SAE 2.01
Rapport de la semaine 6

Sommaire

Gestion de l'affichage des méthodes

L'un des objectifs que l'on s'est fixé la semaine dernière est de gérer l'affichage des méthodes déjà appelées dans leur super-classe. Pour cela, on a modifié la classe Méthode pour qu'elle prenne en compte si elle sera visible ou non avec le booléen isPumlVisible.

Ensuite, la classe PumlDiagram s'est vu ajouté une nouvelle méthode afficheMethodeHeritage() qui regarde dans toutes les classes et interfaces si elles héritent d'une super-classe et qui va ensuite comparer si elles ont des méthodes en commun ayant le même nom et le même type de retour.

Cependant, comme il était demandé pour cette semaine, il fallait vérifier si les méthodes étaient réécrites dans la classe enfant et les ré-afficher. Nous n'avons pas pu nous occuper de cette éventualité même en regardant la documentation java. C'est-à-dire que même si les méthodes sont réécrites, elles n'apparaissent tout de même pas.

```java
// Méthode pour afficher ou non une méthode
1 usage  daonguyen.tran *
public void afficheMethodeHeritage(){
    for (Association association : associations){
        if (association.getTypeAssociation() == TypeAssociation.HERITAGE){
            for (Methode methods1 : ((ClassContent)association.getElement1()).getMethodes()){
                for (Methode methods2 : ((ClassContent)association.getElement2()).getMethodes()){
                    if (methods1.getNom().equals(methods2.getNom()) && methods1.getType() == methods2.getType()){
                        methods1.setToPumlInvisible();
                    }
                }
            }
        }
    }
}
```

Méthode de tri des méthodes

Finalisation de l'API

Corrections

Le travail de cette semaine s'est principalement concentré sur la correction de bugs et erreurs que l'on avait rencontré et en organisant l'API. Nous allons les énumérer par type de modifications.

Associations

Les associations ont représenté une bonne partie des rectifications de la semaine. Nous avons corrigé le bug des affichages multiples des dépendances, nous les avons aussi effacés si une autre association existait déjà entre les deux éléments.

Nous avons aussi géré le problème des multiplicités qui ne doivent pas s'afficher au départ de la flèche pour les agrégations et compositions, il en est aussi de même pour les crochets autour des * pour ces mêmes associations.

L'une des plus grandes modifications pour les associations a été la façon dont elles sont générées. En effet, lors du rendu de la semaine dernière, les diagrammes étaient étalés sur la hauteur parce que chaque association mettait obligatoirement le premier élément au-dessus de l'autre. Pour régler ce problème, chaque instance des ElementContent dans PumlDiagram ont un entier correspondant à l'étage Puml auquel ils essaieront d'être.

Éléments

Les éléments ont aussi reçu beaucoup de modifications. Les attributs des ClassContent ont maintenant leur vrai affichage Puml au lieu de leur nom. Les EnumContent ont eu le même changement mais dans le sens inverse, ils ont juste le nom de leur attribut au lieu d'un affichage d'attribut de classe.

Un autre changement est aussi la correction du double affichage des paramètres dans les méthodes.

