

# Rapport TP2 évolution et rustructutation

## Explication code Exo1 et 2

La classe principal de l'application est la classe Parser qui permet de créer l'AST de l'application et de récupérer les méthodes et leur Invocations.

J'ai fait pour le TP un graphe de couplage Unidirectionnel.  
Il y a un arc de A->B si la classe A appelle une méthode de la classe B.

Pour calculer le couplage entre 2 classes la moyenne entre l'arc A->B et B->A ce qui me donne le couplage entre 2 classes.

Le couplage unidirectionnel me semblait meilleur que le bidirectionnel car cela permet de bien voir si c'est surtout la classe A qui appelle les méthodes de la classe B ou l'inverse. Alors que le bidirectionnel avait comme défaut de pas mettre en évidence le sens des appels de méthodes. Cette méthode du graph bidirectionnelle me paraissait donc plus cohérente.

Le programme dessine le graph d'appel sous forme de PNG.

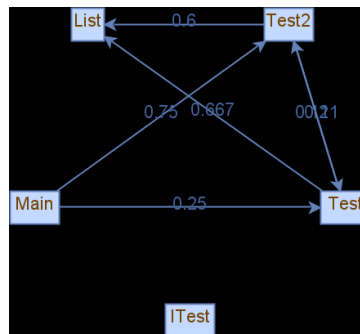


FIGURE 1 – Example représentation graph

il dessine aussi un graphe représentant le regroupement (clustering) hiérarchique à une étape  $i$  de l'algorithme passé en paramètre. Les clusters sont représentés sous forme de noeud qui a comme nom l'ensemble des noms des classes que le cluster contient. Par exemple si un cluster contient une classe A B et C le nom du cluster sera A :B :C.

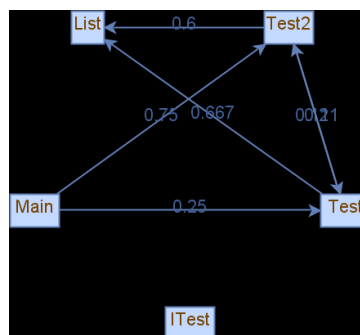


FIGURE 2 – Example représentation de l'algorithme Cluster a 0 étape

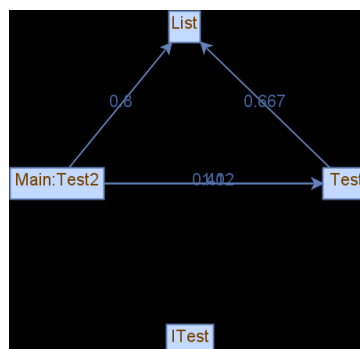


FIGURE 3 – Example représentation de l'algorithme Cluster a 1 étape

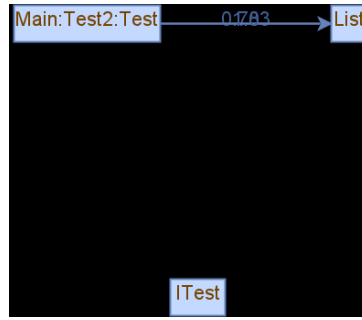


FIGURE 4 – Exemple représentation de l’algorithme Cluster a 2 étape



FIGURE 5 – Exemple représentation de l’algorithme Cluster a 3 étape

il dessine aussi le dendogramme sous forme de graph (j’ai utilisé la libraiiri JgraphT pour dessiner le dendogramme la représentation est donc pas parfaite comme sur l’exemple du TP de plus l’image ne ce dessine pas sur des gros projet car la taille de l’image sera trop grosse a dessiner pour JgraphT (dépassement mémoire) l’erreur ce produit sur l’exemple de projet test).

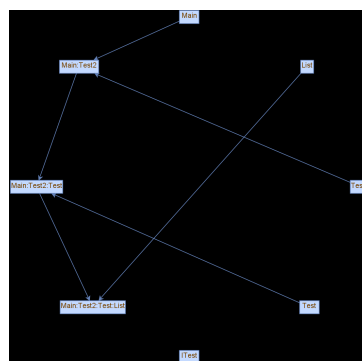


FIGURE 6 – Exemple de dendogramme

Je dessine de plus un graph représentant l'algorithme d'indentification des groupes de classes couplées avec une valeur de CP passé en paramètre.

cette fonction parcourt récursivement l'arborescence d'un dossier et retourne tous les liens des fichiers java de ce dossier.

```
1 public ArrayList<String> Scan(String Path)
```

Retourne dans un String le contenu d'un fichier passé en paramètre

```
1 public String readFileToString(String filePath)
```

Intialise CompilationUnit en fonction d'un fichier java passé en paramètre

```
1 public CompilationUnit AST(String Path)
```

Cette fonction permet en fonction d'un arraylist de lien de fichier java (qui sera créé par la fonction Scan(String Path)) et qui retourne le graphe d'appel avec les différents couplage sur chaque arret.

```
1 public DirectedWeightedPseudograph Parse(ArrayList<String> f)
```

