

SAE 2.01

Rapport de la semaine 1

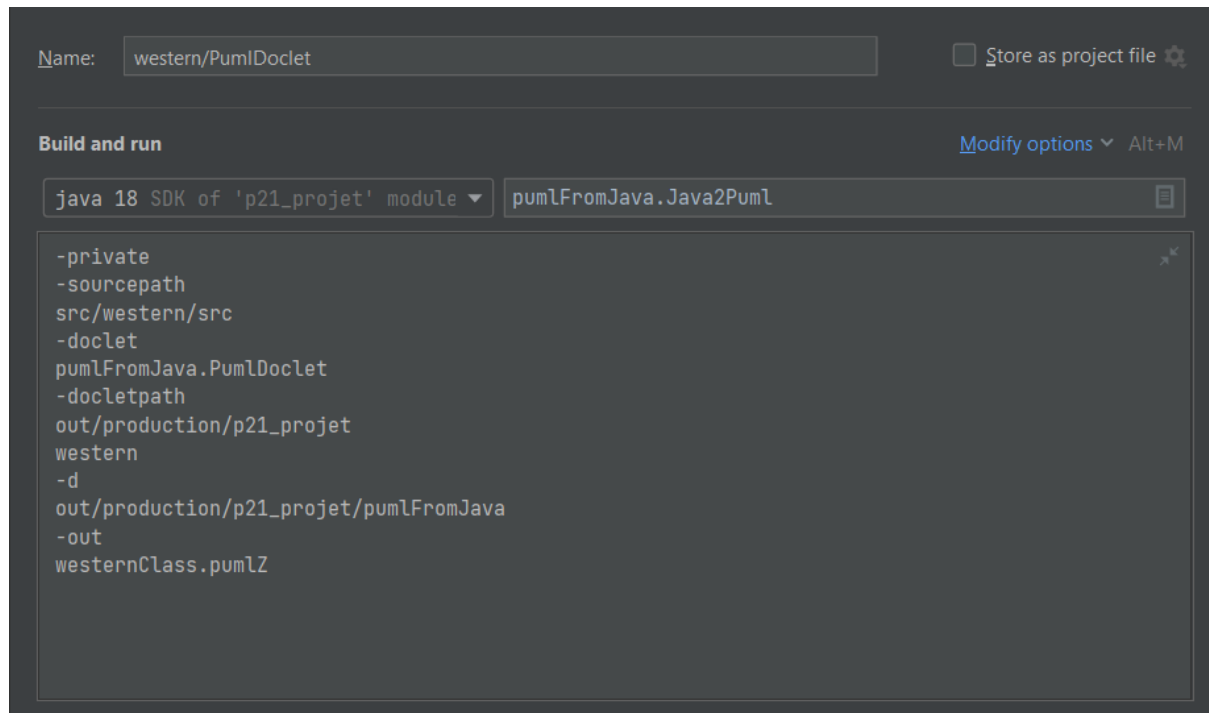
Sommaire

Sommaire.....	1
Rattrapage du travail de la première semaine.....	2
Règlement du problème des options non reconnues.....	2
Traitement des fichiers.....	2
Génération du fichier de sortie.....	3
Génération du DCA par l'API.....	3
Moyens utilisés.....	3
Récupération des attributs.....	4
Ecriture dans le fichier.....	4
Production des diagrammes de classe de l'API.....	4
Production du DCC de java.lang.model.....	6

Rattrapage du travail de la première semaine

Règlement du problème des options non reconnues

Lors du dernier rapport, le projet était bloqué au stade de l'ajout des options où l'option `-out` ne fonctionnait pas. Il se trouvait qu'il manquait des arguments dans la configuration d'exécution du programme. Après l'ajout de ces arguments, le programme a enfin pu s'exécuter sans erreur et cela nous a permis d'avancer sur le projet.



Configuration d'exécution pour le package western

Traitement des fichiers

Un fois l'exécution réussie, nous avons essayé de trouver un moyen de traiter tous les fichiers que le doclet a récupéré. Pour cela on a créé une classe `PumlDiagram` qui contient toutes les étapes de conception du diagramme, qui reçoit toutes les informations à traiter par `PumlDoclet` et qui sera lancé par ce dernier.

