README — TP1 / Partie 2 : Métriques & Graphe d'appel (JDT)

Projet : `Partie2TPEvo`

Réalisé par : Ouezzani Rahma

Avant de commencer (dézipper l'archive)

1. Décompressez l'archive `Partie2TPEvo.zip` pour obtenir le dossier `Partie2TPEvo/`.
2. Vérifiez la présence de `pom.xml`, `src/`, `target/` (optionnel), `README.tex`, etc.
3. (Optionnel) En ligne de commande :

```
unzip Partie2TPEvo.zip -d .
```

Structure du projet

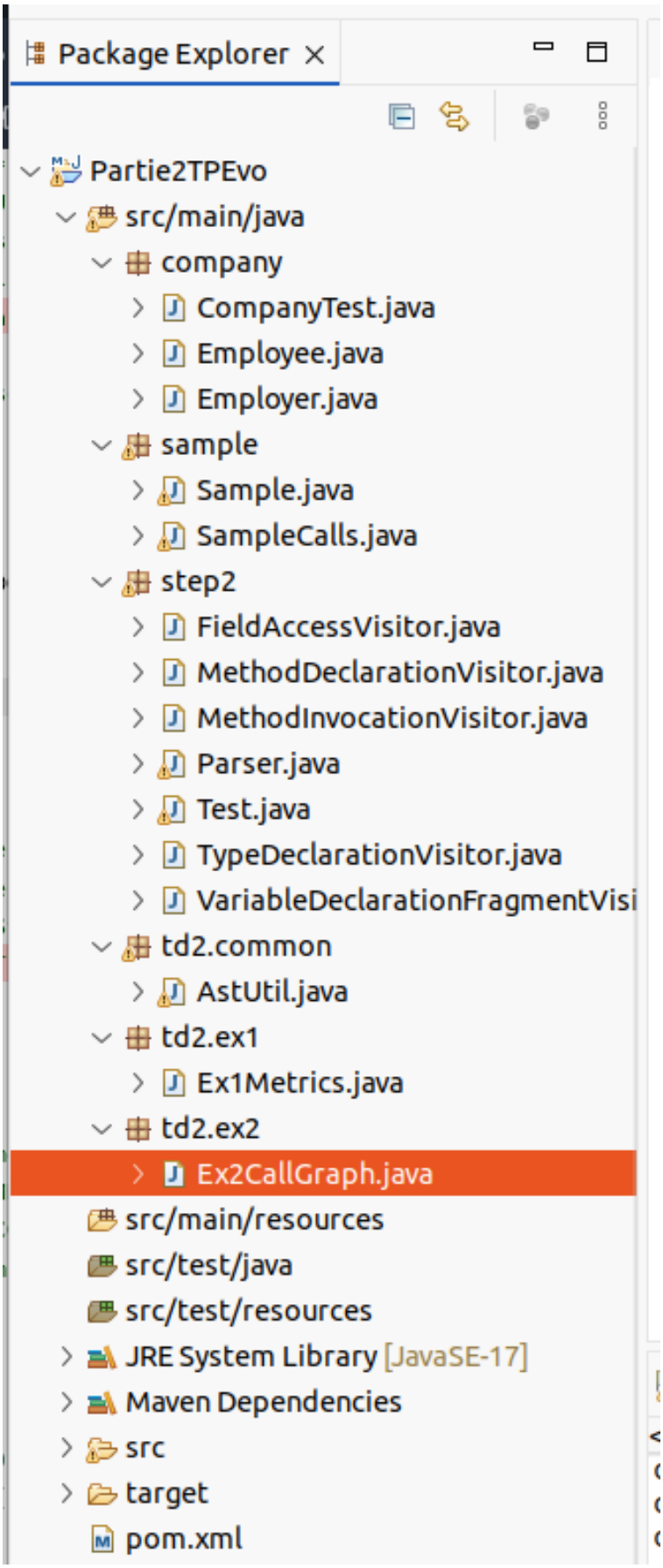

FIGURE 1 – Structure du projet (Eclipse — Package Explorer).

Prérequis

— **JDK 17** (avec `JAVA_HOME` configuré).
— **Eclipse** (projet Maven importé, JRE `JavaSE-17`).
— (Optionnel) **GraphViz** pour exporter le graphe en PNG :

```
sudo apt update && sudo apt install -y graphviz
```

Versions utilisées

— **JDK :** 17
— **Eclipse :** 202x-xx (compat. JDT Core)
— **Maven :** 3.x
— **GraphViz :** 2.42 (ou équivalent)

Import / Build (Eclipse)

— **Eclipse** → **File** → **Import** → **Existing Maven Project** → sélectionner le dossier `Partie2TPEvo/` → **Finish**.