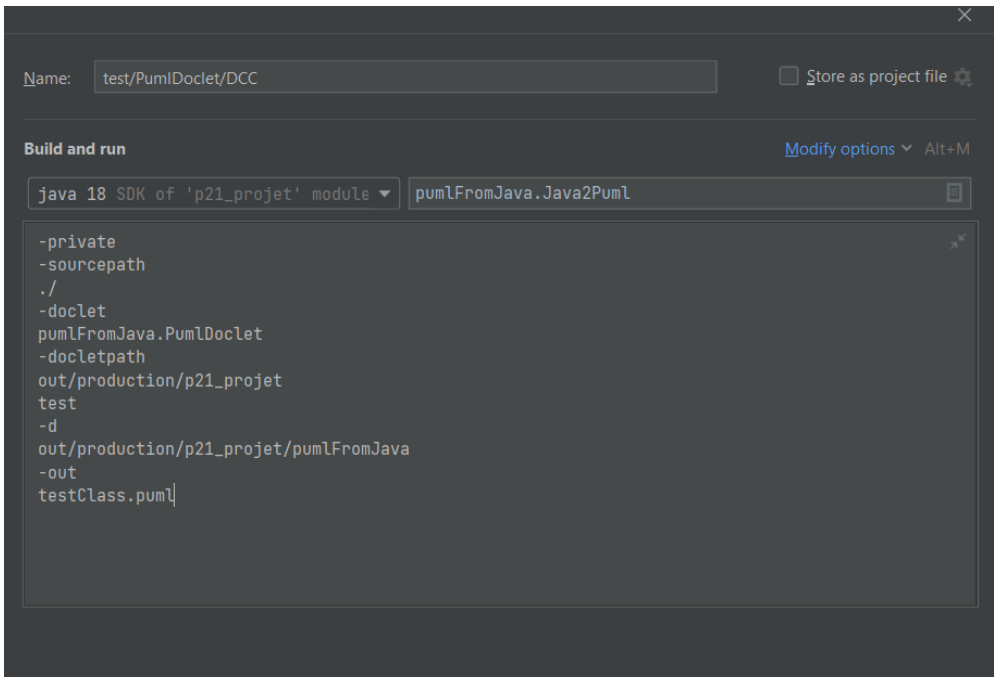
Autre

D'un point de vue plus global, nous avons changé la façon dont les visibilités sont gérés pour afficher les types de base des attributs, que ce soit sur les associations ou les attributs et méthodes. Nous avons aussi revu la détection des collections en regardant si le TypeMirror est de type Declared et qu'il a au moins un argument.

Package 'test'

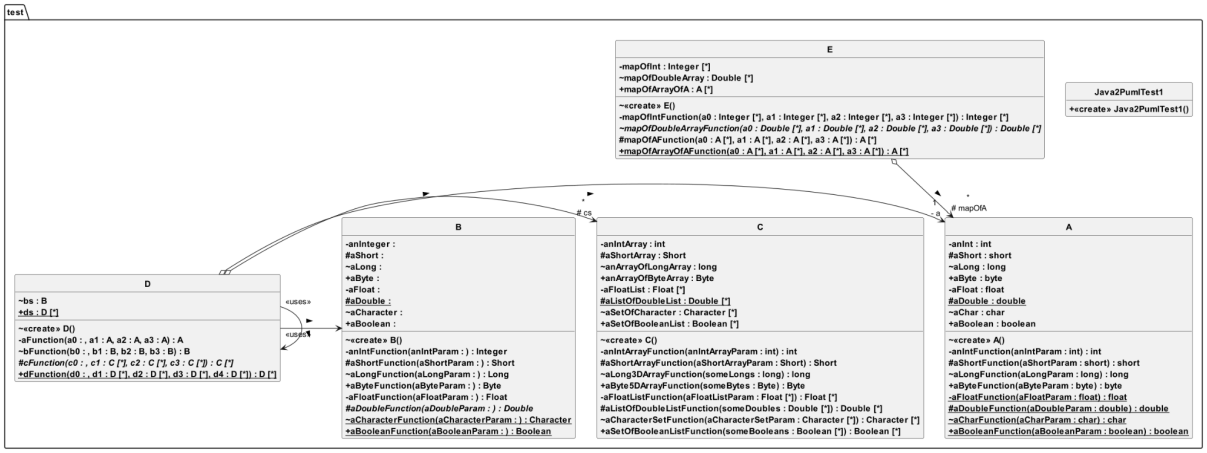
Au début de la semaine, nous avons reçu un mail pour tester notre API sur un nouveau package nommé *test*. Le package nous a aidé à voir tous les problèmes de notre API, et donc de pouvoir nous corriger.

Pour générer le diagramme avec notre API, nous avons ajouté dans le fichier Java2PumlTest1.java le nom de package '*test*' et nous avons créé une nouvelle configuration pour la tester.