Afin de trouver le nom de chaque fichier, on a utilisé le paramètre passé dans la méthode `run` du `PumlDoclet` et on a utilisé sa méthode `getEnclosedElements()` qui nous a donné accès à tous les fichiers. On a ensuite utilisé une boucle pour traiter tous ces fichiers et donner à l'instance de la classe `PumlDiagram` le nom de ces fichiers.

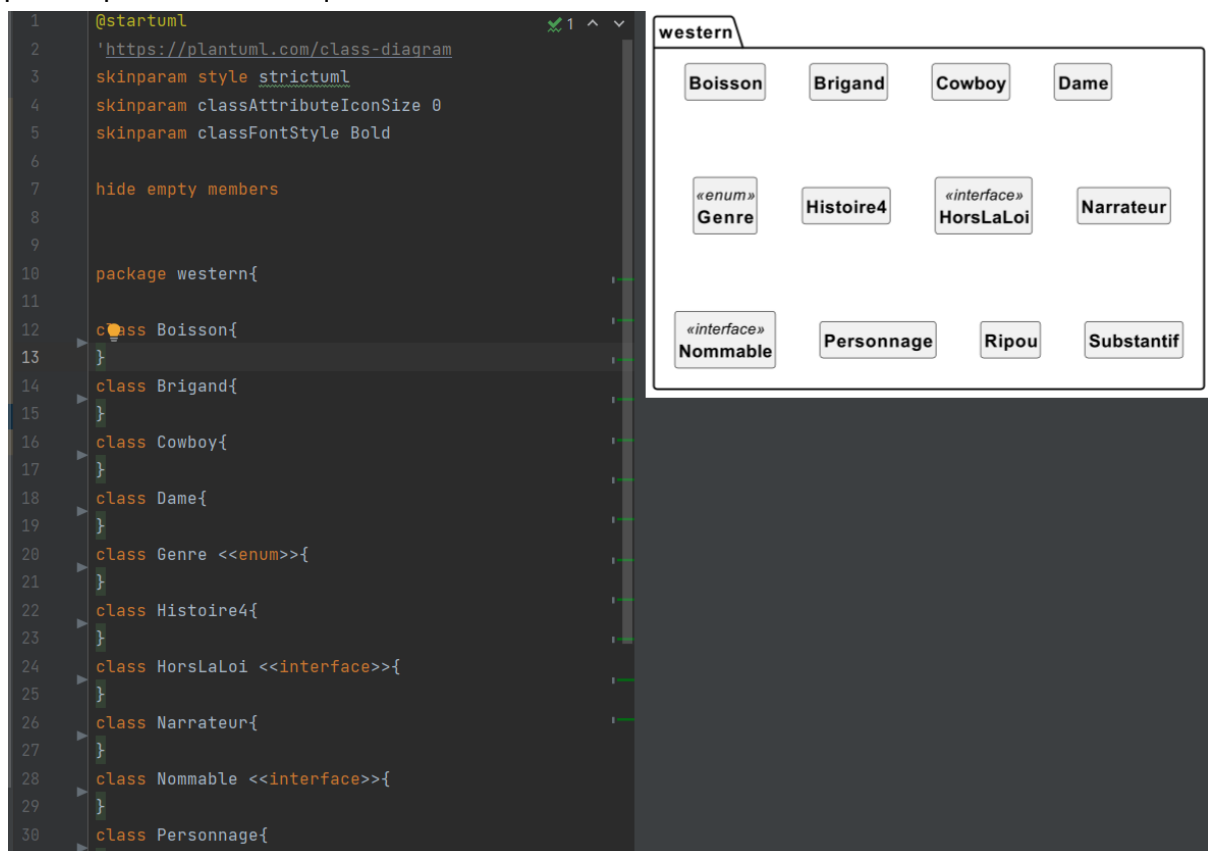
A ce moment-là, la classe `PumlDiagram` a été conçue pour seulement stocker les noms dans un tableau de `String`, ce qu'on a ensuite abandonné pour le travail de la deuxième semaine. Ce tableau de `String` était ensuite traité pour aller dans une variable de type `String` contenant l'ensemble du fichier Plant UML qui était construit au fur et à mesure du traitement.

Génération du fichier de sortie

Une fois le contenu du fichier puml créé et stocké dans une variable, on a cherché à créer un nouveau fichier et écrire dedans, pour cela on a importé deux packages : `java.nio.file.Files` et `java.nio.file.Paths`. Ces deux packages nous ont permis de créer un fichier à l'aide du type `File` et d'ensuite écrire dans le fichier.

