Projet : Programmation orienté objet

Ce projet a été réalisé avec le logiciel IDE Eclipse (sous sa version 4.6 Neon).

Présentation des classes :

Le dossier « display » continent les deux fichiers support données avec la consigne du projet soit : « Displayable « et « Gui_For_Displayable » que nous avons renommé en « Gui ».
Il contient également le fichier « Objet.java » qui va nous permettre de communiquer avec l'interface Displayable via la classe 'Objet' qui l'implémente.
Cette classe 'Objet' est composée de deux constructeurs permettant de manipuler la mise en forme graphique.

Le dossier « Models »contient tous les fichiers des classes contenant les objets qui communiquent avec la classe principale, un procédé d'encapsulation est mis en œuvre pour garantir l'intégrité des données via des Getters.
Il contient donc les fichiers :
- « Contraintes.java » qui définit les événements qui ont lieu lors de la simulation (apparition de population aux différents arrêts, timer d'apparition, fin de l'évènement …).

- « Line.java » qui va permettre de créer les lignes d'un arrêt à un autre, la couleur de la ligne en fonction du type de véhicule qui la traverse ou la fréquence de passage (vitesse d'apparition des différents véhicules).

- « Population.java » permet de définir les occupants des véhicules ou des arrêts en fonction de leur zone d'apparition et de leur zone de destination.

- « Stop.java » va permettre de contenir une liste de population en attente des arrêts en fonction de leur zone et également de faire le lien graphique pour dessiner des ellipses.

- « Trajet.java » va permettre au code de définir le trajet d'un véhicule sur une ligne donnée en fonction d'un événement de la classe 'Contraintes.java'.

- « Transport.java » permet de définir le mouvement des véhicules sur les lignes en récupérant la position sur les axes X/Y des arrêts ainsi que la vitesse du véhicule, cette classe enregistre également dans un tableau la population qui va monter dans le véhicule pour la soustraire de l'arrêt.

- « TypeLine.java » : permet de différencier une ligne de bus d'une ligne de tramway pour éviter qu'elles puissent se chevaucher et retrouver un bus sur les rails…

- « Zone.java » : qui va permettre de définir l'emplacement des arrêts dans la fenêtre en fonction des axes X/Y. Lors de l'édition du document XML nous avons mis en corrélation les numéros des zones avec des numéros d'ID se rapportant aux postions dans l'espace.

Chacune des informations utilisées par les objets définis dans ces classes utilise des attributs qui sont récupérés dans le fichier « data.xml » qui est un fichier modulable a destination de l'utilisateur et permettant d'augmenter la praticité du code.

Le dossier « projet » contient lui trois fichiers :
- « Game.java » qui est le cœur du code et aussi qui contient la classe 'Game ' qui permet la mise en relation des autres classes citées plus haut. Cette classe permet la lecture du fichier XML ainsi que la restitution de ses informations dans les différents attributs de la classe. Il possède la boucle principale de gestion du temps et permet ainsi le bon déroulement des opérations souhaités.

- « Main.java » qui est le fichier de lancement du projet, il va créer l'objet de la classe Game, lancer le chargement du fichier xml à destination de la classe Game également, générer la population et reporter d'éventuelles erreurs ou exceptions.

- « data.xml » qui est à destination de l'utilisateur et permet la lecture des informations souhaités.

Présentation des difficultés et résolutions :

- Le plus long au début a été d'étudier les documents mis à dispositions avec le projet et de les rendre utilisable pour notre code, pour ce faire nous avons créé la classe 'Objet' qui implémente la classe 'Displayable' fournit avec les documents et permet l'utilisation de ses méthodes.

- Nous aurions aimé réaliser une interaction avec l'utilisateur du type un bouton pause ou ralenti mais cela impliqué de modifier les documents support, nous avons alors abandonné cette idée pour nous concentrer sur la correction des derniers bugs et la lisibilité du programme.

- Sur l'aspect graphique après la première création du document xml la lisibilité du rendu était impossible, nous avons alors refait le xml pour améliorer sa lisibilité et son rendu.

