0x2 - Organisation du code

Partie recherche de dépendances :

La classe principale du projet est la classe Puck2Main.java. C'est la classe qui gère les points d'entrée vers les différentes fonctionnalités du logiciel (voir rapport).

La classe ConfigurationUI.java s'occupe de l'affichage graphique du programme (quand lancer dans ce mode là).

Le package graph contient toutes les classes qui permettent la mise sous forme de graphe de dépendance (sous forme XML), le programme. Il contient donc des classes permettant de représenter les différents nœuds (Node.java) et les relations entre ces derniers (Edge.java) et notamment la classe permettant de générer le graphe XML (XMLExporter.java) utilisé par la partie graphique.

La classe graph.readers contient des classes appelées « readers » qui servent à trouver les potentielles dépendances des différents types pour lesquelles elles ont étés crées.

Par exemple la classe MethodReader.java permet d'analyser le type de données retourné ou les types des données passées en paramètres des méthodes afin de trouver les dépendances inter-classes.

D'autres classes éponymes permettent le refactoring (rename pour l'instant) mais nous n'avons pas touché à ces parties. Le reste des packages contiennent les classes des bibliothèques utilisées comme celle de ExtendJ.

Partie graphique :

Cette partie s'occupe de la visualisation graphique du graphe de dépendances ainsi que de l'interaction avec ce dernier. Le projet que nous avons repris comporte toute une partie sur le refactoring rename, que nous n'avons pas traité (package com.puck.refactoring), une partie sur la possibilité de revenir en arrière et refaire une action (undo/redo) que nous n'avons pas non plus traitée (package com.puck.undoRedo), et enfin une partie chargée de reproduire sous forme d'un graphe XML le graphique à laquelle nous ne nous sommes pas non plus intéressé.

La classe « *Genration2ToDisplayMain.java* » est la classe principale du programme. Cette dernière se charge de toute la partie graphique créée à partir de JavaFx (les fenêtres, boutons, etc …, mais pas de la représentation du graphe).

La classe XmlToStrucuture.java contenue dans le package com.puck.utilities.piccolo2d) est chargée de récupérer le graphe XML, d'y extraire les données nécessaires à la génération du graphique. Elle crée les différents nœuds ou nodes et récupère les relations (edges) entre ces nœuds.

La classe NodeToPnodeParser.java s'occupe de transformer en éléments graphique les nœuds et les edges. En effet, elle fait appel aux classes du package com.puck.arrows. Ces classes qui s'appuie sur la bibliothèque Piccolo2D, servent à représenter graphiquement les dépendances (sous forme de flèches) et les nodes (sous forme de boîtes imbriquées). Ce même package contient la classe ArrowNodesHolder.java qui stocke les différentes flèches et gère leur affichage.

Viens ensuite la classe NewDisplayDG.java (com.puck.display.piccolo2d) qui permet l'affichage du graphe et de le gérer.

Il est possible de cacher des flèches ou des nœuds. Ces possibilités sont regroupées dans un menu accessible en faisant un clic droit sur un objet. Les classes gérant ce menu sont contenues dans les packages com.puck.menu et com.puck.menu.items.

Enfin la classe PNode est la classe grâce à laquelle nous pouvons représenter les différents nœuds. Pour ajouter certaines fonctionnalités et attributs (texte, icône …) aux nœuds, la classe PiccoloCustomNode.java a été créée.

Enfin la classe PCustomInputEventHandler.java (com.puck.utilities.piccolo2d) définie quelques actions qu'il est possible d'effectuer sur les PiccoloCustomNode.

Le package « com.puck.arrows » contient toutes les classes responsables de la représentation des flèches ...