Avant de créer un fichier, nous avons vérifié qu'il n'existait pas déjà, dans ce cas-là, le fichier existant est vidé et ensuite on écrit dans le fichier avec la variable contenant l'ensemble du fichier.

A la fin de l'exécution, un fichier a été créé dans le répertoire spécifié dans les arguments du lancement du programme et le diagramme attendu pour la première séance a pu être produit automatiquement.



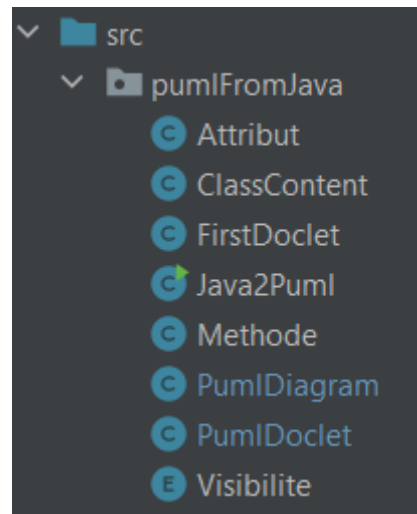
Résultat de l'exécution du doclet

Génération du DCA par l'API

Moyens utilisés

Une fois le retard de la première semaine rattrapé, on a commencé à chercher des moyens d'obtenir les attributs et aussi de les stocker pour les traiter ensuite. On a donc conçu trois nouvelles classes et une énumération. La première classe se nomme `ClassContent`, elle stocke son nom, son type (classe, énumération, ...), sa liste d'attributs et sa liste de méthodes. Les deux autres classes concernent les attributs et méthodes de la

classe précédente, elles sont identiques et stockent leur nom, leur type ainsi que leur visibilité.



Ensemble des fichiers nécessaire au fonctionnement du doclet

Récupération des attributs

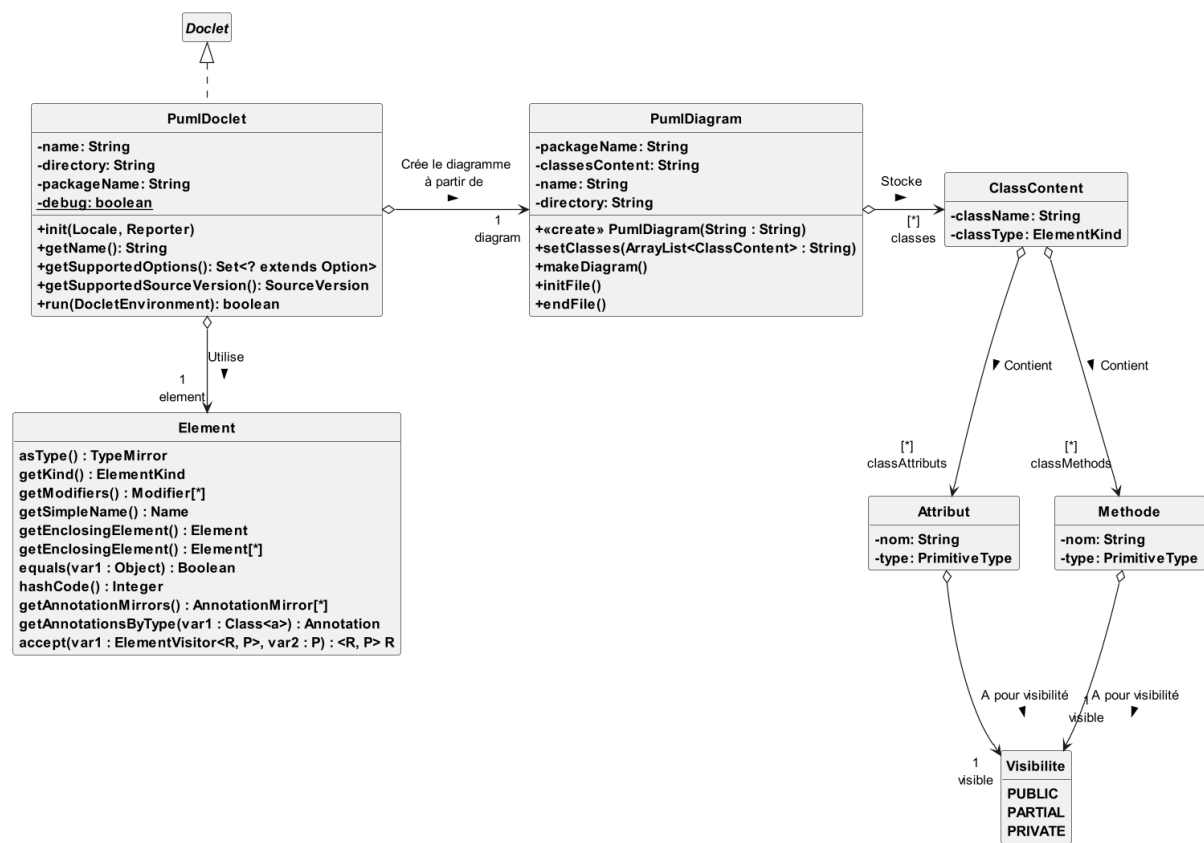
Une fois les nouvelles classes créées, on a une fois de plus utilisé la méthode *getEnclosedElement()* mais elle a été utilisée cette fois-ci sur les fichiers pour obtenir les éléments internes. Grâce à cela, on a pu obtenir à la fois les méthodes, les attributs ainsi que d'autres informations sur les fichiers, on a cependant trié ces informations pour seulement avoir les attributs afin de faire le DCA. Après avoir récupéré les attributs, on les a d'abord traité afin qu'il n'y ai que les types primaires en triant les nom de leur type, on les a ensuite ajoutés dans une instance temporaire de *ClassContent* qui sera ensuite transférée à *PumlDiagram*.

Ecriture dans le fichier

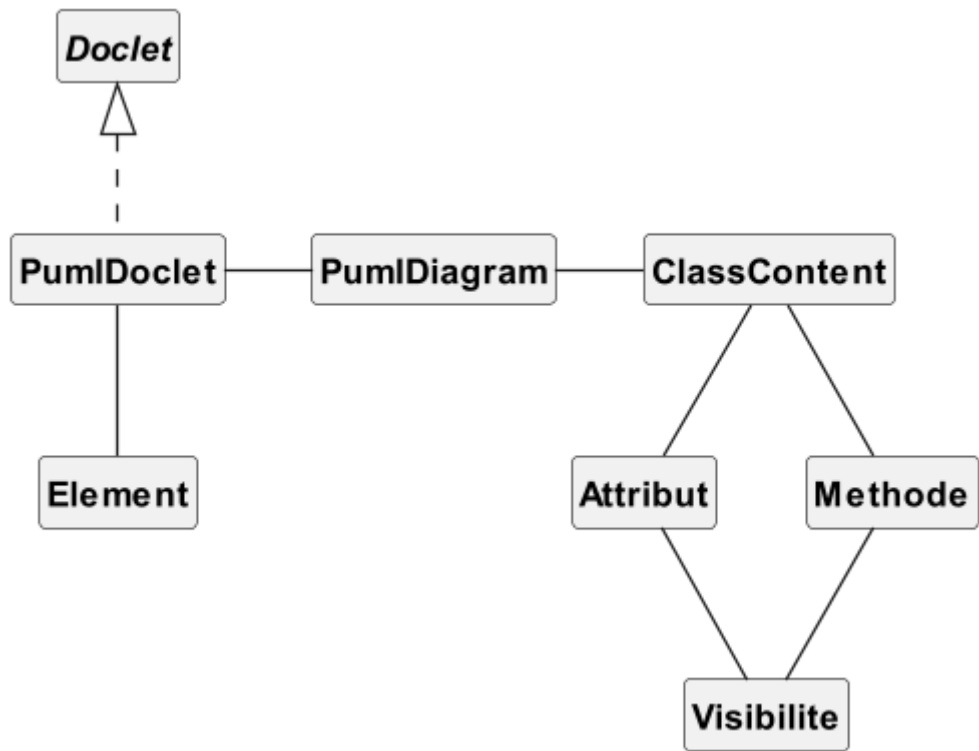
Nous avons repris la façon d'écrire dans les fichiers que nous avons utilisés pour le contenu de la semaine précédente et nous l'avons modifié pour afficher en plus les attributs. Pour cela, entre chaque classe, on a simplement ajouté le nom de l'attribut dans la chaîne de caractère et ajouté un retour à la ligne.