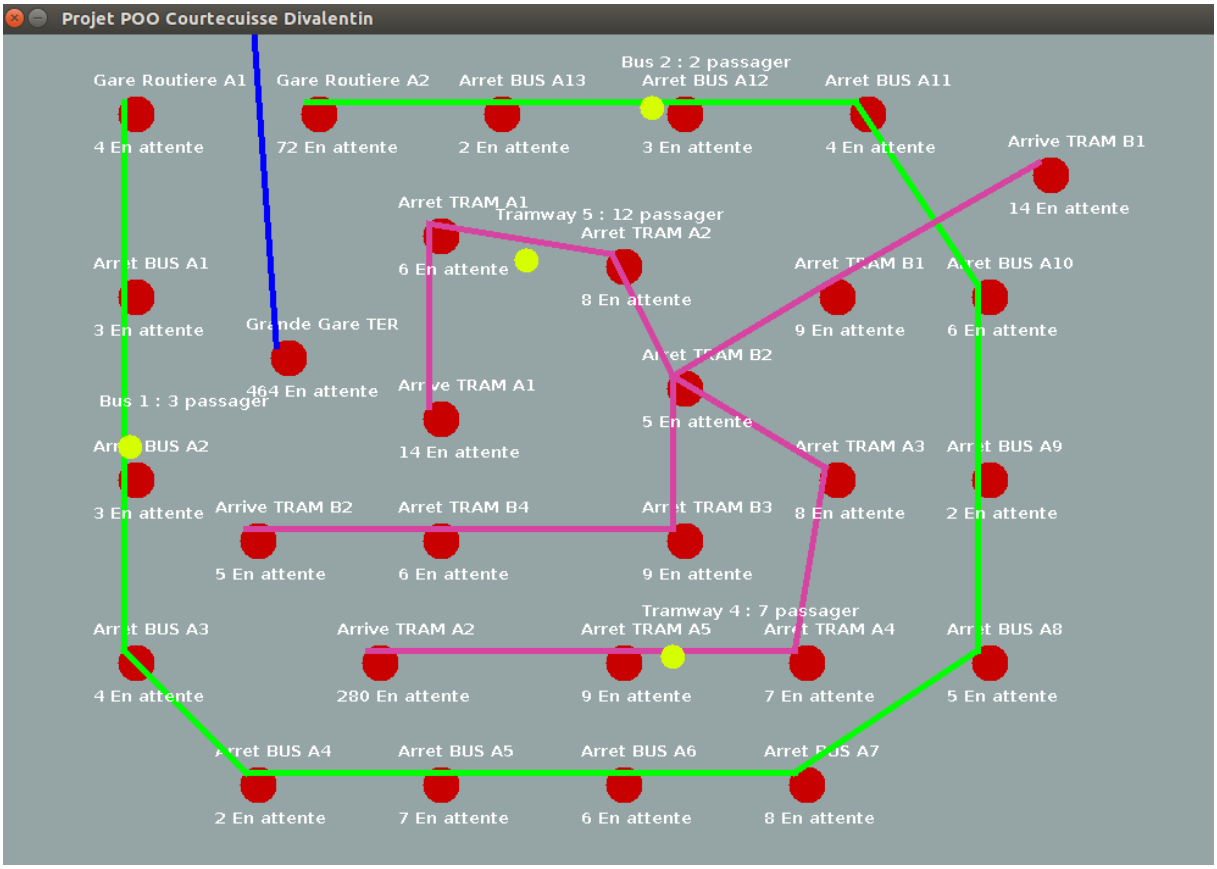
- Après avoir bien avancé dans le projet une erreur était récurrente mais ne gênait pas le fonctionnement du programme :
« Exception in thread "AWT-EventQueue-0" java.util.ConcurrentModificationException
    at java.util.ArrayList$Itr.checkForComodification(Unknown Source)
    at java.util.ArrayList$Itr.next(Unknown Source)
    .... »

Après un certain temps de recherche nous avons déterminé qu'il s'agissait d'une exception présente dans le support 'Gui' du projet concernant la modification d'un arraylist pour l'affichage qui entre en concurrence avec la méthode de suppression d'un diplayable.

- Nous avons également effectué beaucoup de recherches pour trouver la solution la plus adapté pour introduire les informations nécessaires au fonctionnement du code. Nous avons finalement décidé de nous pencher sur la lecture d'un document XML via le code du programme, l'avantage du document XML est qu'il est facilement lisible grâce à ses balises et ses items qui dialoguent avec le programme java à la manière de variables.

- Nous nous sommes rendu compte après la création des classes qui fallait faire correspondre les informations directement avec le fichier XML. Nous avons alors décidé de faire correspondre les zones des éléments du graphique avec les ID des items qui possèdent les positions sur les axes.

Rendu final :



Ici la ligne bleu représente une ligne TER qui sort de la ville, ses véhicules sont des TER au départ de 'Grande Gare TER', ici il y a actuellement 464 personnes en attentes dans la grande gare.

Les lignes roses sont les lignes de Tramway (la ligne A et la ligne B), chaque lignes possèdes ses arrêts qui lui sont propre, il y a un arrêt centrale par ou passent les deux lignes, il s'agit de 'Arret TRAM B2', les lignes de type Tramway possèdes des véhicules qui les parcours dans les deux sens.

Le dernier type de ligne est la ligne verte qui représente celle des BUS, ces véhicules ont une vitesse de déplacement moins importante que celle des Tramway et des TER , elles sont également parcourues dans les deux sens.

Les ellipses jaunes correspondent aux véhicules, soit BUS, TER ou Tramway et il est indiqué au-dessus d'eux le nombre de passager dans leur liste ainsi que leur nom.

Enfin les ellipses rouges représentent les arrêts ou Stops, ils possèdent un nom et le nombre de population en attente dans leur liste.

