SAE 3.01 – Développement d'une application

GENERATION AUTOMATIQUE DE DIAGRAMME DE CLASSES

Description du sujet

Un diagramme de classe a pour objectif de représenter l'ensemble des classes d'un programme et les relations entre ces classes (héritage, sous-type, association). Il existe cependant peu d'applications simples d'accès qui permettent en quelques clics de générer un diagramme de classe à partir d'un code (car ce type d'application inclut de nombreuses autres fonctionnalités). Le plugin objectAid était un plugin très efficace et facile d'utilisation sous Eclipse, mais il n'est plus disponible et il n'existe pas sous d'autres IDE.

Ce projet a pour objectif de développer un logiciel permettant de générer et de manipuler des diagrammes de classe tout en restant simple d'utilisation et ergonomique.

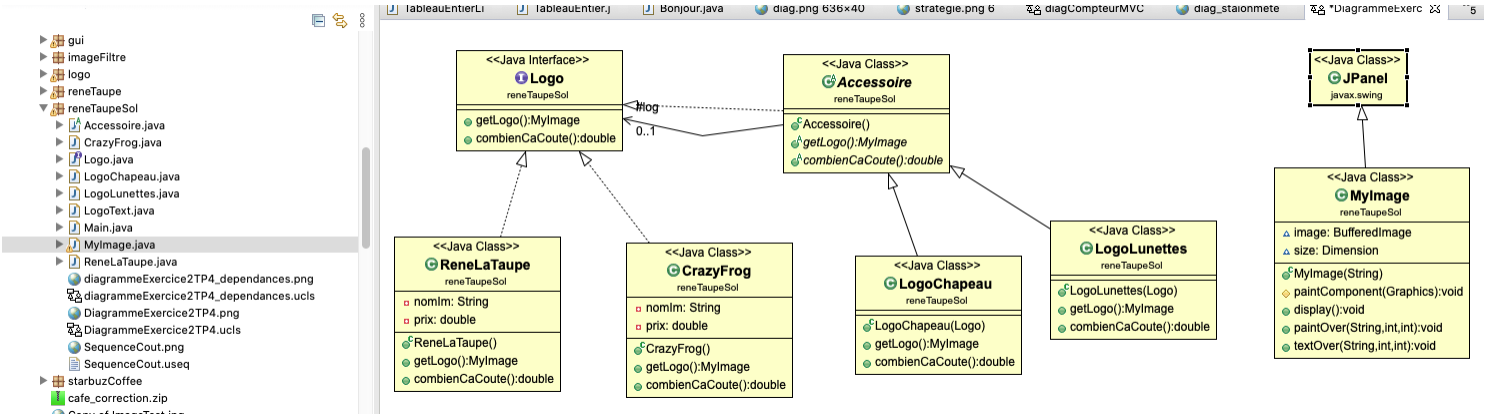

Figure 1: Exemple de diagramme généré avec objectAid

Objectifs à atteindre

L'application à développer doit pouvoir

- représenter les classes et les relations entre classes d'un package JAVA en utilisant l'introspection (la capacité qu'a une classe JAVA de connaître ses attributs, ses constructeurs et ses méthodes) ;
- générer à partir de cette représentation des diagrammes de classe à l'aide d'une interface graphique simple d'utilisation (déplacer les classes sur l'écran, afficher / masquer certaines classes ou méthodes, afficher / masquer la classe parent et les interfaces d'une classe, …) ;
- exporter les diagrammes de classe sous différents formats (image, sources plantUML, résultat d'une compilation plantUML) ;

- modifier le diagramme résultant en ajoutant de nouvelles classes ou des méthodes à des classes existantes ;
- générer les squelettes des classes construites au sein de cette application.

Le développement se fera en **JavaFx** et reposera sur une modélisation de ce qu'est un diagramme de classe, sur les patrons de conception vus dans les cours de l'IUT et sur l'introspection. En plus d'une interface graphique dédiée en javaFx, on pourra utiliser le package plantuml.jar (https://plantuml.com/fr/download) pour générer les diagrammes de classe et prévisualiser les diagrammes plantuml exportés.

Un **trello** devra être utilisé pour chaque groupe qui permettra de suivre la progression du travail et la répartition des tâches, ceci pour l'ensemble du projet.

Un dépôt **github** devra être créé pour chaque projet et les différents documents et codes sources devront y figurer.

