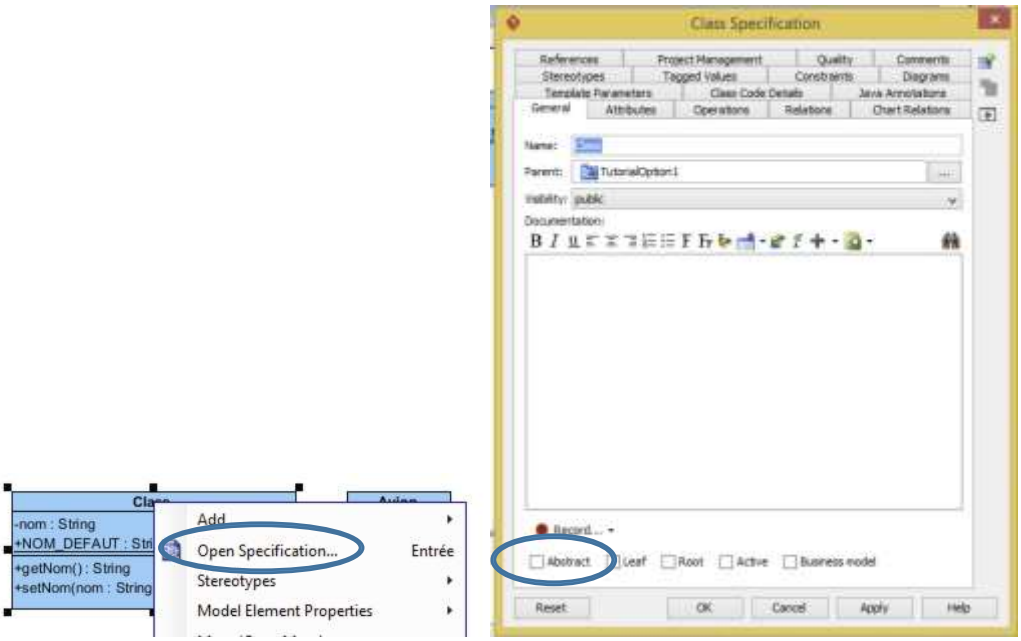
Pour ajouter une classe on peut :

- Faites glisser un élément du modèle directement sur le diagramme
- Cliquez sur le bouton class puis cliquer sur le diagramme à l'endroit où l'on veut voir apparaître la classe. Ici on a fait glisser la classe *Avion* et l'on a ajouté une classe générique *Class*.

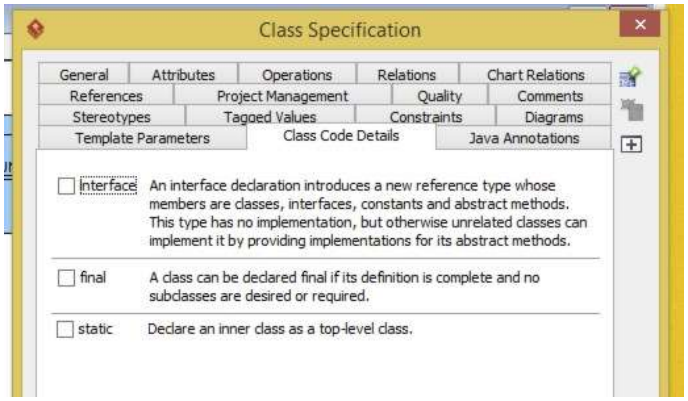


Remarquez que la classe *Class* a été ajoutée dans votre modèle.

Pour spécifier davantage les caractéristiques de votre classe utiliser le menu contextuel sur le nom de la classe et sélectionner *Open specification*. Vous aurez alors plus d'options



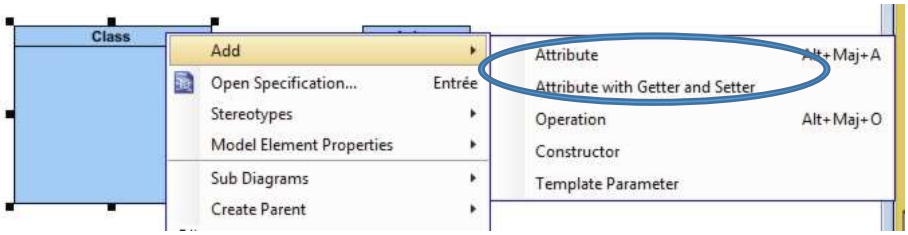
- Dans cet onglet vous pouvez dire que la classe est *abstract* (une classe qui ne peut pas être instanciée).
- Dans l'onglet *Class Code Details*, vous aurez les options suivantes :



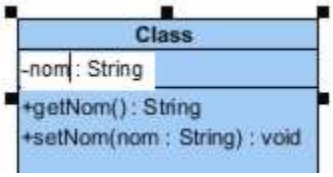
- On peut spécifier s'il s'agit d'une interface.
- Si la classe est *final* (donc qu'on ne peut plus hérité de cette classe)
- Notez qu'en Java une classe ne peut pas être statique (sauf les classes imbriquées)

Ajouter des attributs

Pour ajouter des attributs il suffit de cliquer sur le nom de la classe et de sélectionner le menu *Add / Attribute ou Add / Attribute with Getter and Setter*. Si vous choisissez le second menu, l'accesseur et le mutateur seront automatiquement créés pour votre attribut.



On entre alors en mode Édition dans l'éditeur de diagramme. (On peut également entrer en édition en double cliquant sur un élément).



Notez qu'en UML on déclare un attribut comme suit

<visibilité> <nomDeAttribut> : <type>

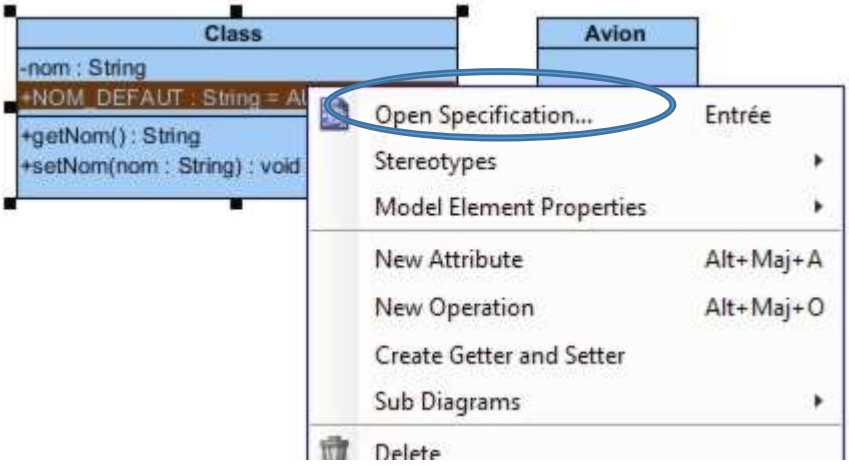
Contrairement au Java le type est à la fin et non au début!

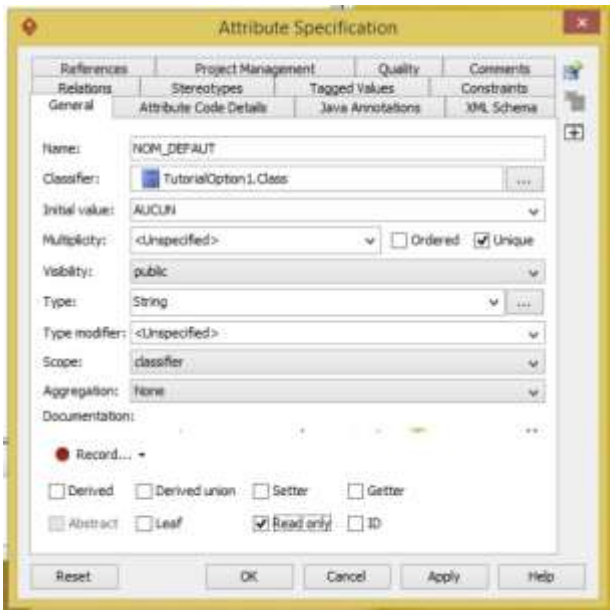
On rappelle que la visibilité d'un attribut doit toujours être privée sauf si c'est une constante.

Ajouter une constante

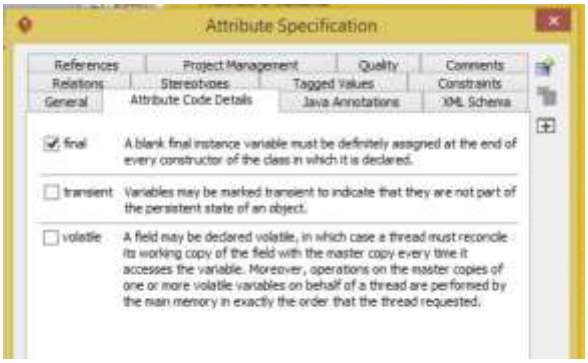
Pour ajouter une constante, il faut d'abord ajouter un attribut. Ici on ajoute l'attribut NOM_DEFAUT. Pour en faire une constante, il faudra ouvrir les spécifications de l'attribut.

- Sélectionnez attribute
- Ouvrir le menu contextuel sur l'attribut
- Choisissez *Open Specification*



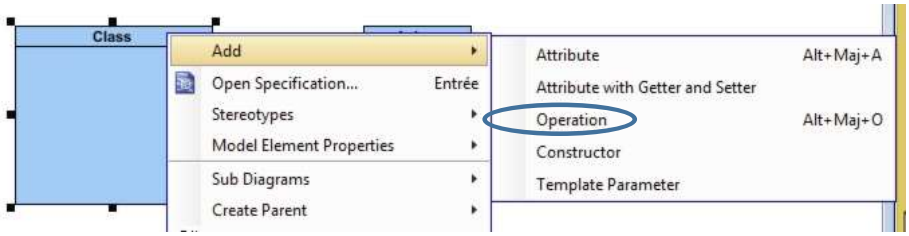


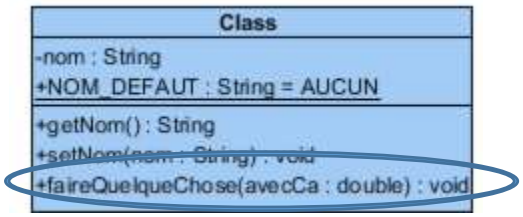
- Une constante est statique. Pour créer un élément **statique** il faut changer le *Scope* d'instance vers classifier.
- On peut mettre la valeur initiale au besoin
- Une constante est souvent *public* on peut donc sélectionner la visibilité *public.*
- N'oubliez pas de mettre le nom de votre constante en majuscule.
- Choisissez le type de votre constante (ici String).
- Cochez *read-only* pour rendre l'attribut final.
- Il faut ensuite aller dans l'onglet *Attribute Code Details*. Cochez la case *final*. (Notez que l'option *transient* est réglable ici).



Ajouter une opération (une méthode)

Pour ajouter des opérations, il suffit de cliquer sur le nom de la classe et de sélectionner le menu *Add / Operation*.



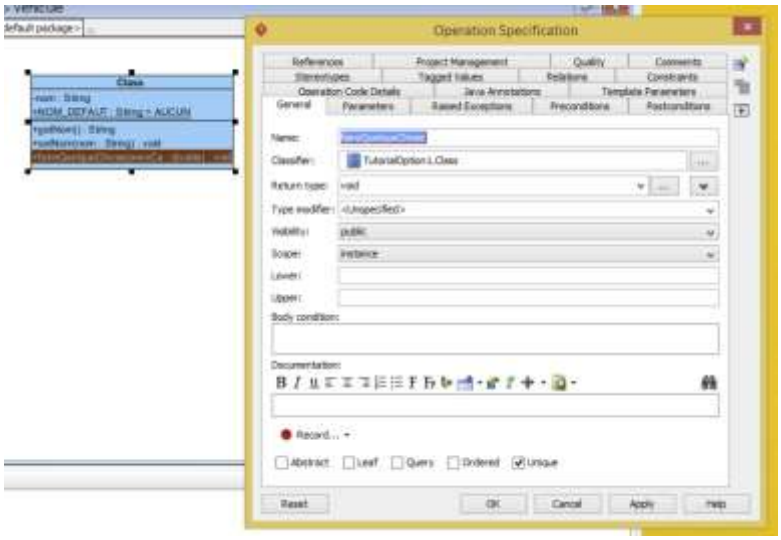


Ici on a ajouté l'opération faireQuelqueChose(). On remarque que la syntaxe est différente de celle du Java.

En UML une opération s'écrit :

<visibilité><nom de l'opération>(<parametres>);<type de retour>

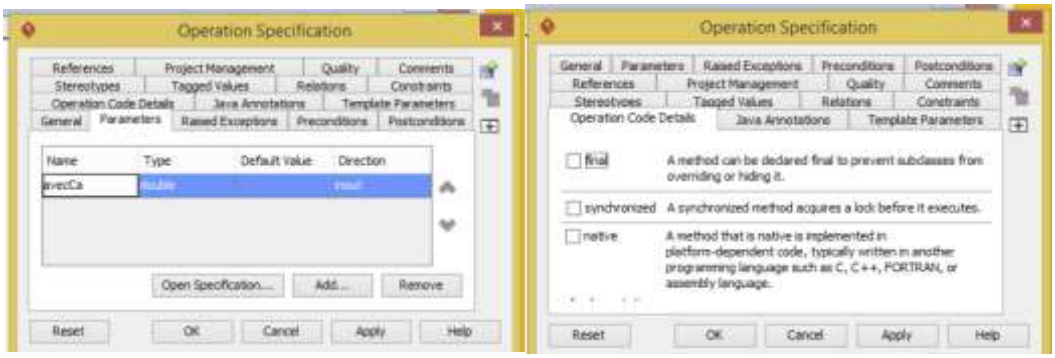
Une « Operation » possède également une boîte de spécification qui peut nous permettre de faire certains réglages:



Ici on peut :

- Modifier son nom
- Son type de retour
- Déterminer si elle est statique (Scope classifier)
- Sa visibilité

L'opération comporte également des onglets pour : spécifier ses paramètres, ajuster les détails du code et...