Projet réalisé par Courtecuisse Erik et Divalentin Ugo.
A l'intention de : Mr Furst et Mme Irastorza.
Licence 2 Informatique Upjv UFR Sciences.

Annexe le code :

Fichier ‘objet.java’ :

```
package display;

import java.awt.Color;
import java.awt.Point;
import java.awt.Shape;

/***
 * Objet qui permet d'afficher
 * @author PCPack
 *
 */
public class Objet implements Displayable{
	private Shape s;
	private Color c;
	private String text;
	private Point p;

	/**
	 * Constructeur 1 pour afficher uniquement une forme sans texte
	 * @param shape
	 * @param color
	 */
	public Objet(Shape s, Color c){
		this.s = s;
		this.c = c;
		this.text = "";
		this.p = new Point();
	}

	/**
	 * Contructeur 2 pour afficher une forme et un texte
	 * @param shape
	 * @param color
	 * @param text
	 * @param point
	 */
	public Objet(Shape s, Color c, String text, Point p){
		this.s = s;
		this.c = c;
		this.text = text;
		this.p = p;
	}

	@Override
	public Shape getShape() {
		return this.s;
	}

	@Override
```

```
	public Color getColor() {
		return this.c;
	}

	@Override
	public String getString() {
		return this.text;
	}

	@Override
	public Point getStringPosition() {
		return this.p;
	}

}
```

Fichier 'Contraintes.java' :

```
package models;

import java.awt.Point;

/**
 * Lieu de dÃ©placement des personnes
 * @author PCPack
 *
 */
public class Contraintes{
	private int id; // Id
	private String name; // Nom de repÃ©re
	private Zone zone; // Zone ou se situe le lieu
	private long heureDebut; // Heure ou les personnes partent
	private long heureFin; // Heure ou les personnes s'en vont

	private long nombreVisiteur; // Nombre de visiteur

	public Contraintes(int id, String name, Zone zone, long heureDebut, long heureFin, int
nombreVisiteur) {
		this.id = id;
		this.name = name;
		this.zone = zone;
		this.heureDebut = heureDebut;
		this.heureFin = heureFin;
		this.nombreVisiteur = nombreVisiteur;
	}

	// ************************************
	// GETTER
	// ************************************
	public Zone getZone() {
		return this.zone;
	}
```

```
	public long getHeureDebut() {
		return this.heureDebut;
	}

	public long getHeureFin() {
		return this.heureFin;
	}

	// ************************************
	// OTHER METHOD
	// ************************************
	public String toString(){
		return "id:" + this.id + ", name:" + this.name + ", zone:{" + this.zone.toString()  + "},
heureDebut:" + this.heureDebut  + ", heureFin:" + this.heureFin + ", nombreVisiteur:" +
this.nombreVisiteur;
	}

	public long getNombreVisiteur() {
		return this.nombreVisiteur;
	}
}
```

Fichier 'Line.java' :

```
package models;

import java.awt.Color;
import java.util.ArrayList;

/**
 * Model contenant chaque ligne avec ses informations
 * @author PCPack
 *
 */
public class Line{
	private int id; // Identifiant unique de la ligne
	private String name; // Nom de la ligne pour repÃ©re
	private TypeLine type; // Type de ligne (Metro, Train, etc...)
	private Stop[] stop; // Liste de ses arrets
	private long frequence; // FrÃ©quence de passage
	private Color color; // Couleur d'affichage

	/**
	 * Constructeur unique
	 * @param id
	 * @param name
	 * @param type
	 * @param stop
	 * @param frequence
	 * @param color
	 */
	public Line(int id, String name, TypeLine type, Stop[] stop, long frequence, Color color){
		this.name = name;
		this.id = id;
```

```
                this.type = type;
                this.stop = stop;
                this.frequence = frequence;
                this.color = color;
        }

        // ***************************************
        // GETTER
        // ***************************************
        public Color getColor(){
                return this.color;
        }
        public long getFrequence(){
                return this.frequence;
        }
        public Stop[] getStops() {
                return this.stop;
        }
        public TypeLine getType(){
                return this.type;
        }
        public String getName() {
                return this.name;
        }

        // ***************************************
        // OTHER METHODS
        // ***************************************
        public String toString(){
                return "id:" + this.id + ", name:" + this.name + ", type:{" + this.type.toString() + "}";
        }
}
```

Fichier ‘Population.java’ :

```
package models;

/**
 * Model caractairisant une personne
 * @author PCPack
 */
public class Population{
        private Zone zone; // Zone ou se trouve la personne
        private Contraintes occupation; // L'endroit ou la personne se dÃ©place en gÃ©nÃ©ral
        private Boolean haveCar; // Si la personne a une voiture ou non

        /**
         * Constructeur d'une personne
         * @param zone
         * @param occupation
         * @param haveCar
         */
```

```
	public Population(Zone zone, Contraintes occupation, Boolean haveCar){
		this.zone = zone;
		this.occupation = occupation;
		this.haveCar = haveCar;
	}

	// ********************************************
	// GETTER
	// ********************************************
	public Contraintes getOccupation() {
		return this.occupation;
	}
	public boolean getHaveCar() {
		return this.haveCar;
	}
	public Zone getZone() {
		return this.zone;
	}

	// ********************************************
	// SETTER
	// ********************************************
	public void setOccupation(Contraintes occupation){
		this.occupation = occupation;
	}
	public void setHaveCar(boolean haveCar){
		this.haveCar = haveCar;
	}
	public void setZone(Zone zone){
		this.zone=zone;
	}
}
```