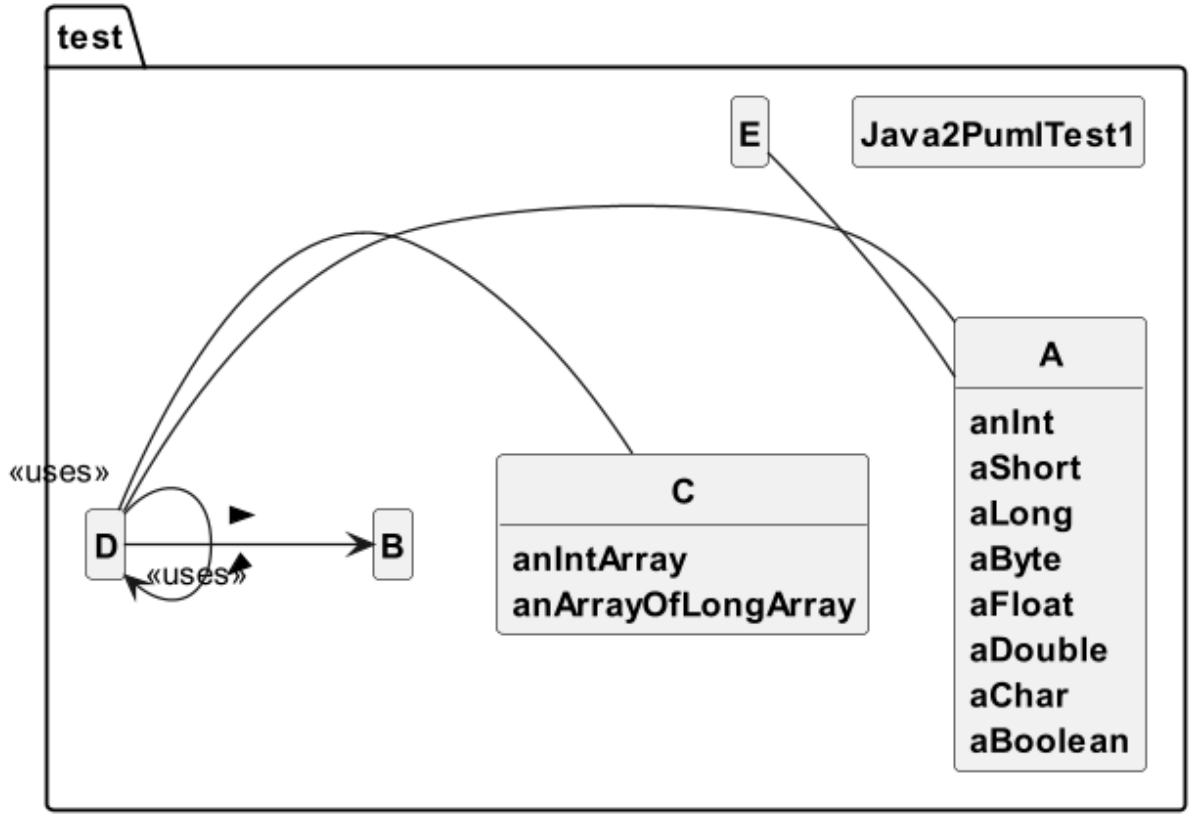


Configuration d'exécution pour le DCC du package test

Voici le DCC et DCA qui ont été produit sur ce package :