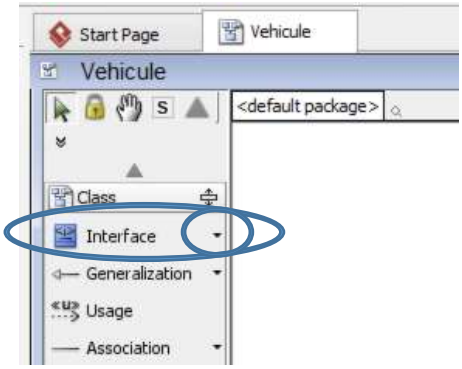
Affichage UML

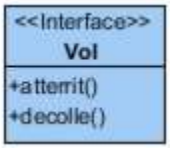
L'UML utilise certaines conventions pour afficher les éléments particuliers. Le tableau suivant en fait un résumé

Élément	Affichage
public	+
privé	-
protégé	#
package	~
statique	Souligné
abstrait	italique
constant	Tout en majuscule

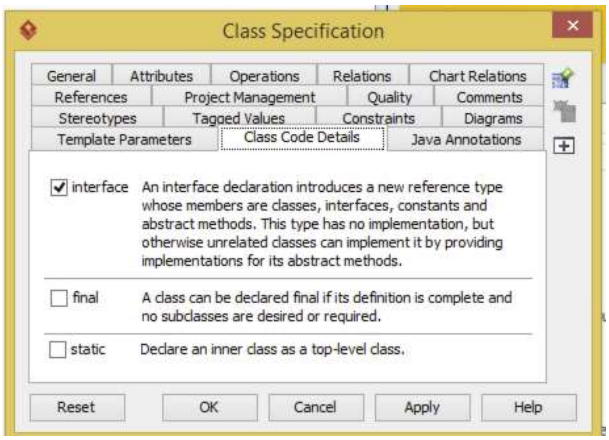
Interface

Pour faire une interface, on procède comme pour la classe, mais on choisit *Interface* au lieu de classe en appuyant sur le triangle dans la section class des outils du diagramme.





En ouvrant l'onglet *Class Code Details* dans les spécifications de l'interface on peut sélectionner Interface. (Cette option ne sert que lorsque le modèle est lié au code)



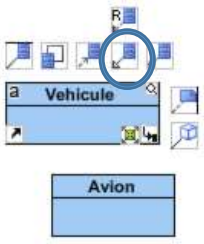
Lien entre les classes

Héritage ou généralisation (extends)

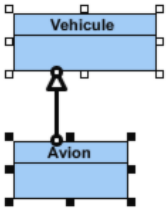
L'héritage montre qu'une classe peut prendre la place d'une autre. Elle est à la base du polymorphisme OO. C'est à dire pouvoir utiliser une variable déclarée

Avec VP pour établir une association d'héritage simplement

- Placez le pointeur de la souris au-dessus de la classe parent
- Cliquer (**sans relâcher le bouton de la souris**) sur la quatrième icône qui est apparue au-dessus de la classe

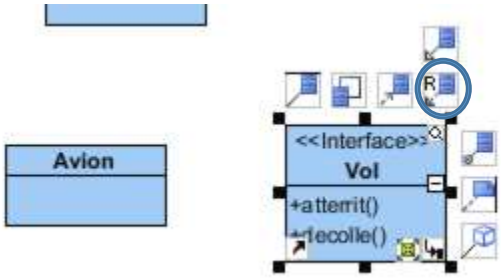


- Glissez le pointeur (**toujours en tenant le bouton de la souris enfoncé**) au-dessus de la classe enfant (ici avion).
- Relâcher le bouton de la souris. Le lien va se créer

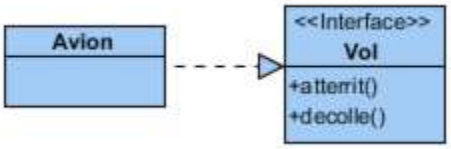


Réalisation d'interface (implements)

On la crée de la même façon que l'héritage. Cependant au lieu de prendre la ligne avec une flèche on choisit la ligne pointillée avec une flèche. Il faut choisir l'interface et glisser vers la classe.



Ce qui donne :



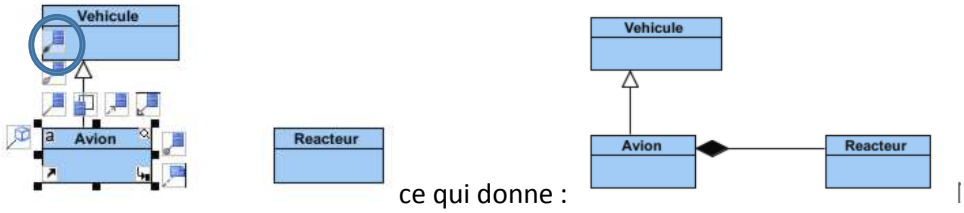
Association

Une association indique qu'une classe possède un attribut dont le type est une autre classe du modèle. Par exemple un *Avion* possède un *Reacteur.* Dans la classe avion on retrouvera une ligne comme :

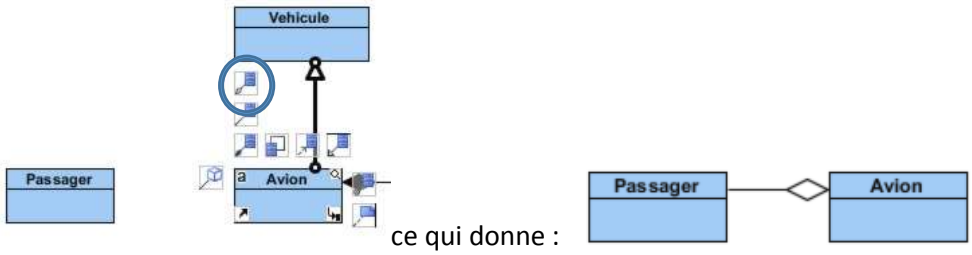
private Reacteur reacteurPrincipal;

Il existe 4 variantes d'associations

- La **composition** : c'est la variante la plus forte, elle représente un lien qui dure pendant toute l'exécution du programme. La classe composante (Reacteur) fait littéralement partie de la classe composée (Avion) elle ne doit jamais être *null*. En UML on la représente en plaçant un rectangle plein sur une ligne qui joint les 2 classes. Le rectangle est placé du côté du composé (Avion). On crée une association de la même manière qu'on crée un héritage, mais en cliquant cette fois sur le losange plein dans la première série d'icônes.