Cette fonction permet de créer une copie d'un graph.

```
1 public DirectedWeightedPseudograph copygraph(  
    DirectedWeightedPseudograph<String , MyWeightedEdge> graph)
```

Cette fonction retourne un couple de noeud avec le plus grand couplage d'un graph passé en paramètre.

```
1 public String [] ClusterProche(DirectedWeightedPseudograph<String ,  
    MyWeightedEdge> graph)
```

Cette fonction prend un graph et deux noeud et va retourner un nouveau graphe avec un noeud contenant les 2 noeuds passé en paramètre.

```
1 DirectedWeightedPseudograph CreateCluster(DirectedWeightedPseudograph graph
    , String[] ex)
```

Cette fonction prend un graph et un entier i et va retourner un graph avec les différents cluster a une étape i de l'algorithme en utilisant les fonctions écrites précédemment.

```
1 DirectedWeightedPseudograph HierarchieCluster(DirectedWeightedPseudograph
    hierarchie , int i)
```

Cette fonction permet de créer un nouveau noeud contenant a partir du noeud cible symbolisant un cluster et lui enlève la class "cible" et retourne le nouveau noeud (cluster) sans la classe "cible"

```
1 String newNode(String remove,String replace)
```

Cette fonction utilise la méthode précédente pour intégrer le nouveau noeud dans le graph et supprimer le noeud "cible"

```
1 DirectedWeightedPseudograph replaceVertex(String remove,String replace ,
    DirectedWeightedPseudograph module)
```

Cette fonction retourne le couplage moyen d'un module en fonction du nom du module et du graph de couplage passé en paramètre.

```
1 float moyennecouplage(String module ,DirectedWeightedPseudograph graph)
```

Cette fonction retourne le noeud d'un cluster qui est le moins couplé en fonction du nom du cluster et du graph de couplage.

```
1 String VertexMustBig(String module ,DirectedWeightedPseudograph graph)
```

Cette fonction répond à la question 2 de l'exo2 et crée différents modules en respectant que le nombre de modules ne dépasse pas  $M/2$  ou M est le nombre de classes. Que chaque classe de chaque module fasse partie de la même branche du dendrogramme. Et que le nombre moyen de couplage par module soit supérieur à CP.

```

1 DirectedWeightedPseudograph GroupeClasse(DirectedWeightedPseudograph
    hierarchie , float CP)

```

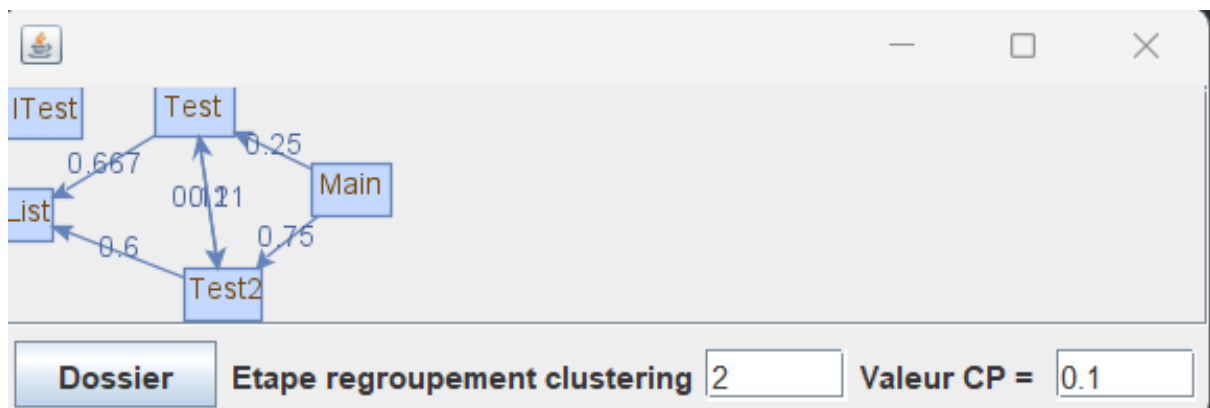
Cette fonction permet de créer le dendrogramme a partir d'un graphe d'appel donné en paramètre

```

1 SimpleDirectedGraph DrawDendrogramme(DirectedWeightedPseudograph hierarchie
    )

```

## Interface



En cliquant sur le boutons "Dossier" s'ouvre une interface pour selectionner le projet à parcourir recursivement.

le champs "Etape regroupement clustering" on indique a qu'elle étape de l'itération on veut arreter l'algorithme de regroupement hiérarchique et le résultats sera ensuite dessiner sous forme d'un PNG dans le dossier "image". le champ de "Valeur CP" indique qu'elle moyenne minimal de couplage doit avoir chaque cluster. le resultats sera ensuite dessiner sous forme de PNG dans le dossier "image".

## Exo 3

L'exo 3 contient les mêmes classe que l'Exo1 et 2 sauf que pour la fonction Parser on utilise les fonctions fournis par Spoon.