Fichier 'Stop.java' :

```
package models;

import java.util.ArrayList;
import display.Displayable;

/**
 * Model contenant les dÃ©tails de chaque stop
 * @author PCPack
 *
 */
public class Stop {
	private int id; // Id
	private String name; // Nom de repÃ©re
	private Zone zone; // Endroit ou se trouve l'arret
	private TypeLine type; // Type de ligne auquel il correspond
	private Displayable displayable; // Objet graphique
	private ArrayList<Trajet> pepoleWithTrajet; // Liste des personne en attente a l'arret
```

```
/**
 * Constructeur unique
 * @param idT
 * @param nameT
 * @param zone
 * @param type
 */
public Stop(int id, String name, Zone zone, TypeLine type) {
	this.id = id;
	this.name = name;
	this.zone = zone;
	this.type = type;
	this.displayable = null; // Il sera ajoutÃ© au premier affichage
	this.pepoleWithTrajet = new ArrayList<Trajet>();
}

// ************************************
// GETTER
// ************************************
public Displayable getDisplayable(){
	return this.displayable;
}

public ArrayList<Trajet> getPeople(){
	return this.pepoleWithTrajet;
}

public int getId(){
	return this.id;
}

public Zone getZone() {
	return this.zone;
}

public TypeLine getType() {
	return type;
}
public String getName() {
	return this.name;
}

// ************************************
// SETTER
// ************************************
public void setHisDisplayable(Displayable d) {
	this.displayable = d;
}

public void addPeople(Trajet tj) {
	// Ajoute des personnes a l'arret
	this.pepoleWithTrajet.add(tj);
}
```

```
	// ************************************
	// OTHER METHODS
	// ************************************
	public String toString(){
		return "id:" + this.id + ", name:" + this.name + ", zone:{" + this.zone.toString() + "}, type:{" +
this.type.toString() + "}";
	}
}
```

Fichier 'Trajet.java' :

```
package models;

public class Trajet {
	private Population p;
	private int start;
	private int end;
	private long heureDebut;
	private long heureFin;
	private int sens;
	private String nameLine;

	public Trajet(Population p, int stopStart, int stopEnd, long heureDebut, long heureFin, String
nameLine) {
		this.p = p;
		this.start = stopStart;
		this.end = stopEnd;
		this.heureDebut = heureDebut;
		this.heureFin = heureFin;
		this.nameLine = nameLine;
	}

	public String getNameLine(){
		return this.nameLine;
	}

	public int getStartStop(){
		return this.start;
	}

	public int getEndStop(){
		return this.end;
	}

	public Population getPersonn(){
		return this.p;
	}

	public long getHeureDebut(){
		return this.heureDebut;
	}
```

```
	public long getHeureFin(){
		return this.heureFin;
	}

	public void setAller() {
		this.sens = 0;
	}

	public void setRetour() {
		this.sens = 1;
	}
}
```

Fichier 'Transport.java' :

```
package models;

import java.util.ArrayList;
import display.Displayable;

/**
 * Model reprÃ©sentant les transports et leurs informations
 * @author PCPack
 *
 */
public class Transport{
	private int positionX; // Position x
	private int positionY; // position Y
	private Stop[] stop; // Liste des stop ou se rend le transport ten question
	private int vitesse; // Sa vitesse
	private String name; // Son nom
	private int goTo; // Le prochain id ou le bus se rend
	private String nameLine; // Nom de la ligne
	private Displayable displayable; // Sont affichage
	private ArrayList<Trajet> trajetPeople; // Les personnes qui sont dans le bus
	private int capaciteMax;

	/**
	 * Constructeur unique
	 * @param i
	 * @param j
	 * @param name
	 * @param k
	 * @param stops
	 * @param nameLine
	 * @param capaciteMax
	 */
	public Transport(int i, int j, String name, int k, Stop[] stops, String nameLine, int capaciteMax) {
		this.positionX = i;
		this.positionY = j;
		this.name = name;
```

```
        this.vitesse = k;
        this.stop = stops;
        this.goTo = 0;
        this.displayable = null;
        this.nameLine = nameLine;
        this.trajetPeople = new ArrayList<Trajet>();
        this.capaciteMax = capaciteMax;
}

// ************************************
// GETTER
// ************************************
public String getNameLine(){
        return this.nameLine;
}

public int getCapaciteMax(){
        return this.capaciteMax;
}

public Displayable getDisplayable(){
        return this.displayable;
}

public Stop[] getStop(){
        return this.stop;
}

public int getGoTo(){
        return this.goTo;
}

public int getX(){
        return this.positionX;
}

public int getY(){
        return this.positionY;
}

public int getVitesse() {
        return this.vitesse;
}

public String getName(){
        return this.name;
}

public void goToNextStop(){
        this.goTo+=1;
}

public void updateX(int toX) {
        this.positionX = toX;
```

```
	}

	public void updateY(int toY) {
		this.positionY = toY;
	}

	public void setHisDisplayable(Displayable d) {
		this.displayable = d;
	}

	public ArrayList<Trajet> getTrajetPeople() {
		return this.trajetPeople;
	}
}
```