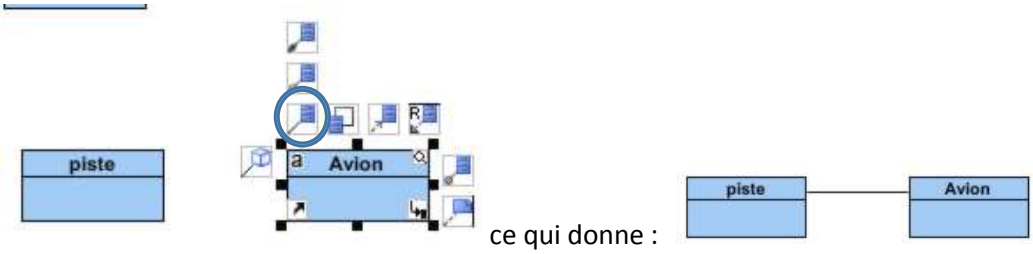


L'agrégation : c'est un lien fort, mais dont l'agrégé pourrait être enlevé à l'agrégeant. Par exemple, on enlève un compte bancaire à un client. Pour créer une agrégation, on utilise le losange vide dans la première série d'icônes.



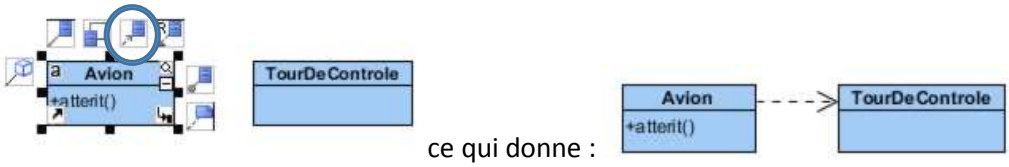
ce qui donne :

L'association simple : c'est un lien neutre. On l'utilise lorsque les liens plus forts ne s'appliquent pas. Par exemple, un *Vendeur* et un *Client* pourraient avoir un tel lien. Pour créer une association simple, on utilise la ligne sans losange dans la première série d'icônes.



ce qui donne :

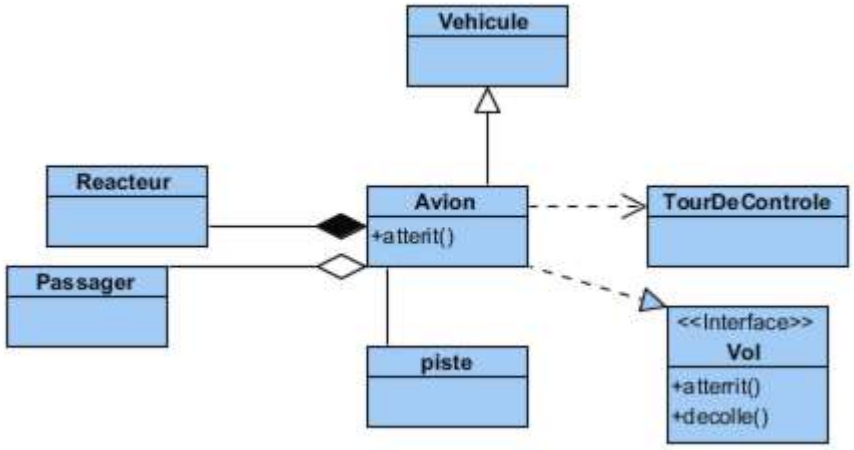
La **dépendance** est l'association la plus faible. Elle se produit lorsqu'une classe utilise une autre classe sans pour autant déclarer un attribut du type de l'autre classe. Par exemple, la méthode *atterrit* de la classe *Avion* pourrait recevoir un paramètre de type *TourDeControle.* Ici la classe *Avion* n'a pas d'attribut de type *TourDeControle,* mais elle aura tout de même besoin de cette dernière pour effectuer son travail (il y aura un énoncé import xxx.*TourDeControle* dans la classe Avion). Pour marquer cette association faible, on utilise la dépendance.



ce qui donne :

Attention de ne pas confondre la dépendance avec la réalisation. Notez que la pointe de la flèche diffère

En mettant toutes les associations ensemble on obtient le diagramme suivant:

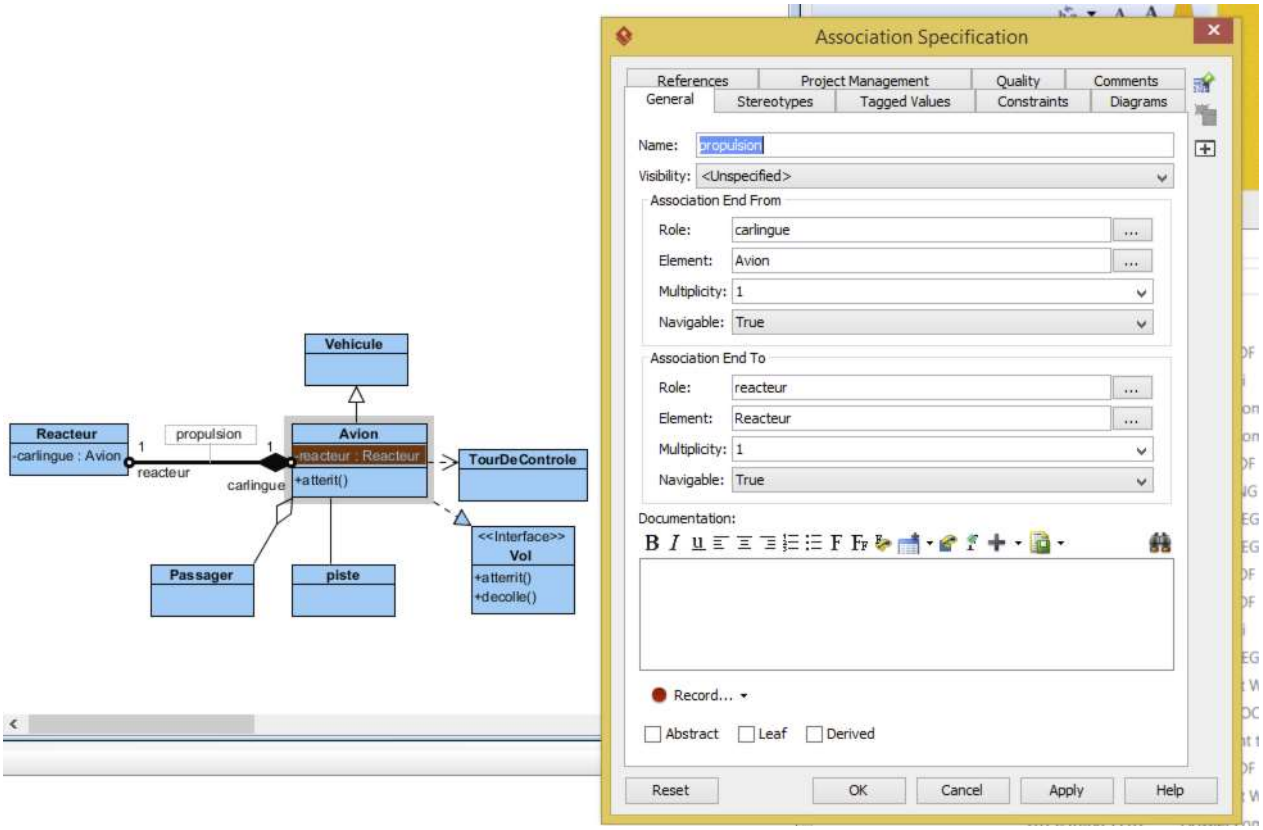


Mieux définir les associations

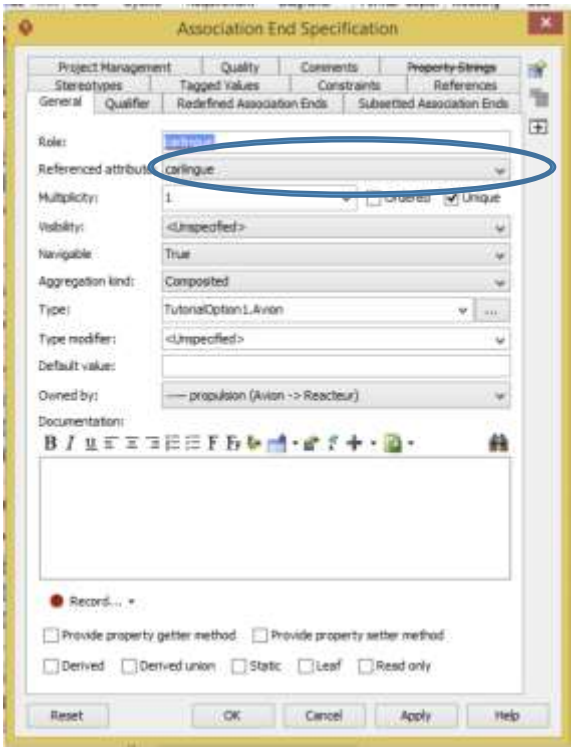
Les associations que nous venons de voir peuvent être précisées davantage.

Le nom et les rôles

En UML l'association est très riche. Elle unit toujours 2 classes qu'on nomme respectivement **rôle A** et **rôle B**. Les rôles servent à identifier ce que chacune des classes fait dans l'association. On peut voir le détail en ouvrant le menu contextuel directement sur une ligne d'association. Par exemple, en demandant les spécifications de l'agrégation dans le diagramme précédent on obtient alors :



- Le nom permet de différencier les associations lorsqu'il y en a plusieurs entre deux mêmes types. Ce nom ne correspond à rien dans le code. Par exemple on pourrait avoir une association nommée propulsionAile et une autre propulsionQueue.
- Le nom des deux rôles (voir Element) permet de différentier chacun des participants de l'association. Ce nom n'apparaît pas non plus dans le code. Cependant, observez qu'il y a un attribut (*carlingue* dans *Reacteur* et *reacteur* dans *Avion*) dans chacune des classes qui est lié à cette association. Pour lier effectivement l'association avec ces attributs on peut cliquer sur « … » à la gauche du rôle pour voir apparaître la spécification du rôle. Dans la spécification du rôle on retrouve le champ *referenced attribute*. C'est avec ce champ qu'on peut lier un attribut avec un rôle dans une association. VP sait désormais que cet attribut prend en charge l'association.

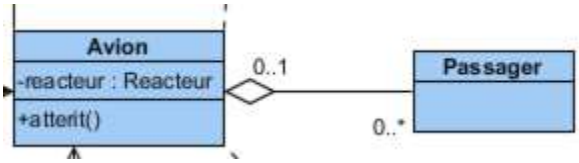


La multiplicité (ou cardinalité)

La multiplicité (parfois appelée cardinalité) permet de définir combien de participants sont liés simultanément à chaque rôle. UML définit les multiplicités suivantes :

Symbole	signification
1	Exactement 1
0..* ou *	De 0 à plusieurs (maximum indéterminé)
1..*	1 ou plusieurs
a..b	de a à b. Si a vaut 3 et b vaut 12, il y aura entre 3 et 12 éléments.
1..a	De 1 à a. a étant un entier
0..a	De 0 à a. a étant un entier

Notez que le chiffre se trouve du côté opposé au diagramme d'entité (utilisé avec les banques de données). Le nombre d'éléments qu'un rôle voit est écrit à l'autre bout de la ligne (ironiquement, près de l'autre rôle). Par exemple avec *Avion* et les *Passager* on devrait avoir



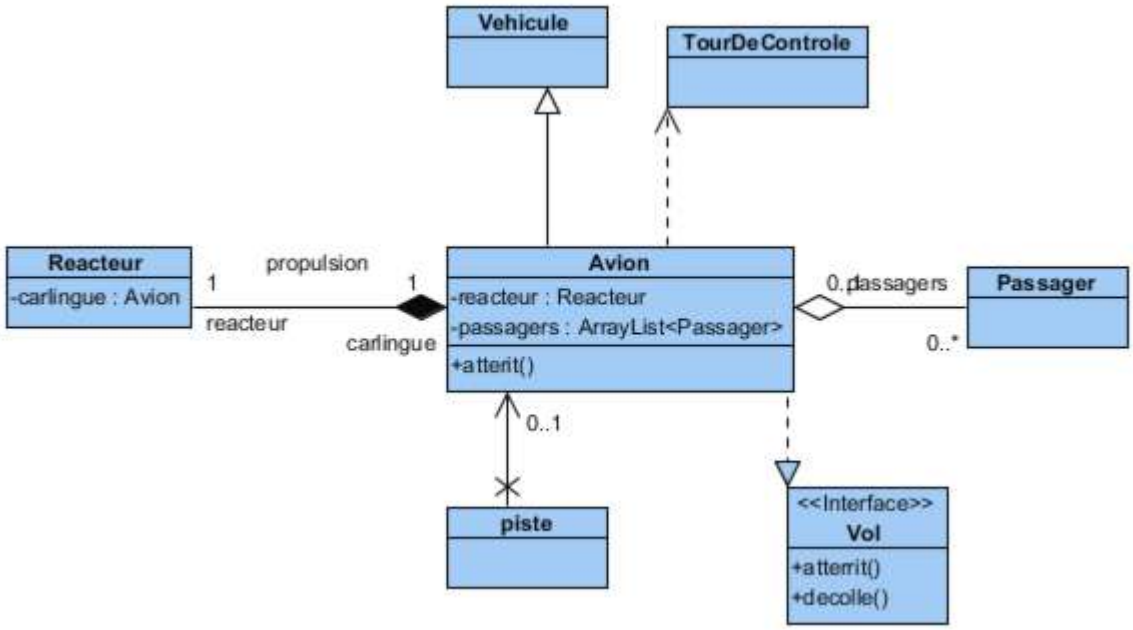
Il y a donc un seul avion de visible par passager (0..1). Mais il y a possiblement plusieurs passagers visibles depuis l'avion (0..*). Évidemment, au niveau du code l'agrégation entre *Avion* et *Passager* va devoir utiliser une collection (comme un ArrayList).

La navigabilité

La navigabilité indique simplement si les classes de chaque côté de la relation peuvent voir la classe qui se trouve de l'autre côté de la relation (via un attribut).

- Si les 2 rôles ne sont pas spécifiés ou s'ils sont tous deux navigables, on a ligne simple sans artifices sur les bouts (association Reacteur- Avion)
- Si un rôle est navigable et l'autre non navigable on obtient une flèche et un x (association Piste-Avion).
- Si un rôle est navigable et l'autre non spécifié, on a une flèche du vers le rôle visible, mais il n'y a pas de x de l'autre côté.

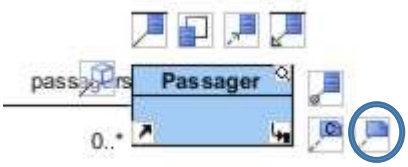
Au final :



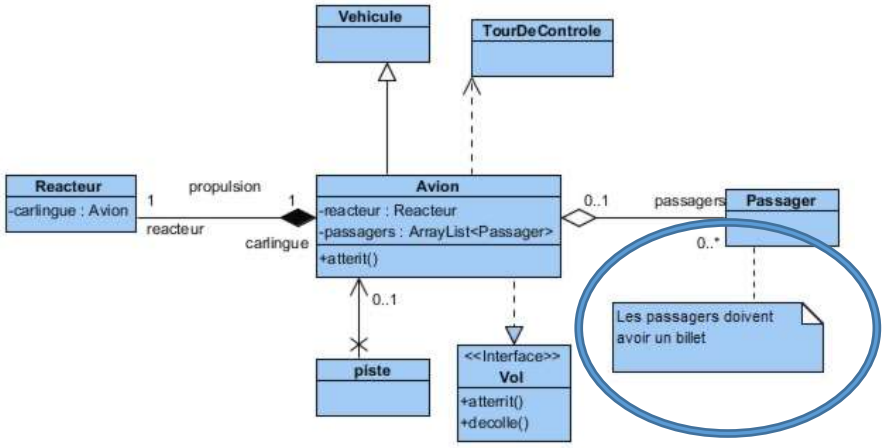
Commentaires

En UML, il est facile d'ajouter un commentaire.

- On clique sur l'icône de note sur la classe à laquelle on désire ajouter une note,



- on glisse jusqu'à l'endroit où l'on veut mettre la note
- On relâche le bouton de la souris et la note va être déposée à cet endroit.

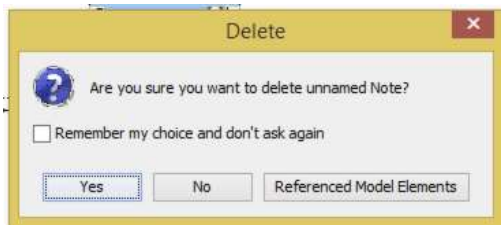


Pour être plus efficace (astuces)

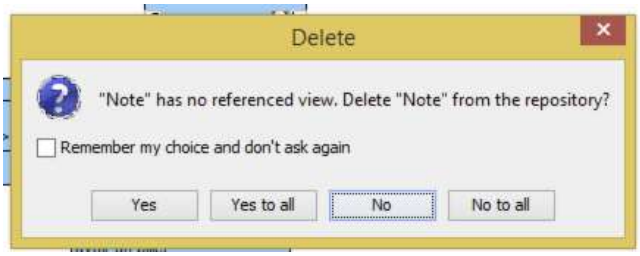
Effacer des éléments

Ici, il faut bien faire la différence entre le modèle et les diagrammes. Le modèle contient tous les éléments que vous avez créés, alors que les diagrammes ne contiennent que les éléments qui sont pertinents pour ce que vous voulez montrer. Dans le modèle on retrouvera l'ensemble des attributs, opérations, association et autres qu'on a définis alors que sur le diagramme on ne retrouve qu'un sous-ensemble pertinent pour ce qu'on veut montrer.

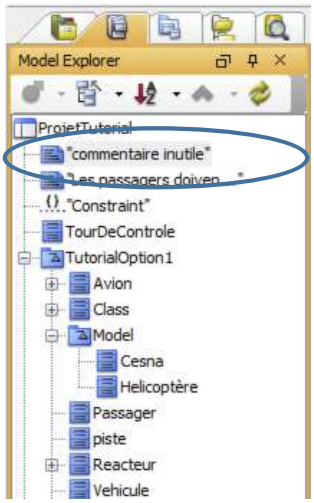
Cette différence entre le modèle et les diagrammes est importante surtout lorsqu'on doit **effacer un élément** sur un diagramme (en appuyant sur la touche *delete*). VP nous demande alors de confirmer que l'on veut bien effacer l'élément



Si l'on répond *Yes*, il nous demande ensuite si l'on veut détruire la note dans le répertoire,



