IFT2255 - Génie Logiciel

TP1

Date de remise : 16 novembre 2012
(dernière mise à jour: 9 novembre 2012)

Objectifs

Le travail à réaliser vise à explorer la structure et le comportement du logiciel libre Violet, un éditeur de diagrammes UML écrit en Java. Les résultats de cette exploration seront documentés à l'aide de diagrammes UML.

Conditions de réalisation

Le travail est à effectuer individuellement ou en groupe de deux personnes.

Obtention du code source

Une copie de Violet a été placée dans le dépôt du cours sur github (notez que ce dépôt n'est accessible qu'aux membres du compte UdeM-IFT2255). *Vous devez obligatoirement utiliser cette version afin d'assurer la cohérence des versions utilisées par toutes les équipes.* Les instructions pour accéder à ce dépôt sont disponibles sur la page web du cours.

Travail à réaliser

Considérez les activités suivantes :

1. Démarrage de l'application
2. Création d'un diagramme de classes
3. Ajout d'une nouvelle classe à un diagramme de classes existant
4. Renommer une classes existante
5. Ajouter une association entre deux classes existantes

Vous devez modéliser la structure et le comportement *existants* de Violet à l'aide d'UML. Vous devrez créer les diagrammes UML suivants à partir du code de l'application :

- **Diagramme de classes** : Définissez le diagramme de classes du système. Vous ne devez prendre en compte que les classes impliquées dans les activités décrites plus haut, ainsi que que les classes nécessaires à la réalisation et la cohérence du diagramme (ex: classes parentes, interfaces, etc.). Pour chaque classe, incluez ses attributs importants ainsi que ses méthodes publiques. Prenez soin d'inclure tous les classes parentes et les interfaces implémentées par chacune des classes représentées. Vous pouvez ignorer les classes et interfaces qui ne font pas partie de Violet (ex: java.util.*, etc.) dans vos diagrammes.

- **Diagramme d'états** : Définissez le diagramme d'états d'une instance de la classe `ClassNode`.

- **Diagramme de séquence** : Définissez un diagramme de séquence pour chacune des activités mentionnées plus haut. Représentez explicitement les actions de l'utilisateur.

Certaines classes peuvent être ignorées dans les diagrammes :

- les classes correspondant au "*Splash Screen*" lors du chargement de l'application (attention: "*Welcome Panel*" correspond à la page d'accueil et doit apparaître dans les diagrammes)
- les classes reliées l'injection de dépendances (ex: BeanFactory, ResourceBundleInjector, etc.)
- les classes qui ne font pas partie du paquetage com.horstmann.violet (ex: com.pagosoft.*, etc.)

Aide et discussions

Vous êtes encouragés à discuter du projet et à poser vos questions en utilisant le forum créé à cette fin sur StudiUM.

Remise

Le travail doit être remis électroniquement sur StudiUM au plus tard vendredi 16 novembre **à minuit**. Vous devrez remettre une archive zip ou tar.gz contenant tous les diagrammes, ainsi qu'un fichier texte indiquant le nom de tous les membres de l'équipe ayant contribué à la réalisation du travail. Une seule remise remise électronique est nécessaire par équipe.

Vous devez utiliser Violet pour créer les diagrammes. Pour faciliter la correction, vous devez nommer vos fichiers de la façon suivante :

Diagramme	Nom de fichier
Diagramme de classes	`Violet.class.violet`
Diagramme d'état	`ClassNode.state.violet`
Diagrammes de séquence	
Démarrage de l'application	`1-demarrage.seq.violet`
Création d'un diagramme de classes	`2-diagClasses.seq.violet`
Ajout d'une nouvelle classe à un diagramme de classes existant	`3-ajoutClasse.seq.violet`
Renommer une classes existante	`4-renommerClasse.seq.violet`
Ajouter une association entre deux classes existantes	`5-assoc.seq.violet`

Barème

Ce travail compte pour 10% de la note finale, divisé comme suit :

4 pts Diagramme de classes
1 pts Diagramme d'états
5 pts Diagrammes de séquences (5)