

Université Paul Sabatier – Toulouse III
IUT A - Toulouse Rangueil
Projet tuteuré #20

Antoine de ROQUEMAUREL
Mathieu SOUM
Geoffroy SUBIAS
Marie-Ly TANG
Groupe B

Pour Monsieur Max CHEVALIER (Responsable projets)
Monsieur Thierry MILLAN (Client)
Madame Caroline KROSS (Tutrice)

Analyse et conception

Bibliothèque d'objets graphiques UML

Table des matières

| | | |
|----------|---|----------|
| 1 | Analyse | 3 |
| 2 | Conception | 4 |
| 2.1 | Architecture générale du projet | 4 |
| 2.1.1 | Package eltGraphique | 5 |
| | Package eltModelisation | 5 |
| | Package ligne | 5 |
| 2.1.2 | Package diagramme | 5 |

Chapitre 1

Analyse

Chapitre 2

Conception

2.1 Architecture générale du projet

L'architecture est la base d'une conception telle que nous l'avons choisie. En utilisant la notation UML, nous sommes parvenu à élaborer un "meta-modèle" de cette notation mais en isolant uniquement l'aspect graphique, faisant ainsi abstraction des multiples règles de conception que la norme UML impose. Après discussion avec le client, de nombreuses modifications ont été apportées à l'architecture d'origine pour aboutir à une solution stable. Cette nouvelle architecture se compose de vingt classes, réparties en quatre paquetages comme suit :

- **eltGraphique**
 - *ElementGraphique*
- **eltModelisation**
 - *Acteur*
 - ActeurActif
 - ActeurPassif
 - Attribut
 - CasUtilisation
 - Classe
 - *ElementModelisation*
 - Interface
 - Methode
 - Traitement
 - Visibilite
 - Variable
- **ligne**
 - Cardinalite
 - Lien
 - MessageTraitement
 - TypeLien
- **diagramme**
 - Diagramme
 - DiagrammeCasUtilisation
 - DiagrammeClasse
 - DiagrammeSequence

Dans cet arbre représentant notre architecture, on peut voir certains noms **en gras** ou *en italique*. Les noms **en gras** représentent les différents paquetages que nous avons séparés. Ceux *en italiques* sont des classes abstraites créés afin de factoriser le code dans l'optique de réaliser une programmation objet optimale et de suivre les objectifs de propreté du code imposés par le client.

Dans un premier temps, nous avons séparé les diagrammes des éléments graphiques. En effet, un diagramme sera composé de toute sorte d'éléments graphiques. Puis nous avons découpé ces derniers en deux, isolant ainsi les lignes des éléments de modélisation tels que les classes, les traitements ou les cas d'utilisation.

ElementGraphique et *ElementModelisation* sont des classes abstraites car elle regroupe les fonctions communes à toutes les classes de leur paquetage respectifs sans pour autant en fournir une implémentation de chacune d'elles – comme par exemple la méthode qui crée la représentation graphique d'un élément ou celle qui permet de le supprimer d'un diagramme.

Prenons maintenant chaque paquetage séparément

2.1.1 Package `eltGraphique`

Le paquetage `eltGraphique` regroupe toutes les classes qui représentent des éléments graphiques. Il regroupe la classe *ElementGraphique* et deux paquetages **`eltModelisation`** et **`ligne`**. Cette classe possède deux attributs `graph` et `diagramme`, correspondant respectivement au graphe dans lequel sont stockés les éléments et le diagramme afficher à l'écran. Elle comprend également (en plus d'un constructeur initialisant les attributs) les méthodes `supprimer` permettant de supprimer un élément du graphe et du diagramme, ainsi que `creer`, méthode abstraite réimplémentée dans les classes descendantes servant à créer la représentation graphique de l'objet et l'afficher sur le diagramme.

Package `eltModelisation` Ce paquetage regroupe toutes les classes représentant les différents éléments de modélisation que l'on trouve dans les diagrammes UML de cas d'utilisation, classe et de séquence. *Acteur* est une classe abstraite car les acteurs actifs et passifs ont beaucoup de caractéristiques identiques mais n'ont pas la même représentation sur un diagramme. Les classes *Visibilite*, *Methode* et *Attribut* permettent au client de créer facilement une interface de saisie de ces éléments, facilitant l'usage du logiciel final. De plus, l'ajout de ces méthodes et attributs dans des classes, des acteurs, des traitements ou des interfaces pourra faciliter l'ajout futur de nouvelles fonctionnalités comme par exemple de la génération de code Java.

Package `ligne` Ce paquetage regroupe peu de classes. *TypeLien* est une classe énumérée servant à recenser tous les types graphiques de liens existant dans la notation UML. *Cardinalite* quant à elle, permet comme *Methode* ou *Attribut*, d'aider le client à permettre une saisie facilitée des cardinalités d'un lien dans un diagramme de classe par exemple. La classe *Lien* comprend la méthode `creer` qui va configurer tous les styles de liens et applique au nouveau lien le style choisi par l'utilisateur. *MessageTraitement* spécialise *Lien*, permettant de créer un style de lien particulier aux messages entre traitements dans les diagrammes de séquences.

2.1.2 Package `diagramme`

Le paquetage **`diagramme`** comprend plusieurs types de diagramme prédéfinis qui sont cas d'utilisation, classe et séquence. Ils descendent de la classe *Diagramme*. Chaque type de diagramme implémente deux méthodes `eltAutorise` et `lienAutorise` qui permettent respectivement d'autoriser ou interdire un type d'élément particulier et un type de lien entre deux types d'éléments particuliers. La classe *Diagramme* est générique. Par défaut elle autorise tous les éléments et tous les liens. Il est donc possible au client de réimplémenter ses propres méthodes pour créer ses propres règles.