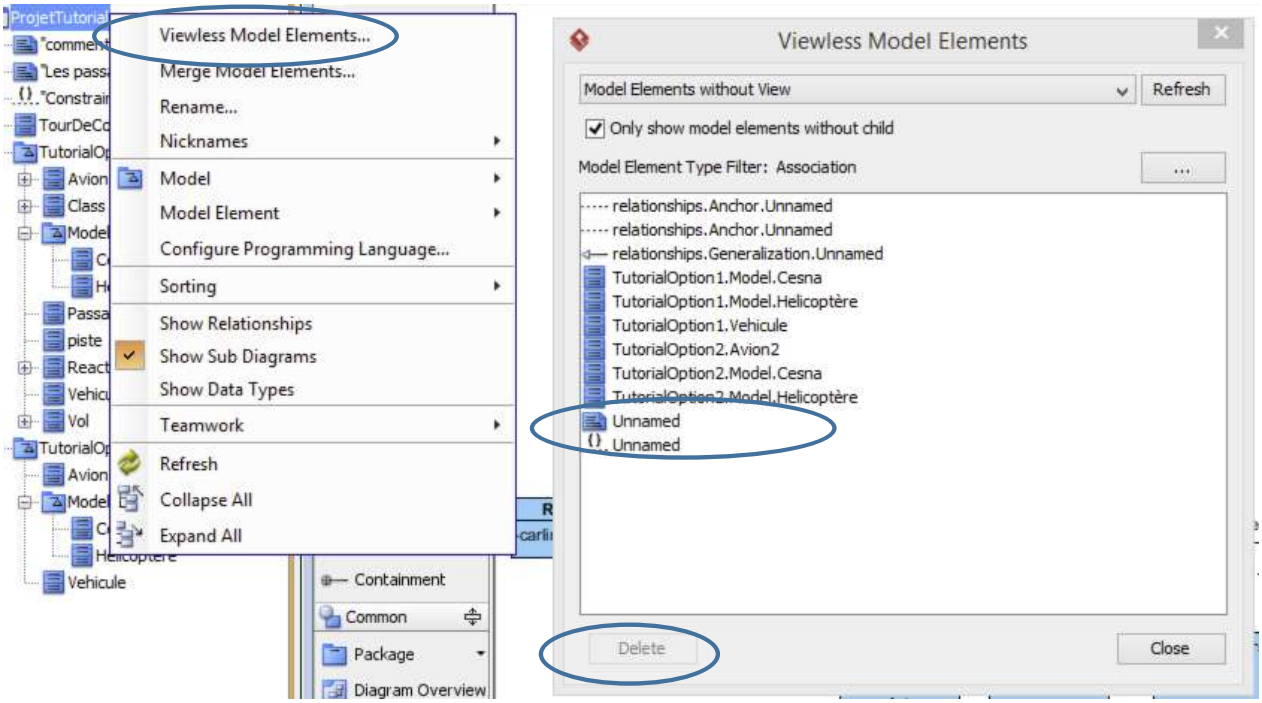
Si l'on répond *Yes* ou *Yes to All* il va détruire l'élément autant sur le diagramme que dans le modèle. Par contre si l'on répond *No* ou *No To All*, il ne l'effacera que sur le diagramme. Par contre, si l'on désire garder un élément dans le modèle (pour qu'il demeure cohérent avec le code), mais qu'on ne désire pas l'afficher (pour rendre le diagramme plus lisible) on peut répondre non à cette question.



Le commentaire inutile a été effacé du diagramme, mais il apparaît toujours dans le modèle.

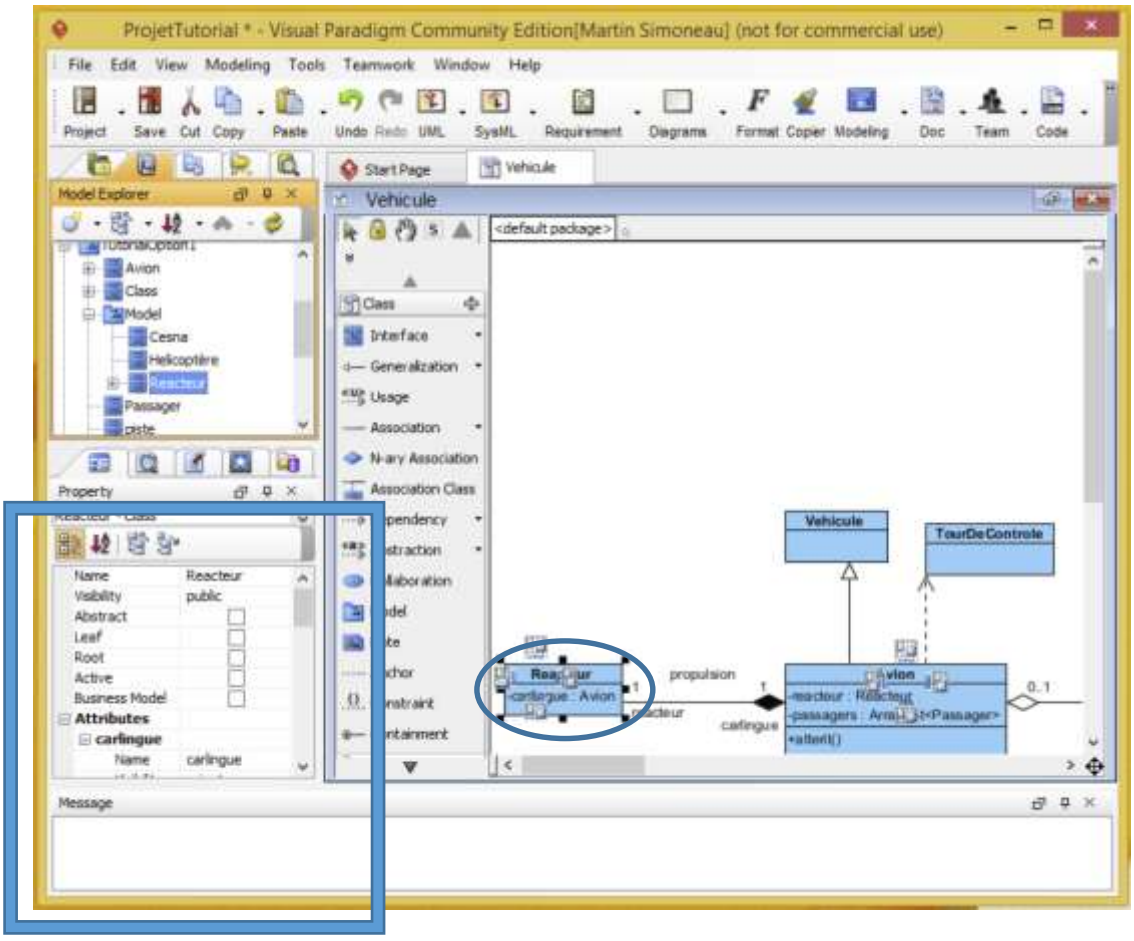
Cette capacité d'effacer des éléments sur le diagramme qui demeurent dans le modèle est très pratique, mais elle peut devenir problématique si l'on ne prend pas garde. Si par mégarde vous effacez un élément important sur le diagramme, mais que vous ne l'effacez pas dans le modèle, lorsque vous essayerez de le remettre sur le diagramme VP va réagir en fonction de ce qui est déjà dans le modèle. Par exemple une association d'héritage ne pourra pas être placée deux fois et VP va refuser de l'ajouter à nouveau.

