SAE P21 — Java2Puml

Table des matières

[Semaine 1 1](#_Toc134731254)

[Notes 1](#_Toc134731255)

[Digrammes de séquence Javadoc 2](#_Toc134731256)

[Digrammes de classe (analyse) Javadoc 2](#_Toc134731257)

[Digrammes de classe (conception) Javadoc 3](#_Toc134731258)

[Diagramme du package Western (généré) 3](#_Toc134731259)

[Semaine 2 3](#_Toc134731260)

[Notes 3](#_Toc134731261)

## Semaine 1

### Notes

Tout d’abord, une définition de l’API comme nous l’entendons : une API est l’acronyme de « Application programming interface » (interface de programmation d'applications) ; c’est un ensemble de ressources, de classes qui permettent à deux applications de communiquer entre elles. Les API sont donc un moyen d'extraire et traiter des données de plusieurs contextes radicalement différents.

Nous avons appris à appréhender l’api javadoc et ses bases, tel la récupération d’« éléments » — les Class, Interfaces et Enumérations — via l’environnement java (autrement dit la machine virtuelle qui lit, interprète, exécute le programme).

La commande javadoc de base, non pas pour l’utilisation de Doclet comme pour notre propre API mais pour la génération de fichier HTML, est d’ailleurs extrêmement pratique puisqu’elle nous permet d’obtenir rapidement et automatiquement un site de documentation pour notre code — en usant de la syntaxe « /\*\* comment \*/ » et les balises associées, tels « @param », « @return », etc avec lequel Samuel F est déjà familier.

Cette partie n’a pas été d’une très grande difficulté. Néanmoins, notre API ne contient à ce stade que très peu d’objet, aucune classe abstraite ni interface ; il s’agira de diversifier l’ensemble la semaine prochaine est donner l’allure d’une véritable API à notre projet !

### Digrammes de séquence Javadoc

Une image contenant capture d’écran, texte, diagramme, Plan

Description générée automatiquement

### Digrammes de classe (analyse) Javadoc

Une image contenant texte, capture d’écran, Logiciel multimédia, Logiciel de graphisme

Description générée automatiquement

### Digrammes de classe (conception) Javadoc

Une image contenant texte, capture d’écran, Logiciel de graphisme, Logiciel multimédia

Description générée automatiquement

### Diagramme du package Western (généré)

Une image contenant texte, capture d’écran, Logiciel multimédia

Description générée automatiquement

## Semaine 2

### Notes

L’amélioration de l’API avec l’ajout de nombreuses classes et interfaces a été long et quelque peu fastidieux (ayant assez mal organisé notre code), et le projet manque encore d’optimisation et de clarté. Néanmoins, les objectifs de la semaine 2 ont été atteints avec brio.

Les fonctions pour récupérer les détails des classes (visibilités des champs, des méthodes ainsi que le nom de ces dernières) et des interfaces (nom des méthodes) sont d’ailleurs déjà prêtes.

L’idée serait, à termes, qu’à chaque fois que l’utilisateur génère un schéma via notre PumlDoclet, ce dernier génère un DCA ET un DCC avec le nom précisé (Exemple : nomfichier\_DCA et nomfichier\_DCC) représentant le package souhaité.

### Digrammes de classe (conception) Java language model

Une image contenant texte, diagramme, ligne, capture d’écran

Description générée automatiquement

### Diagramme du package Western (généré)

Une image contenant texte, capture d’écran, affichage, Rectangle

Description générée automatiquement

### Commande pour créer **uniquement** un DCA

-private

-sourcepath

src

-doclet

pumlFromJava.doclets.PumlDoclet

-d

generates/pumls/

-g

dca

-out

ox.puml

western

L’option -g dca/dcc/both permet de générer un type particulier de schéma, ou les deux en même temps !

### Digrammes de notre API à cet instant

Une image contenant texte, capture d’écran, Plan, diagramme

Description générée automatiquement

## Semaine 3

### Notes

Les objectifs de la semaine ont été atteint dans les temps et la branche assignée a été merge.

Même si cela n’était pas demandé, nous avons chercher à créer les flèches « Use » dans notre schéma en recherchant les classes définies dans les paramètres des fonctions de chacune de nos classes/interfaces. Sans succès pour le moment. Cependant, notons qu’il faudra peut-être utiliser le type « ExecutableElement ».

### Digrammes de notre API à cet instant

Une image contenant texte, capture d’écran, diagramme, Plan

Description générée automatiquement

### Diagramme de classe (conception) du langage (amélioré)

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement

### Diagramme de classe (analyse) du package Western (généré)

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement