Contents

Début du rapport

SAE S2.02 – Rapport pour la ressource Dev

SERE Benjamin LEGRAND Alexandre POUPARD–RAMAUT Rémi

Version 1 :

Commande pour lancer Application.jar

```
java -jar Application.jar
```

Architecture permettant l'éxecution du projet en dehors d'Application.jar

```
klonk@ordidefou:~/Documents/cours/F5$ tree
.
├── Application.jar
├── bin
│   ├── app
│   │   ├── Main.class
│   │   ├── Plateforme.class
│   │   └── Voyageur.class
│   └── graphes
│       ├── MonLieu.class
│       ├── Troncon.class
│       └── TypeCout.class
├── dev
│   ├── rapport_dev.md
│   └── UML
├── lib
```

```
│   ├── jgrapht-core-1.5.1.jar
│   ├── jheaps-0.14.jar
│   ├── junit-platform-console-standalone-1.10.2.jar
│   └── sae_s2_2024.jar
├── src
│   ├── app
│   │   ├── Main.java
│   │   ├── Plateforme.java
│   │   └── Voyageur.java
│   └── graphes
│       ├── MonLieu.java
│       ├── Troncon.java
│       └── TypeCout.java
└── test
    └── app
        └── TestPlateforme.java
```

Diagramme UML

Diagramme UML en mermaid.js

```
classDiagram
    class Plateforme {
        - trancons : HashMap<Trancon, Double>
        - lieux : HashSet<Lieu>
        - graphePrix : MultiGrapheOrienteValue
        - grapheCO2 : MultiGrapheOrienteValue
        - grapheTemps : MultiGrapheOrienteValue
        - graphe : MultiGrapheOrienteValue
        - voyageur : Voyageur
        - m : ModaliteTransport
        - coutChoisi : TypeCout

        + Plateforme(String[] data)
        + choisirFiltre() void
        + ventilation(String[] data) ArrayList<String[]>
        + extraireLieu(ArrayList<String[]> data) HashSet<Lieu>
        + extraireSommet(HashSet<Lieu> lieu, MultiGrapheOrienteValue graphe) voi
        + extraireTroncon(ArrayList<String[]> data, TypeCout cout) HashMap<Tranc
        + triData(ArrayList<String[]> data) void
        + getTypeCout(int i, ArrayList<String[]> data) Double
        + _getUserInput() String
        + extraireArete(HashMap<Trancon, Double> hashMapTrancon, MultiGrapheOrie
        + genererGraphe(ArrayList<String[]> data) void
        + clearMap() void
```