Lancer dans Eclipse

Exercice 1 — Métriques

1. **Run** → **Run Configurations...** → **Java Application** → **Ex1Metrics**
2. Onglet **Arguments** → **Program arguments** :

   ```
   --src=src/main/java/sample 4
   ```

3. (Optionnel) Ajouter `-gui` pour ouvrir la fenêtre.
4. **Run**

Sorties (dans `target/reports/`) :
— `metrics-classes.csv`
— `metrics-top-long-per-class.csv`
— `metrics-top-long-global.csv`

Exercice 2 — Graphe d'appel

1. **Run** → **Run Configurations...** → **Java Application** → **Ex2CallGraph**
2. Onglet **Arguments** → **Program arguments** :

   ```
   --src=src/main/java/sample
   ```

3. (Optionnel) Ajouter `-gui` pour ouvrir la fenêtre.
4. **Run**

Sorties (dans `target/reports/`) :
— `callgraph.dot`
— `callgraph-edges.csv`
Export PNG (optionnel) :

```
dot -Tpng target/reports/callgraph.dot -o target/reports/callgraph.png
```

Exécuter en terminal (optionnel)

Depuis la racine du projet `Partie2TPEvo/` :

```
# M triques (Q1.1)
mvn -q exec:java -Dexec.mainClass=td2.ex1.Ex1Metrics \
  -Dexec.args="--src=src/main/java/sample 4"

# Graphe d'appel (Q2.1)
mvn -q exec:java -Dexec.mainClass=td2.ex2.Ex2CallGraph \
  -Dexec.args="--src=src/main/java/sample"

# (optionnel) Image du graphe si GraphViz est install
dot -Tpng target/reports/callgraph.dot -o target/reports/callgraph.png
```

Projet métier complémentaire — `company`

Une extension du TP a été réalisée avec un **projet métier** intitulé `company`, afin d'appliquer l'analyse sur un code Java plus concret.

Structure

Le package `company` contient :
- `Employee.java` : représente un employé avec un nom et un salaire.
- `Employer.java` : gère les employés (embauche, affichage, travail collectif).
- `CompanyTest.java` : classe principale permettant de tester le système.

Exécution

Pour générer le graphe d'appel du projet métier :

```
mvn -q exec:java -Dexec.mainClass=td2.ex2.Ex2CallGraph \
  -Dexec.args="--src=src/main/java/company"
```

Sorties (dans `target/reports/`) :
- `callgraph-company.dot`
- `callgraph-company.csv`
- `callgraph-company.png`

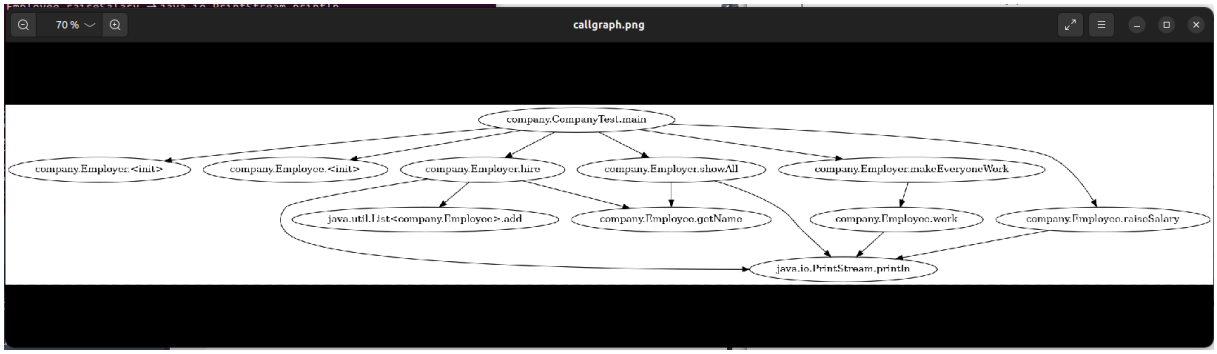

FIGURE 2 – Graphe d'appel généré pour le projet métier `company`.

Ce graphe illustre les appels entre `CompanyTest`, `Employer` et `Employee`, confirmant la bonne réutilisation de l'outil d'analyse JDT sur un projet applicatif réel.

Notes

— `-src` : dossier racine des sources (défaut : `src/main/java/sample`).
— Dans l'exercice 1, le dernier entier est le **seuil X** pour « classes avec > X méthodes ».

Liens du projet

Lien GitHub :
https://github.com/Rahma121-crtl/TP1-Partie2-EvolutionLogiciels-Final

Lien de la vidéo de démonstration (Google Drive) :
https://drive.google.com/file/d/1sgfY0oa3NMkgYLukfRZNwftJbrZNI6BJ/view?usp=sharing