Rapport TP1 évolution et rustructutation

Explication code Exo1

Pour pouvoir récuperer les classes, méthodes et package de l'AST j'utilise des classes avec le design pattern visitor pour récupérer chaque information.

- ClassVisitor
  ce visitor permet toutes les classes et interface de l'AST.
- MethodeVisitor
  ce visitor permet toutes les méthode de l'AST.
- packageVisitor
  ce visitor permet tout les packages de l'AST.

Le premier exercice j'ai créé plusieurs classe qui me permet de stocker les informations dont j'ai besoin.

- JavaFichier
  Cette représente un un fichier java elle stock toutes les classes contenu dans ce fichier et aussi le nombres de lignes du fichier.
- javaClass
  Cette classe représente une classe en java. Cette classe contient le noms de la classe, la liste des packages a qui appartient cette classe, la liste de methodes et d'attribut que contient la classe.
- javaAttribut
  Cette classe représente un attribut d'une classe elle contient le nom de l'attribut.
- JavaMethode
  Cette classe représente une méthode java. Elle contient le nom de la méthode, le nombre de ligne qu'elle fait et le nombre de paramètre qu'elle contient.

Parser

L'application contient une classe Parser qui permet à l'aide du module CompilationUnit d'initialiser les classes présentés au-dessus en parcourant toute l'arborescence.
la fonction :

```
1            public String readFileToString(String filePath)
2
```

Permet à partir d'une adresse d'un fichier de lire le contenu de ce fichier et retourne son contenu en String.

```
1  public ArrayList<String> Scan(String Path)
2
```

Cette méthode prend en paramètre un lien vers un dossier et parcourt récursivement l'arborescence de ce dossier. La fonction rend une ArrayList de String contenant tout les liens de chaque fichier java à analyser.

```
1 public CompilationUnit AST(String Path)
2
```

Cette méthode utilise la fonction readFileToString pour instancier le module CompilationUnit pour créer l'AST et retourne cette AST.

```
1 public void AnalyseAST(String path)
2
```

Cette méthode utilise les deux fonctions précédentes pour récupérer le texte de chaque fichier java d'une arborescence et instancier un CompilationUnit pour chaque fichier de l'arborescence. Avec CompilationUnit remplit l'ArrayList de JavaFichier qui va pourvoir stocker tous les fichiers, classe, méthode, etc...

La classe contient ensuite toutes les méthodes repondant au question posé sur le TP.

```
1 public int nbClasses()
```

Retourne le nombre de classe du projet.

```
1 public int nbLigneApplication()
```

Retourne le nombre total de ligne de l'application.

```
1     public float nbMoyenLigneMethode()
```

Retourne le nombre moyen de ligne des méthodes du projet

```
1  public ArrayList<JavaMethode> AllMethode()
```

Retourne toutes les méthodes du projet.

```
1 public ArrayList<JavaMethode> methode10ligne()
```

retourne les 10% des méthodes du projet qui contiennent le plus de ligne.

```
1 public ArrayList<JavaClass> Classe10methode()
```

Retourne les 10% des classes qui contiennent le plus de méthodes.

```
1 public int nbMethode()
```

Retourne le nombre de méthode de l'application.

```
1 public int ParametreMax()
```

cette méthode retourne le nombre de paramètre maximal de paramètre que contient une méthode dans tout le projet.

```
1 public int nbPackage()
```

Retourne le nombre de package du projet.

```
1 public float nbMoyenMetode()
```

Retourne le nombre moyen de méthode par classe dans tout le projet.

```
1 public ArrayList<JavaClass> Classe10Attribut()
```

Retourne les 10% de classes qui contiennetn le plus d'attribut dans tous le projet.

```
1     public ArrayList<JavaClass> Classe10Attribut10methode()
```

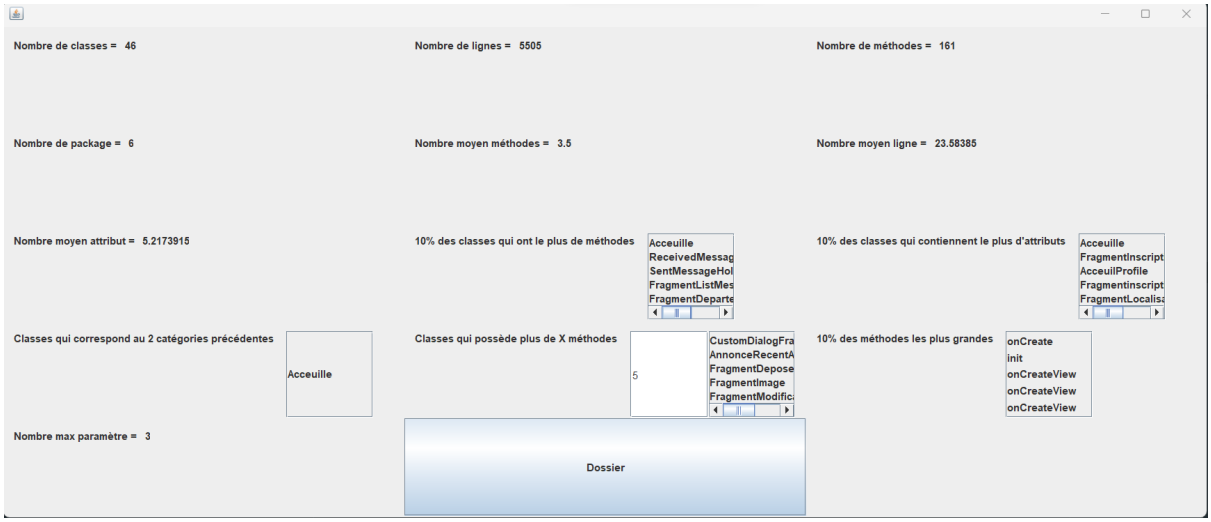
Retourne l'intersection entre les reponses de la méthode Classe10Attribut() et Classe10methode().

```
1  public ArrayList<JavaClass> Xmethodes(int X)
```