Fichier 'TypeLine.java' :

```
package models;

import java.awt.Point;

/**
 * Model contenant les types de lignes
 * @author PCPack
 *
 */
public class TypeLine {
	private int id;
	private String name;
	private int nombrePersonneMaxParTransport;

	/**
	 * Consturcteur unique
	 * @param id
	 * @param name
	 * @param nombrePersonneMaxParTransport
	 */
	public TypeLine(int id, String name, int nombrePersonneMaxParTransport){
		this.id = id;
		this.name = name;
		this.nombrePersonneMaxParTransport = nombrePersonneMaxParTransport;
	}

	// ************************************
	// GETTER
	// ************************************
	public int getId() {
		return this.id;
	}

	public int getNombrePersonneMaxParTransport(){
		return this.nombrePersonneMaxParTransport;
```

```
	}

	// ************************************
	// OTHER METHOD
	// ************************************
	public String toString(){
		return "id:" + this.id + " name:" + this.name;
	}
}
```

Fichier ‘Zone.java’ :

```
package models;

import java.awt.Point;

/**
 * Zone dans la ville
 * @author PCPack
 *
 */
public class Zone {
	private int id; // Id unique pour repere
	private String name; // Nom pour repÃ©re
	private Point position; // La position XY sur la map

	/**
	 * Consturcteur unique
	 * @param id
	 * @param name
	 * @param position
	 */
	public Zone(int id, String name, Point position){
		this.id = id;
		this.name = name;
		this.position = position;
	}

	// ************************************
	// GETTER
	// ************************************
	public Point getPosition(){
		return this.position;
	}

	public int getId(){
		return this.id;
	}

	// ************************************
	// OTHER METHOD
	// ************************************
```

```
	public String toString(){
		return "id:" + this.id + " name:" + this.name + " position:" + this.position.getX() + "/" +
this.position.getY();
	}
}
```