Production des diagrammes de classe de l'API

Pour faire les diagrammes de classe de notre API, on a bien sûr attendu d'avoir fini le code afin d'éviter de tout changer en cours de route. La production des diagrammes représente actuellement l'ensemble des éléments nécessaires à la conception du diagramme.



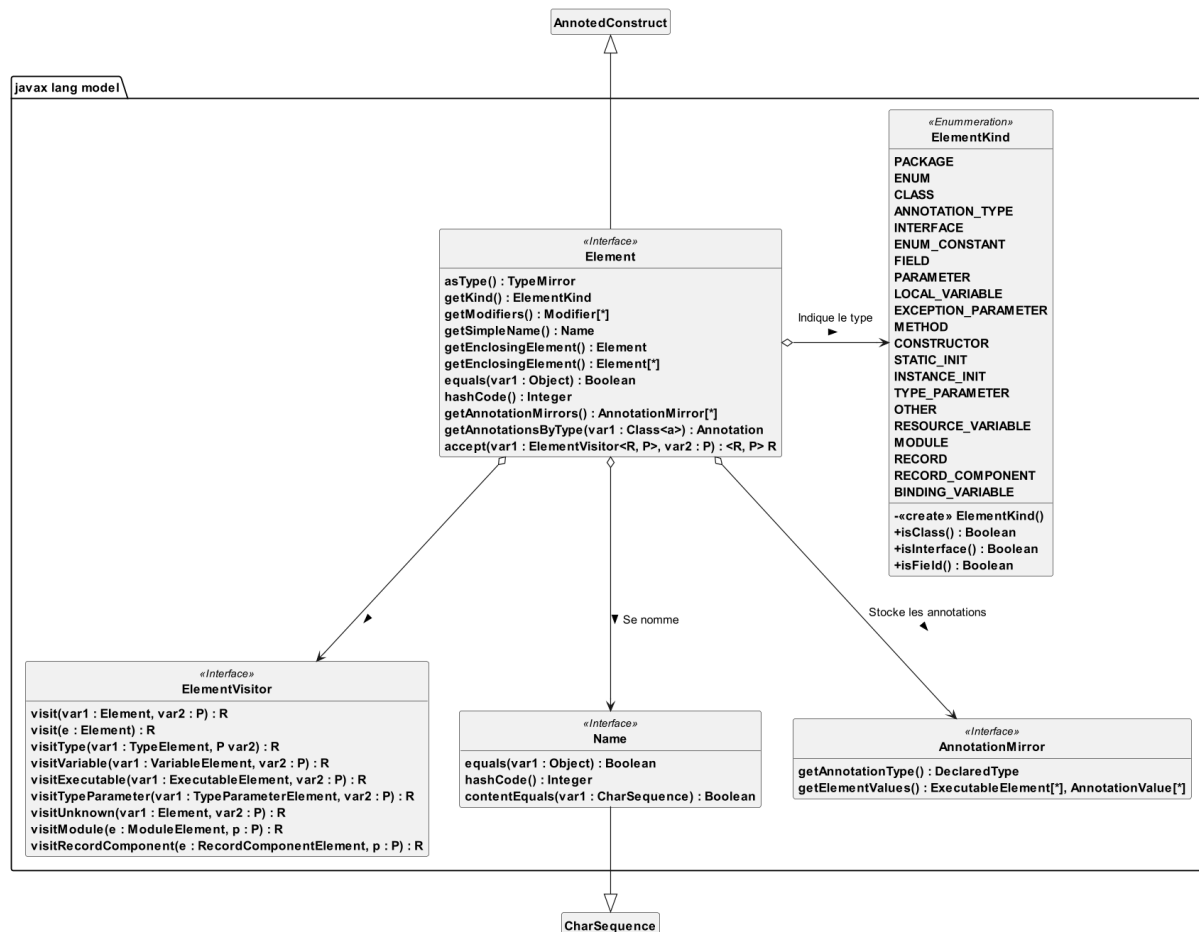
DCC de l'API PumlDoclet



DCA de l'API PumlDoclet

Production du DCC de java.lang.model

La conception du DCC du package javax.lang.model s'est effectuée sans problème. Le choix des classes, interfaces et énumérations s'est fait si elles apparaissent dans la classe Element.



DCC du package javax.lang.model à partir de l'interface Element