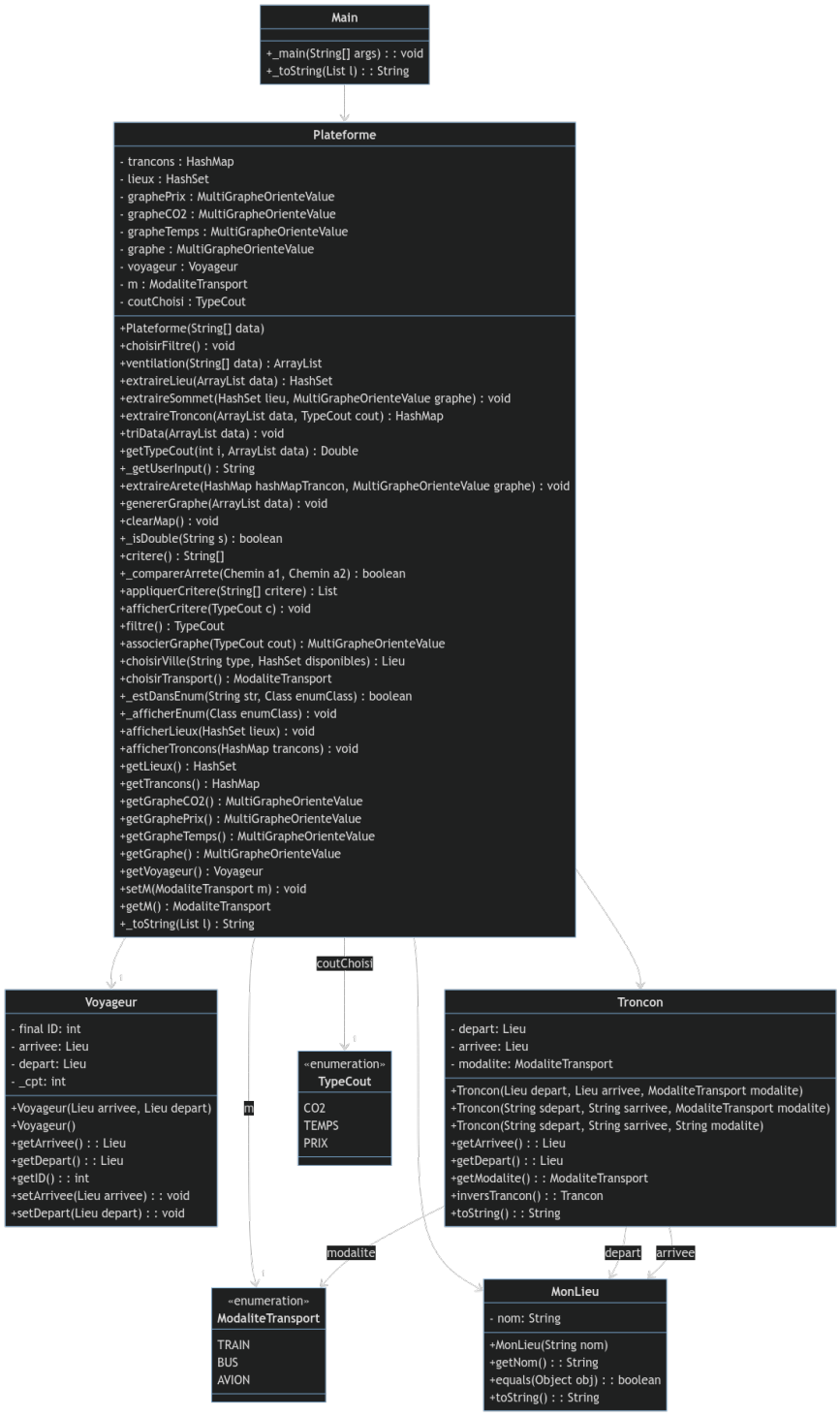

Figure 1: Image de l'UML

```
    + _isDouble(String s) boolean
    + critere() String[]
    + _comparerArrete(Chemin a1, Chemin a2) boolean
    + appliquerCritere(String[] critere) List<Chemin>
    + afficherCritere(TypeCout c) void
    + filtre() TypeCout
    + associerGraphe(TypeCout cout) MultiGrapheOrienteValue
    + choisirVille(String type, HashSet<Lieu> disponibles) Lieu
    + choisirTransport() ModaliteTransport
    + _estDansEnum(String str, Class<T> enumClass) boolean
    + _afficherEnum(Class<T> enumClass) void
    + afficherLieux(HashSet<Lieu> lieux) void
    + afficherTroncons(HashMap<Trancon, Double> trancons) void
    + getLieux() HashSet<Lieu>
    + getTrancons() HashMap<Trancon, Double>
    + getGrapheCO2() MultiGrapheOrienteValue
    + getGraphePrix() MultiGrapheOrienteValue
    + getGrapheTemps() MultiGrapheOrienteValue
    + getGraphe() MultiGrapheOrienteValue
    + getVoyageur() Voyageur
    + setM(ModaliteTransport m) void
    + getM() ModaliteTransport
    + _toString(List<Chemin> l) String
}

class Voyageur {
    - final ID: int
    - arrivee: Lieu
    - depart: Lieu
    - _cpt: int
    + Voyageur(Lieu arrivee, Lieu depart)
    + Voyageur()
    + getArrivee(): Lieu
    + getDepart(): Lieu
    + getID(): int
    + setArrivee(Lieu arrivee): void
    + setDepart(Lieu depart): void
}

class Troncon {
    - depart: Lieu
    - arrivee: Lieu
    - modalite: ModaliteTransport

    + Troncon(Lieu depart, Lieu arrivee, ModaliteTransport modalite)
    + Troncon(String sdepart, String sarrivee, ModaliteTransport modalite)
```

```
    + Troncon(String sdepart, String sarrivee, String modalite)
    + getArrivee(): Lieu
    + getDepart(): Lieu
    + getModalite(): ModaliteTransport
    + inversTrancon(): Trancon
    + toString(): String
}

class MonLieu {
    - nom: String

    + MonLieu(String nom)
    + getNom(): String
    + equals(Object obj): boolean
    + toString(): String
}

class Main {
    + _main(String[] args): void
    + _toString(List<Chemin> l): String
}

Plateforme --> "1" Voyageur
Plateforme --> "1" ModaliteTransport : m
Plateforme --> "1" TypeCout : coutChoisi

Main --> Plateforme
Plateforme --> Troncon
Plateforme --> MonLieu
Troncon --> MonLieu : depart
Troncon --> MonLieu : arrivee
Troncon --> ModaliteTransport : modalite


class ModaliteTransport {
    <<enumeration>>
    TRAIN
    BUS
    AVION
}

class TypeCout {
    <<enumeration>>
    CO2
    TEMPS
    PRIX
```

```
}
```

Explication de l’architecture

Les programmes java sont séparés en deux packages ;

graphe : - contient l’enum TypeCout qui liste les différent criteres et les classes MonLieu et Troncon qui implémente respectivement les interfaces Lieu et Trancon qui sont fournies dans l’archive jar sae_s2_2024.jar.

app : - contient les classes Main, Plateforme et Voyageur. Elles utilisent les classes présentes dans graphes. Voyageur sers a représenter l’utilisateur auprès de Plateforme. Plateforme créé les graphes en fonction des input et des données de l’utilisateur. Main se lance au démarage de Application.jar et appelle successivement les fonctions permettant a l’utilisateur d’interagir avec la classe Plateforme.

TestPlateforme.java

La classe de test TestPlateforme.java implémente 10 test différents sur la classe Plateforme.