Fichier 'Game.java' :

```
package projet;

import java.util.ArrayList;
import java.util.Observable;
import java.util.Observer;
import java.awt.Color;
import java.awt.Point;
import java.io.File;
import java.io.IOException;
import java.awt.Shape;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Line2D;
import java.awt.geom.Path2D;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import display.Displayable;
import display.Objet;
import display.Gui;
import models.Contraintes;
import models.Line;
import models.Population;
import models.Stop;
import models.Trajet;
import models.Transport;
import models.TypeLine;
import models.Zone;

/**
 * Cette classe est la classe principale du programme
 * @author PCPack
```

```
 */
public class Game{
	private ArrayList<Population> population; // Liste d'une population
	private Line[] lines; // Liste des lignes dans la ville
	private Stop[] stops; // Liste des stops
	private Zone[] zones; // Liste des zones de la ville
	private TypeLine[] typeLine; // Liste des types de lignes
	private Contraintes[] contraintes;
	private Point mapSize; // Taille de la map
	private ArrayList<Trajet> trajetList;
	private ArrayList<Transport> transport;
	private Gui aff; // Objet pour l'affichage

	private long time; // Heure en minutes
	private int speed; // Vitesse d'execution

	/**
	 * Constructeur unique, initialise les donnÃ©es
	 * Pensez a appeller les methodes suivantes <br>
	 * {@link #loadData()} <br>
	 * {@link #createPopulation()}
	 */

	public Game(){
		this.time = 0;
		this.speed = 1;
		this.mapSize = null;
		this.trajetList = new ArrayList<Trajet>();
		this.transport = new ArrayList<Transport>();
	}

	/**
	 * Permet de charger les donnÃ©es
	 * @return Code d'erreur, 0 => OK ; -1 FAILED
	 */
	public int loadData(){
		/* loadData() ne charge pas directement les donnÃ©es a partir
		 * du xml car si on souhaite faire Ã©voluer le programme on pourra
		 * par exemple ici choisir de ne pas charger a partir du xml car l'utilisateur
		 * aura choisi d'entrer ses donnÃ©es via une interface ou encore permettre de charger
		 * a partir d'un autre format comme le txt ou autre
		 */

		System.out.println("Chargement des data");
	System.out.println("-------------------------------------");

		if(loadingDataFromXml() == -1){
			return -1;
		}
		return 0;
	}

	/**
	 * Met a jour les donnees affichÃ©s
```

```
	 */
	private void udpateUI() {
		// Met a jour l'affichage des stops
		for(int i = 0 ; i < stops.length ; i++){
			Stop s = stops[i];

			String textToShow = s.getPeople().size() + " En attente";

			// Objet qui sera affichÃ©
			Displayable d = (Displayable)new Objet(new Ellipse2D.Float(
					0,
					0,
					0,
					0),
					new Color(213,110,0),
					textToShow,
					new Point((int)s.getZone().getPosition().getX()-
20,(int)s.getZone().getPosition().getY() + 35));

			// Si un objet Ã©tait deja affichÃ© on le supprime pour afficher le nouveau
			if(s.getDisplayable() != null){
				this.aff.removeDisplayable(s.getDisplayable());
			}

			// Affichage du nouvel objet
			s.setHisDisplayable(d);
			this.aff.addDisplayable(d);
		}

		// Met a jour l'affichage des transports en transit
		for(Transport tr : transport){
			String textToShow = tr.getNameLine() + " : " + tr.getTrajetPeople().size() + "
passager";

			// Objet d'affichage
			Displayable d = (Displayable)new Objet(new Ellipse2D.Float(
					tr.getX(),
					tr.getY(),
					20,
					20),new Color(213,255,0), textToShow, new Point(tr.getX() - 15,
tr.getY() - 35));

			// Si un objet etait deja afficher on le supprime pour afficher le nouveau
			if(tr.getDisplayable() != null){
				this.aff.removeDisplayable(tr.getDisplayable());
			}

			// Affichage du nouvel objet
			tr.setHisDisplayable(d);
			this.aff.addDisplayable(d);
		}
	}

	/**
```

```
 * Affiche les premiers Ã©lÃ©ments d'affichage (Les points des arret, ..)
 */
private void showUIBase() {
	// Stops
	for(int i = 0 ; i < stops.length ; i++){
		Stop l = stops[i];

		Point p = new Point();
		p.x = (int) l.getZone().getPosition().getX() - 20;
		p.y = (int) l.getZone().getPosition().getY() - 20;

		// Objet d'affichage
		this.aff.addDisplayable((Displayable) new Objet(new Ellipse2D.Float(
					(float)l.getZone().getPosition().getX(),
					(float)l.getZone().getPosition().getY(),
					30,
					30), new Color(200,0,0),
					l.getName(), p));
	}

	// Lines (Ligne entre les stops)
	for(int i = 0 ; i < lines.length ; i++){
		Line l = lines[i];
		Stop[] lineStop = l.getStops();

		for(int u = 0 ; u < lineStop.length - 1 ; u++){

			// Objet
			this.aff.addDisplayable((Displayable) new Objet(new Line2D.Float(
					(float)lineStop[u].getZone().getPosition().getX()+5,
					(float)lineStop[u].getZone().getPosition().getY()+5,
					(float)lineStop[u+1].getZone().getPosition().getX()+5,
					(float)lineStop[u+1].getZone().getPosition().getY()+5),
					l.getColor()));
		}
	}
}

/**
 * Distribue les trajets en attente vers les stop a la bonne heure
 */
// Met les trajet prÃ©vu dans les arret de bus a l'heure demandÃ©
private void goToStop(){
	ArrayList<Trajet> newTJ = new ArrayList<Trajet>(); // Copie de liste

	for(Trajet tj : trajetList){
		// Si l'heure du dÃ©part et dÃ©passÃ©
		if(time >= tj.getHeureDebut()){
			tj.setAller();
			//System.out.println("ajout de personne au stop");
			this.stops[tj.getStartStop()].addPeople(tj);
		}
		// Si l'heure de retour est dÃ©passÃ©
		else if(time >= tj.getHeureFin()){
```

```
                                tj.setRetour();
                                this.stops[tj.getEndStop()].addPeople(tj);
                        }
                        // Sinon il n'y a pas lieu de mettre la personne dans un arret alors on le met dans
la nouvelle liste
                        else{
                                newTJ.add(tj);
                        }
                }

                // On met la nouvelle liste (Cad avec les gens qui sont parti aux arret en moins)
                this.trajetList = newTJ;
        }

        /**
         * Permet de generer une population suivant les lieu ou elle peut se rendre
         */
        // Permet de crÃ©er la population
        public void createPopulation() {
                population = new ArrayList<Population>();

                int contraintesL = contraintes.length;
                int zoneNb = this.zones.length - 1;

                // Pour chaque contraintes existantes
                for(int i = 0 ; i < contraintesL ; i++){
                        Contraintes c = contraintes[i];
                        long nombrePersonne = c.getNombreVisiteur(); // NB de personne pour cette
contrainte