Pour corriger le problème, il faut alors l'effacer directement dans le modèle. Il existe un outil pour trouver tous les éléments de modèle qui ne sont pas sur un diagramme. Il faut utiliser le menu contextuel sur le projet dans l'explorateur de modèle. On choisit ensuite l'option *Viewless Model element.* VP sort alors un panneau contenant tous les éléments sans vue. On peut alors supprimer directement ceux qui ne sont pas utiles.



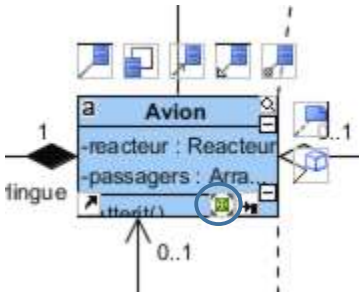
Consulter rapidement les propriétés

Chaque élément du modèle possède plusieurs propriétés qu'on peut consulter en ouvrant le panneau de spécification (comme nous l'avons vu plus tôt). Pour travailler plus rapidement on peut consulter directement les propriétés dans le panneau de propriété situé directement sous l'explorateur de modèle

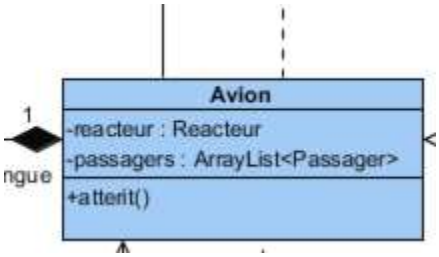


Redimensionner automatiquement

Lorsqu'on modifie le contenu d'une classe il arrive fréquemment que cette dernière n'affiche pas bien son contenu.



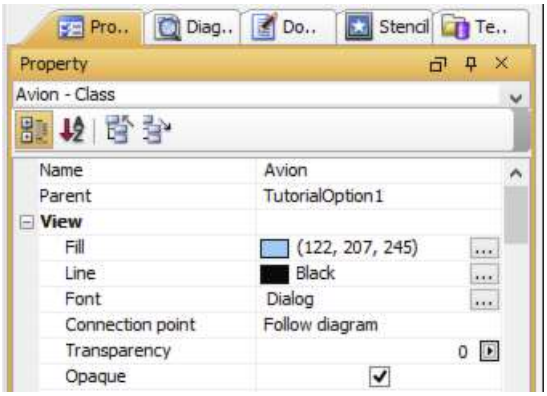
Pour laisser VP ajuster automatiquement les dimensions d'une classe à son contenu, on peut double cliquer sur l'icône vert en bas à droite de la classe. On obtient alors



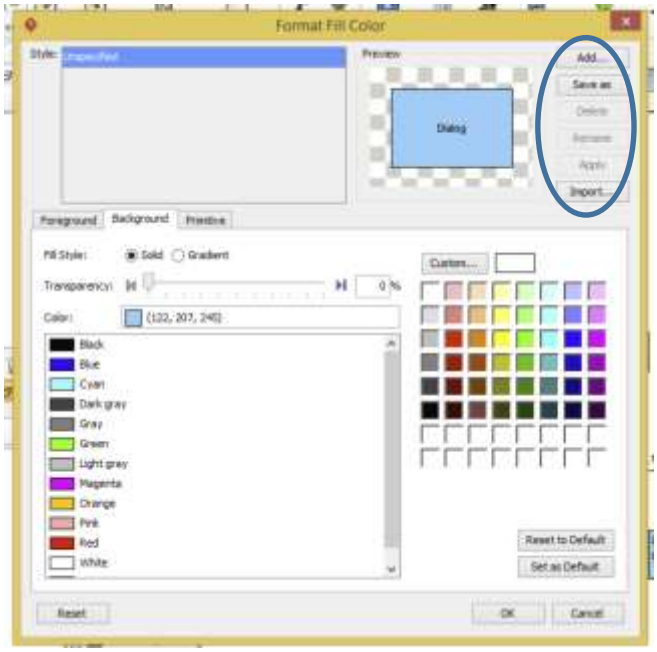
Dans les propriétés du diagramme il y a également une option *auto fit shape*. Lorsque cette option est cochée, VP redimensionne automatiquement tous les éléments du diagramme pour afficher tout leur contenu.

Style Couleur et formatage

Les diagrammes UML doivent être simples à comprendre. Pour faciliter la compréhension, on peut modifier l'apparence des éléments en modifiant les propriétés *View*.



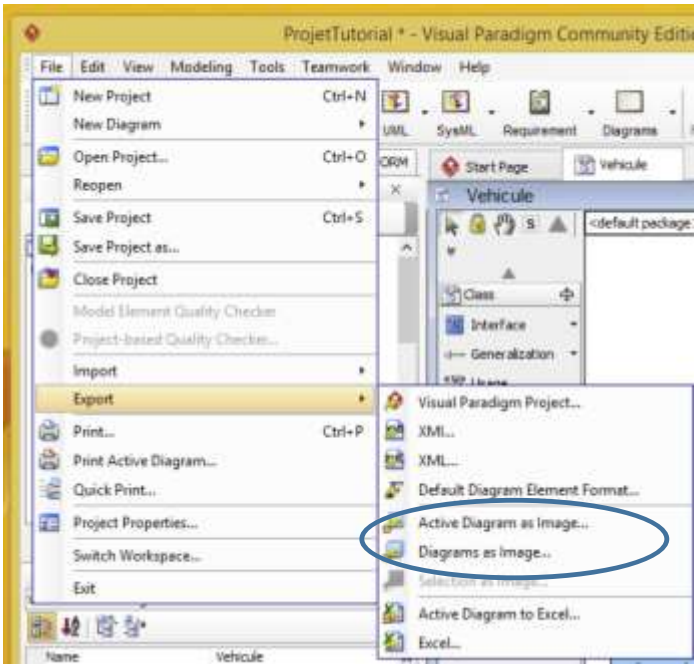
Ces propriétés permettent de modifier l'Affichage (remplissage, ligne de contour, opacité, police de caractère…). Mais ce qui est plus intéressant avec VP c'est qu'il prend en charge des styles qu'on peut créer nous-même. En cliquant sur les « … » à gauche d'une propriété d'affichage (*fill* par exemple). VP ouvre une boite de dialogue qui permet définir des styles. Et donc de faire des ajustements cosmétiques consistant dans tous les diagrammes.



Exporter des images

Pour transformer un diagramme en image, il faut l'exporter.

- On peut le faire directement à partir des menus. *File/ Export / Diagram as Image i ou File/ Export / Active Diagram as Image.* On peut alors exporter en *jpeg*, *svg* ou en *pdf*.



- On peut également exporter une image en utilisant le mécanisme de copier/ coller. On sélectionne uniquement les éléments à exporter puis on fait une copie spéciale.