Déroulement de la SAE

La SAE, d'une **durée de 64 heures** par étudiant, se déroulera sur 4 semaines : les semaines 49 (8h), 50 (16h) et 51 (16h) de 2024 ainsi que la semaine 2 (22h) de 2025. Les étudiants travailleront par groupes de 4 ou 3 (groupe de 19 = 4*4 + 1*3) qui auront été constitués dans le cours de QDev. Des explications sur l'application javaFx à réaliser seront données dans le dernier cours de QDev.
Des explications sur l'utilisation de Trello seront données dans le cours d'Analyse.

A. PARTIE ANALYSE

Semaine 49 et début 50 – 12h + 2h, dépôt d'un document d'étude préalable à la fin de la seconde séance de SAE de la semaine 50 et présentation de l'étude préalable aux enseignants d'analyse (avec éventuellement la présence de l'enseignant de Qualité de Développement du groupe) la séance suivante :

Chaque groupe fera **l'étude préalable du projet** et rédigera un **document** dans lequel figureront :
+ la liste des fonctionnalités,
+ les cas d'utilisation de l'application et les diagrammes de cas d'utilisation si cela est nécessaire, des descriptions textuelles, des DSS et/ou scénarios pourront les compléter,
+ un diagramme d'activités de l'application,
+ conception : un diagramme de classe en y précisant les patrons de conception que vous prévoyez d'utiliser
+ une maquette balsamique de l'application
+ le planning des itérations prévues et les objectifs de chacune (en termes de cas d'utilisations) avec identification des risques. Le trello du groupe devra être rempli selon le planning prévu.

Evaluation en analyse : soutenance de 10 minutes par groupe et évaluation du trello et du document d'étude préalable

B. Partie Qualite du Developpement

Semaine 50 et 1 – 6*8h + 2h. Développement de l'application en javaFx par itérations successives de 8h (6 itérations).

Evaluation en Qualité du Développement : selon le barème figurant dans le fichier *BaremeSAE_3.01* sur arche : la note obtenue tiendra compte des réunions effectuées à chaque itération (cf. fichier *reunion_iteration* sur arche), de la soutenance finale et d'une épreuve individuelle sur feuille qui se déroulera pendant **le vendredi 10 janvier**.

B.1. Documents attendus

Un répertoire ***Sources*** et un répertoire *Documents* devront figurer dans le dépôt git du projet. **Pour chaque itération**, un sous répertoire devra apparaître dans le répertoire ***Document****s* (par exemple *Documents/Iteration3* pour la troisième itération) qui comportera :

- Un fichier *fonctionnalitesCU.txt* décrivant les fonctionnalités et les cas d'utilisations développés ;
- un diagramme de classe et des diagrammes de séquence pour chaque fonctionnalité décrivant le fonctionnement de votre application, vous indiquerez aussi vos choix de conception, si des patrons sont mis en œuvre et si ces choix sont conformes à votre étude préalable ;
- un fichier bilan présentant un état de votre version (bug, validation, ...) ;
- un tag de version finalisant cette version.

Dans le répertoire ***Sources*** :

- Un code mis à jour qui fonctionne conformément à vos choix de conception ;
- un ensemble de classes de tests qui vérifient les critères de validation des différentes fonctionnalités ;
- pas de document inutile (.class, .project, fichiers temporaires).

De plus, pour chaque itération le trello devra être mis à jour et évoluer pendant l'itération. **Chaque itération de 8h de la SAE 3.01 devra fonctionner sur ce principe.**

B.2. Consignes

Fonctionnalités

- Choisissez vos fonctionnalités par priorité
- Ne faites que ce qui est utile pour les fonctionnalités de l'itération en cours
- N'ajoutez pas d'attributs ni de méthodes inutiles
- Ne pensez pas à l'itération suivante, vous y réfléchirez en temps utile
- Ne passez pas à de nouvelles fonctionnalités tant que celles en cours ne sont pas validées

Conception et code java

- Prenez le temps de réfléchir avant d'écrire du code
- Faites des classes simples (5 attributs max, limiter le nombre de lignes)
- Pas de copier-coller (réfléchir et changer la conception si besoin)
- Testez et faites des push régulièrement

B.3. CONTENU D'UNE ITERATION (8h)

(30 min) Choisir les fonctionnalités à ajouter
- Préciser ce qui doit être fait
- Adapter les critères de validation et les tests à réaliser

(1h30) Concevoir l'application
- Reprendre la conception de l'itération précédente
- Diagramme de classe, DSS ou scénarios des nouveautés
- Ajouter les éléments utiles
- Partage des taches à partir du diagramme de classes

(4h30) Réalisation
- Écriture du code et des tests

(1h30) Intégration et validation
- Validation des tests, débogage, mise en forme, tag