DCC du package test

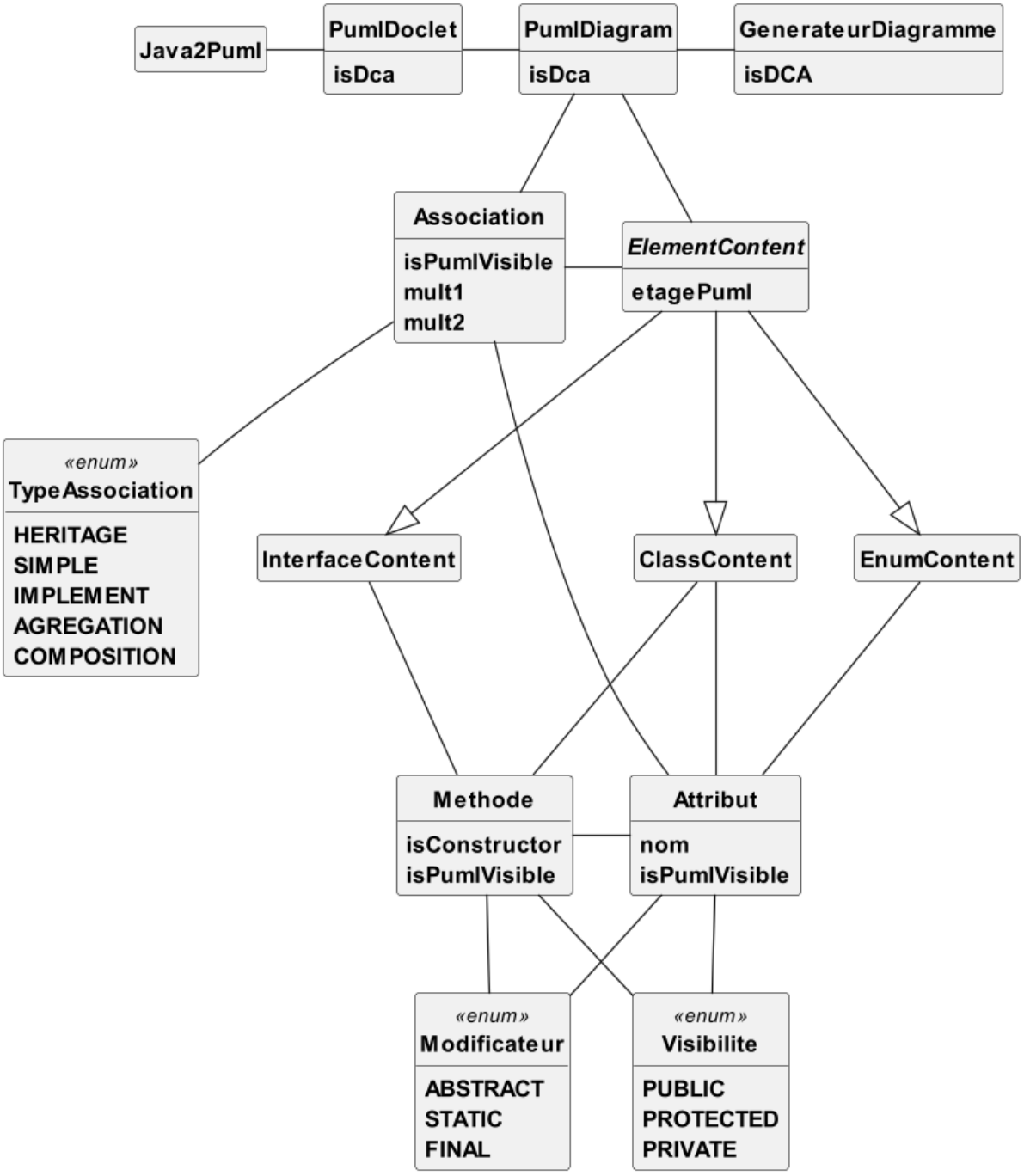


DCA du package test

DCA et DCC finaux de l'API

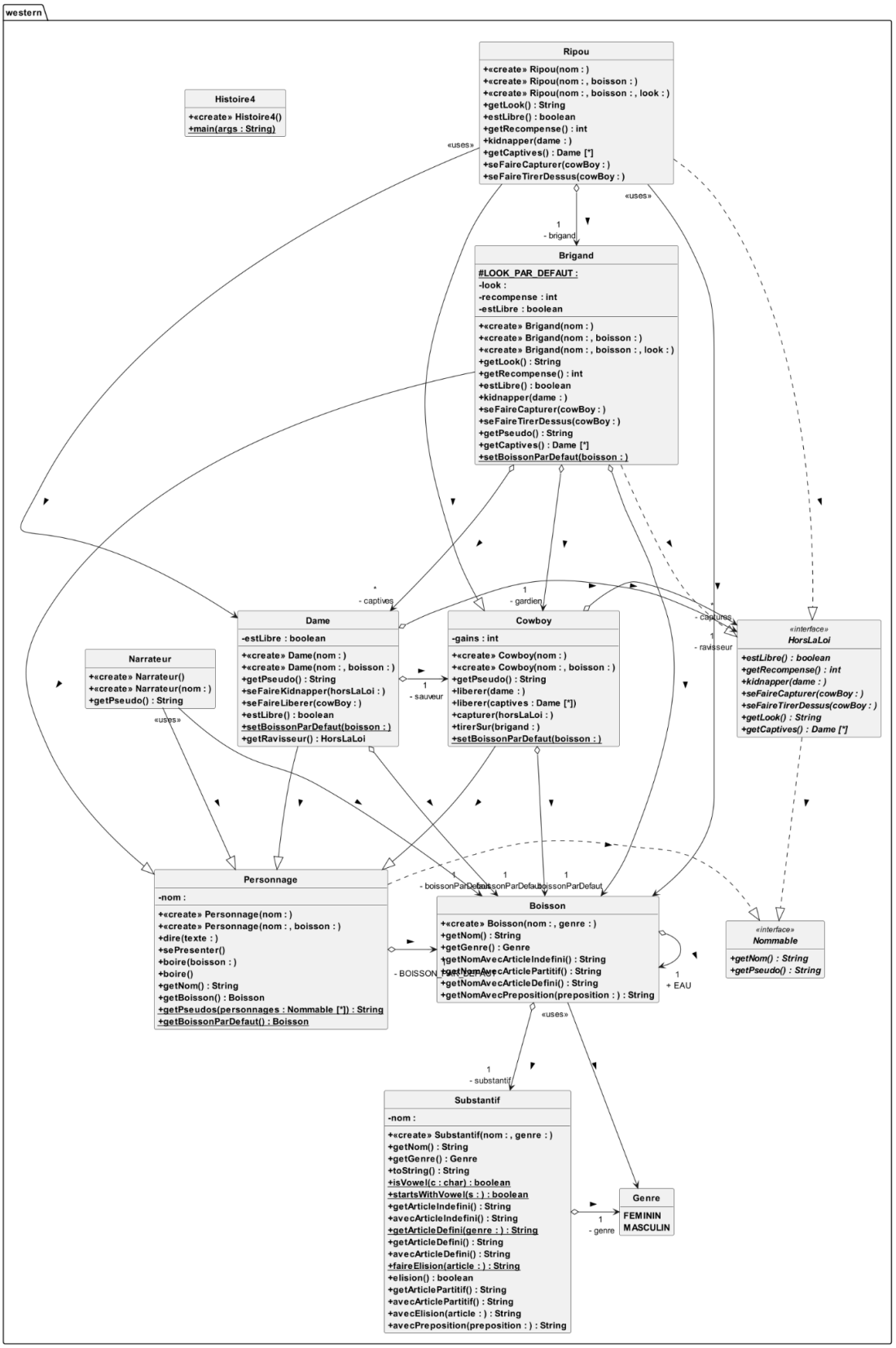
Pour produire le DCA et le DCC final, nous avons pensé à ajouter les méthodes non présente de la dernière version et nous avons supprimé les attributs non primitifs dans le DCA :