                        // Pour chaque personne crÃ©ation d'une zone de rÃ©sidence alÃ©atoire
                        for(int u = 0 ; u < nombrePersonne ; u++){
                                int zoneAleatoire = 0 + (int)(Math.random() * ((zoneNb - 0) + 1));
                                int carAleatoire = 0 + (int)(Math.random() * ((100 - 0) + 1));

                                // Ajout de la personne dans la liste des personnes
                                population.add(new Population(
                                                this.zones[zoneAleatoire],
                                                c,
                                                (carAleatoire < 10) ? true : false
                                ));
                        }
                }

                System.out.println("Population genere de " + population.size() + " personnes");
        }

        /**
         * Pour chaque personnes qui effectue un trajet a cette heure on les met dans leur arrets
         */
        public void addPeopleToStop(){
                System.out.println("Population : " + population.size());
                // Pour chaque personne
                for(Population p : population){
```

```
                    Contraintes ct = p.getOccupation();

                    // si elle a bien une occupation et pas de voiture
                    if(ct != null && p.getHaveCar() == false){
                        Trajet trajetTrouve = null;
                int pZone = p.getZone().getId();
                boolean find = false;

                // Tant que le trajet n'est pas trouvÃ©
                while(trajetTrouve == null && pZone > -1){
                    for(Line l : lines){
                        Stop[] stops = l.getStops();
                        Stop stopStart = null;
                        Stop stopEnd = null;

                        for(Stop s : stops){
                            if(s.getZone().getId() == pZone){
                                stopStart = s;
                            }
                            if(stopStart != null && s.getZone().getId() == ct.getZone().getId()){
                                stopEnd = s;
                            }
                        }

                        if(stopStart != null && stopEnd != null){
                            trajetTrouve = new Trajet(p, stopStart.getId(), stopEnd.getId(),
ct.getHeureDebut(), ct.getHeureFin(), l.getName());
                            break;
                        }
                    }

                    if(trajetTrouve != null){
                        trajetList.add(trajetTrouve);
                    }
                    else{
                        if(!find){
                            pZone=48;
                            find = true;
                        }
                        pZone--;
                    }
                }
            }
        }
      }

        /**
         * Faire tourner la simulation
         * @throws InterruptedException
         */
        public void runGame() throws InterruptedException{
            this.aff = new Gui("Projet POO Courtecuisse Divalentin", mapSize.x, mapSize.y, new
Color(149, 165, 166));
```

```
		showUIBase();

		// Boucle principale
		while(true){
			// Ralenti
			Thread.sleep(100);

			// On met a jour toutes les minutes les personnes qui doivent aller dans les arrets
			if(trajetList.size() != 0){
				goToStop();
			}

			// Pour chaque ligne, a chaque frÃ©quence on rajoute un transport
			for(int i = 0 ; i < lines.length ; i++){
				if(time % lines[i].getFrequence() == 0){
					Stop[] s = lines[i].getStops();
					switch(lines[i].getType().getId()){
						case 0 :
							transport.add(new
Transport(s[0].getZone().getPosition().x,s[0].getZone().getPosition().y,"Bus x", 15, s, lines[i].getName(),
40));
							break;
						case 1 :
							transport.add(new
Transport(s[0].getZone().getPosition().x,s[0].getZone().getPosition().y,"Metro x", 4, s, lines[i].getName(),
90));
							break;
						case 2 :
							transport.add(new
Transport(s[0].getZone().getPosition().x,s[0].getZone().getPosition().y,"Train x", 8,  s, lines[i].getName(),
140));
							break;
					}
					System.out.println("New transport added on line : " + i + " time :
" + (time % lines[i].getFrequence())  + " freq : " + lines[i].getFrequence());
				}
			}

			// Mise a jour de la liste des transports
			ArrayList<Transport> newTransport = new ArrayList<Transport>();

			// Mouvement des transport, mise a jour des passager
			for(Transport tr : transport){
				Zone s = tr.getStop()[tr.getGoTo()].getZone();

				// Le transport est a son arret
				if(tr.getX() == s.getPosition().x && tr.getY() == s.getPosition().y){

					// On fait descendre les passager
					ArrayList<Trajet> trajetDescendre = tr.getTrajetPeople();
					for(int pp = 0 ; pp < trajetDescendre.size(); pp++){

						Trajet tp = trajetDescendre.get(pp);
```

```
                                        if(tp.getEndStop() == tr.getStop()[tr.getGoTo()].getId()){
                                                trajetDescendre.remove(tp);
                                        }
                                }

                                // On fait monter les passager
                                ArrayList<Trajet> trajetMonter =
tr.getStop()[tr.getGoTo()].getPeople();
                                for(int pp = 0 ; pp < trajetMonter.size() && tr.getCapaciteMax() >
tr.getTrajetPeople().size(); pp++){
                                        Trajet tp = trajetMonter.get(pp);
                                        if(tp.getNameLine() == tr.getNameLine()){
                                                tr.getTrajetPeople().add(tp);
                                                trajetMonter.remove(tp);
                                        }
                                }

                                // On l'envoie au prochain arret
                                if(tr.getGoTo() == tr.getStop().length - 1){
                                        this.aff.removeDisplayable(tr.getDisplayable());
                                        System.out.println("Fin de ligne");
                                }
                                else{
                                        newTransport.add(tr);
                                        tr.goToNextStop();
                                }
                        }
                        else{ // On fait bouger le transport
                                int trX = tr.getX();
                                int trY = tr.getY();
                                int vitesse = tr.getVitesse();
