4A – Architecture des Logiciels

ESGI 2013/2014

Lundi 09 Décembre 2013

Journey Calculator

Joey Bronner | Amine Bouabdallaoui

Sommaire

PRESENTATION GENERALE

Présentation générale (1/4)

Les données sources

RATP [beta] open data

http://data.ratp.fr/fr/les-donnees.html

Présentation générale (2/4)

Les données sources

agency.txt	Document texte	1 Ko	Non	15 Ko	100 %	07/10/2013 11:33
calendar.txt	Document texte	23 Ko	Non	199 Ko	89 %	07/10/2013 11:33
calendar_dates.txt	Document texte	326 Ko	Non	2 380 Ko	87 %	07/10/2013 11:33
routes.txt	Document texte	14 Ko	Non	87 Ko	85 %	07/10/2013 11:33
stop_times.txt	Document texte	57 249 Ko	Non	523 809 Ko	90 %	07/10/2013 11:33
stops.txt	Document texte	540 Ko	Non	1 916 Ko	72 %	07/10/2013 11:33
trips.txt	Document texte	1 943 Ko	Non	17 110 Ko	89 %	07/10/2013 11:33

Présentation générale (3/4)

Le stockage des données

Présentation générale (4/4)

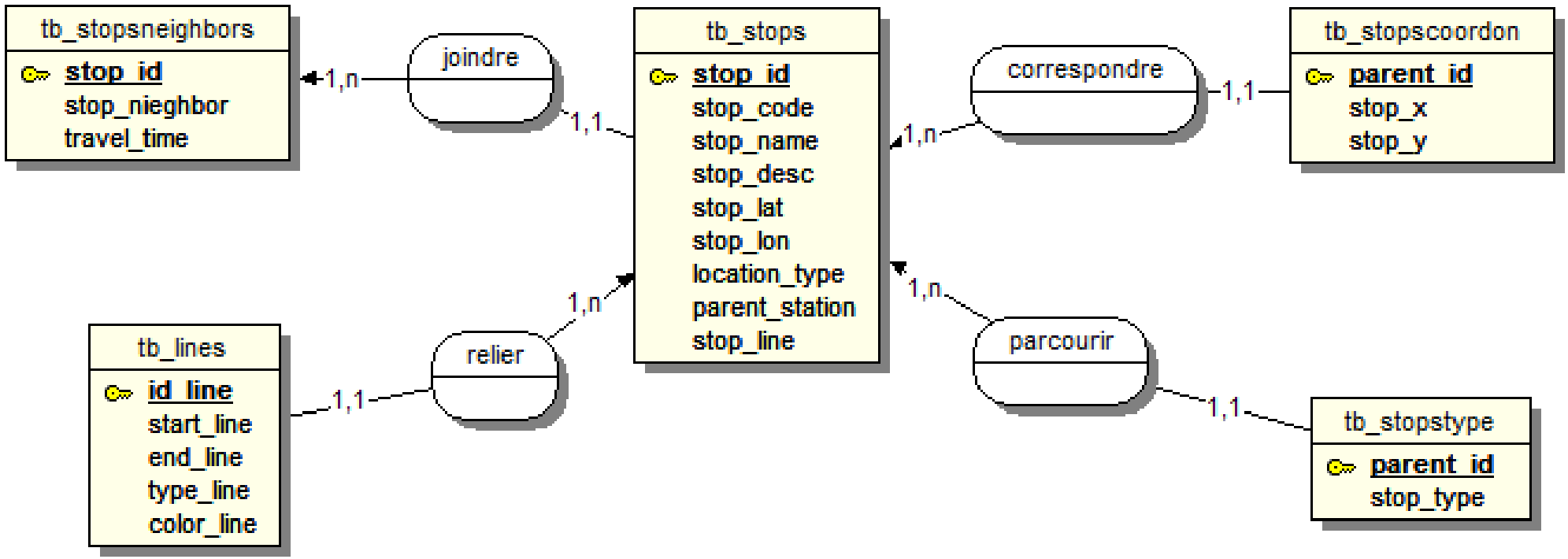
L'interface

Swing

MODELE CONCEPTUEL

Modèle Conceptuel de Données (1/1)

Le MCD



LE CODE

Le Code (1/6)

Station()

```
Station(int id, int x, int y, String nom_stat, String nom_line, String colorLine)
{
    this.id = id;
    neighborList = new ArrayList<Station>();
    this.x = x;
    this.y = y;
    this.visited = false;
    this.distanceFromStart = Integer.MAX_VALUE;
    this.isStart = false;
    this.isDestination = false;
    this.nomStat = nom_stat;
    this.nomLign = nom_line;
    this.color = colorLine;
}
```