DCC du package pumlFromJava

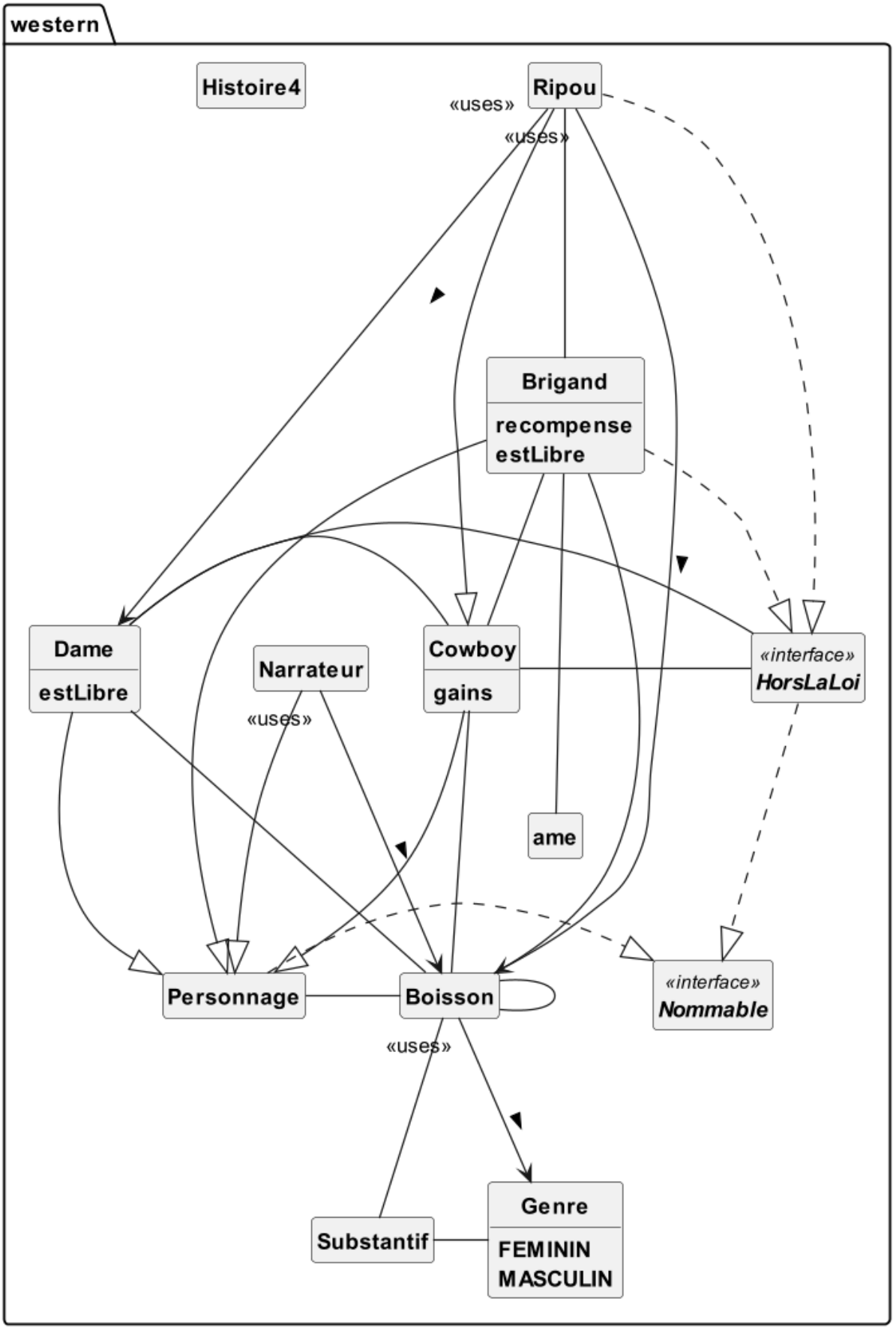


DCA du package pumlFromJava

Diagrammes de classe générés



DCC du package western



DCA du package western

Ressenti personnel

Daonguyen Tran

Au départ, cette SAE me paraissait très difficile et ennuyeuse puisque je n'arrivais pas à comprendre les moyens que l'on devait utiliser pour faire cette API. De plus, le fait qu'on était bloqué au début et qu'on ne comprenait pas le problème m'a découragé et on a donc pris du retard sur la deuxième semaine.

Cependant, avec l'aide de M. Krähenbühl, on a été débloqué et on a pu se lancer dans le code java, la création des classes et des méthodes. C'est à partir de ce moment-là que j'ai pris du plaisir à travailler puisque, même avec des problèmes, je réussissais à avancer avec l'aide de mon coéquipier qui m'a beaucoup aidé à comprendre tout ce qu'on faisait.

Je n'ai pas eu l'impression d'avoir beaucoup contribué puisque la majorité des pistes que je trouvais ne menait pas au résultat voulu et Dimitri a dû repasser sur mes codes presque tout le temps. Néanmoins on a été efficaces et ainsi on a pu être dans les temps malgré les autres projets et rattraper notre retard du début.

Dimitri Toupense

Même si au début, la SAE me semblait être vraiment compliquée parce qu'on n'y arrivait pas, elle a finalement été très intéressante, elle m'a beaucoup appris à la fois sur le java, le travail en équipe et l'utilisation de git. En effet, j'ai pu voir une évolution entre le début du projet où l'on travaillait sur la même branche et qu'on faisait un merge sur la branche de la semaine correspondante au lieu de directement travailler dessus et la fin du projet où chaque semaine je fixais les objectifs hebdomadaires et où je les mettait explicitement dans les rapports.

J'ajouterai que je me sens beaucoup plus confiant pour découvrir de moi-même de nouveaux packages grâce à ce projet, je comprends mieux comment marchent les packages, comment les exploiter et cela a surtout été utile pour maîtriser les bases de java.

Sinon, j'ai trouvé que la quantité de travail qui nous a été demandé chaque semaine convenait totalement avec nos disponibilités par rapport aux heures libres et le fait qu'il y ait de rendus hebdomadaires m'a vraiment aidé à garder un bon rythme de travail dont je devrais m'inspirer pour mes futurs projets.

Conclusion

Pour conclure, après ces six semaines de projet, nous avons pu rendre un API pouvant convertir un package Java vers un diagramme de classe Plant uml d'analyse ou de conception gérant les types d'éléments, les attributs ainsi que les méthodes. Tout au long du projet, nous avons respecté les dates de rendu hebdomadaires, entretenu notre dépôt git, produit différents diagrammes Plant uml pour nous aider à modéliser le travail à faire.