Retourne les méthodes qui contiennet au moins x paramètre. X étant passé en paramètre de la fonction.

Interface

L'application contient une interface graphique qui permet de visualiser le résultat de l'AST et de choisir un projet à analyser.



Explication code Exo2

La classe principal de l'application est la classe Parser qui permet de créer l'AST de l'application et re récupérer les Méthode et leur Invocation.

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

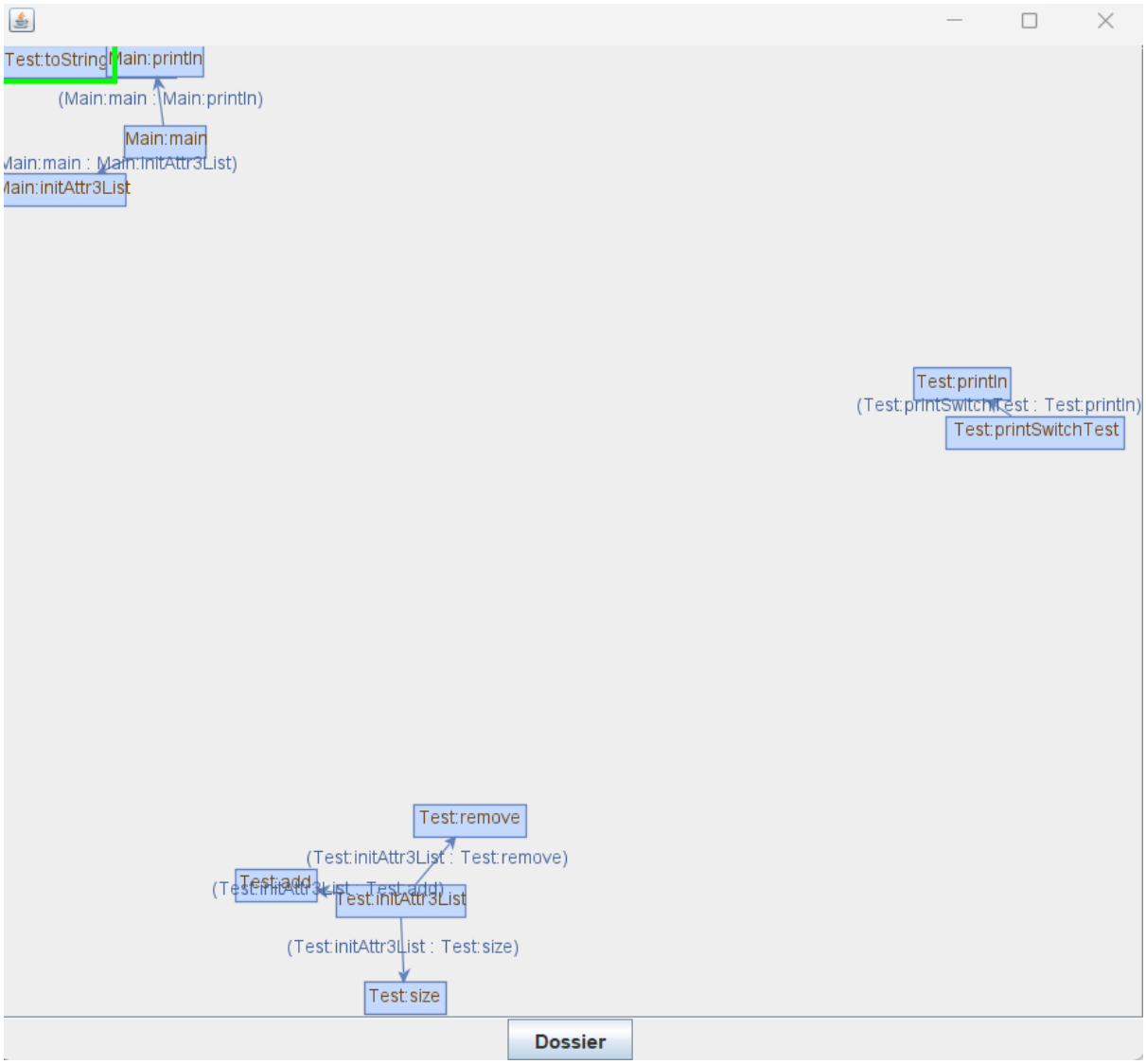
```
1   public CompilationUnit AST(String Path)
```

Constructeur qui initialise le graph en fonction d'une arborescence passé en paramètre.

```
1 public Parser(String path)
```

Interface

L'application à une interface graphique qui permet de représenter le graph.



De plus l'application créer un PNG du graph d'appel.