Le Code (2/6)

createMap()

```
ResultSet stations = connexion.Rechercher("SELECT st.stop_id, stop_name, stop_x, stop_y, parent_id, stop_line, color_line F

int count=0;
try {
    HashMap<Integer, Station> hm;
    while (stations.next()) {
        count++;
        sta = new Station(Integer.parseInt(stations.getString("stop_id")), Integer.parseInt(stations.getString("stop_x")),
        maMapByIndex.put(Integer.parseInt(stations.getString("stop_id")), sta);
        if (maMap.get(sta.getX()) == null) {
            hm = new HashMap<Integer, Station>();

        } else {
            hm = maMap.get(sta.getX());
        }
        hm.put(sta.getY(), sta);
        maMap.put(sta.getX(), hm);
    }
    System.out.println("Nb de stations ajoutées : " + count);
```

Le Code (3/6)

registerNeighbors()

```
ResultSet voisins = connexion.Rechercher("SELECT * FROM tb_stopsneighbors ne, tb_stops st, tb_stopscoordon co WHERE

count=0;
Station s;
while (voisins.next()) {
    if (voisins.getString("stop_x") != null && voisins.getString("stop_y") != null) {
        //System.out.println(voisins.getString("stop_name"));
        hm = maMap.get(Integer.parseInt(voisins.getString("stop_x")));
        if (hm != null && hm.get(Integer.parseInt(voisins.getString("stop_y"))) != null) {
            s = hm.get(Integer.parseInt(voisins.getString("stop_y")));
            s.addNeighborAtList(maMapByIndex.get(Integer.parseInt(voisins.getString("stop_neighbor"))));
            count++;
        }
    }

}
System.out.println("Nb de voisins ajoutés : " + count);
```

Le Code (4/6)

readStops()

```
public static void readStops(String requete) throws IOException, SQLException
{
    Map<Integer, Arret> arrs = new HashMap<Integer, Arret>();
    DatabaseConnect connexion = new DatabaseConnect();
    connexion.Connexion();
    Arret arr;
    arrs.clear();

    ResultSet reqStops = connexion.Rechercher(requete);

    while (reqStops.next())
    {
        arr = new Arret(reqStops.getString("stop_id"), reqStops.getString("stop_name"), reqStops.getString("stop_type"));
        arrs.put(Integer.valueOf(reqStops.getString("stop_id")), arr);
    }
    connexion.Deconnexion();
}
```

```
Collection<Scrapper.Arret> a = Scrapper.getArrs().values();
Iterator<Arret> it = a.iterator();
while (it.hasNext()) {
    Object ar = it.next();
    comboBoxDepart.addItem(ar.toString());
    comboBoxArrivee.addItem(ar.toString());
}
```

Le Code (5/6)

calcShortestPath()

```
pathFinder.calcShortestPath(depaID, destID);
```

- **Affectation du point de départ et de destination**

```
map.setStartLocation(startID);
map.setGoalLocation(goalID);
```

- **Parcours des voisins**

```
for (Station neighbor : current.getNeighborList()) {
```

- **Ajout du meilleur voisin à l'itinéraire**

```
if (neighborIsBetter) {
    neighbor.setPreviousNode(current);
    neighbor.setDistanceFromStart(neighborDistanceFromStart);
    neighbor.setHeuristicDistanceFromGoal(heuristic.getEstimatedDistanceToGoal(neighbor.getX(),
}
```

Le Code (6/6)

printItineraire()

```
public void printItineraire() {

    int nbChange = 0;
    String nomOldLign = map.getStartNode().nomLign;
    System.out.println("\nNombre de stations traversées : " + shortestPath.getLength());
    System.out.println("Départ    : " + map.getStartNode().nomStat + " (" + map.getStartNode().nomLign + ")");
    for (int i = 0; i < shortestPath.getLength(); i++) {
        int numStat = i + 1;
        int x = shortestPath.getWayPoint(i).getX();
        int y = shortestPath.getWayPoint(i).getY();

        // Check pour voir si il y a changement de ligne
        if (map.getNode(x, y).nomLign != nomOldLign && nomOldLign != "" && i != shortestPath.getLength() - 1) {
            nbChange++;
        }
        nomOldLign = map.getNode(x, y).nomLign;

        if (i != shortestPath.getLength() - 1) {
            System.out.println("Arrêt num" + numStat + " : " + map.getNode(x, y).nomStat + " (" + map.getNode(x, y).nomLign + ")");
        } else {
            System.out.println("Arrivée   : " + map.getNode(x, y).nomStat + " (" + map.getNode(x, y).nomLign + ")");
        }

    }
    System.out.println("\nNombre de changements de ligne(s) : " + nbChange);
}
```

CONCLUSION

Conclusion (1/1)

Améliorations

Re-programmer l'heuristique *(ClosestHeuristic.java)*

Modifier la requête d'ajout des voisins *(registerNeighbors())*

MERCI POUR VOTRE ATTENTION