                                int toX = s.getPosition().x;
                                int toY = s.getPosition().y;

                                int maxx, minx, maxy, miny;

                                // DÃ©fini le plus proche, x ou y
                                if(trX > toX){
                                        maxx = trX; minx = toX;
                                }
                                else{
                                        maxx = toX; minx = trX;
                                }

                                if(trY < toY){
                                        miny = trY;
                                        maxy = toY;
                                }
                                else{
                                        miny = toY;
                                        maxy = trY;
                                }

                                if(maxx - minx <= maxy - miny){
```

```
                        // On bouge de Y
                        if(trY > toY){
                                if(trY - toY - vitesse <= 0){
                                        tr.updateY(toY);
                                }
                                else{
                                        tr.updateY(trY - vitesse);
                                }
                        }
                        else{
                                if(toY - trY - vitesse <= 0){
                                        tr.updateY(toY);
                                }
                                else{
                                        tr.updateY(trY + vitesse);
                                }
                        }
                }
                else{
                        // On bouge de X
                        if(trX > toX){
                                if(trX - toX - vitesse <= 0){
                                        tr.updateX(toX);
                                }
                                else{
                                        tr.updateX(trX - vitesse);
                                }
                        }
                        else{
                                if(toX - trX - vitesse <= 0){
                                        tr.updateX(toX);
                                }
                                else{
                                        tr.updateX(trX + vitesse);
                                }
                        }
                }

                newTransport.add(tr);
        }
}
this.transport = newTransport;

// Passage des jours
if(time == 0){
        addPeopleToStop();
}

if(time >= 1440000){
        time = 0;
}
else{
        time+=60*speed;
}
```

```
                udpateUI();
            }
        }

        /**
         * Charge les donnÃ©es a partir d'un fichier xml
         * @return
         */
        // !!
        private int loadingDataFromXml() {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

            try {
                DocumentBuilder builder = factory.newDocumentBuilder();
              File fileXML = new File("src/projet/data.xml");
              Document xml = builder.parse(fileXML);
              Element root = xml.getDocumentElement();
              XPathFactory xpf = XPathFactory.newInstance();
              XPath path = xpf.newXPath();
              String expression;

              // CONFIG
              expression = "/data/config/mapX";
              String mapSizeX = (String)path.evaluate(expression, root);
              expression = "/data/config/mapY";
              String mapSizeY = (String)path.evaluate(expression, root);

              this.mapSize = new Point();
              this.mapSize.x = Integer.parseInt(mapSizeX);
              this.mapSize.y = Integer.parseInt(mapSizeY);

              System.out.println("MAP X SIZE : " + this.mapSize.x);
              System.out.println("MAP Y SIZE : " + this.mapSize.y);


              // ZONES
              System.out.println("-------------------------------------");
              System.out.println("Chargement des Zones");
              System.out.println("-------------------------------------");

              expression = "/data/zone/item";
              NodeList listZones = (NodeList)path.evaluate(expression, root,
XPathConstants.NODESET);
              int listZonesL = listZones.getLength();
              this.zones = new Zone[listZonesL];

              //Parcours de la boucle
              for(int i = 0 ; i < listZonesL; i++){
                Node n = listZones.item(i);
                    NamedNodeMap attr = n.getAttributes();
                int idZ = Integer.parseInt(attr.getNamedItem("id").getNodeValue());
                String nameZ = (String)attr.getNamedItem("name").getNodeValue();
                int posX = Integer.parseInt(attr.getNamedItem("posX").getNodeValue());
```

```
                        int posY = Integer.parseInt(attr.getNamedItem("posY").getNodeValue());
                        Point pos = new Point();
                        pos.x = posX;
                        pos.y = posY;

                        // Verification de l'ordre
                        if(idZ != i){
                                System.out.println("Erreur, les id doivent etre dans l'ordre croissant et se
suivre");
                                return -1;
                        }

                        // Verification que les point sont bien dans la map
                        if(posX > this.mapSize.getX() || posY > this.mapSize.getY()){
                                System.out.println("Erreur : La position de la zone est hors limite de la
map, voir config node");
                                return -1;
                        }

                        // Ajout des zones
                        this.zones[i] = new Zone(idZ, nameZ, pos);
                        System.out.println("Ajout d'une nouvelle zone : " + this.zones[i].toString());
                }

                // TYPES LINES
                System.out.println("-------------------------------------");
                System.out.println("Chargement des types de lignes");
                System.out.println("-------------------------------------");

                expression = "/data/types/item";
                NodeList listTypes = (NodeList)path.evaluate(expression, root,
XPathConstants.NODESET);
                int listTypesL = listTypes.getLength();

                this.typeLine = new TypeLine[listTypesL];

                //Parcours de la boucle
                for(int i = 0 ; i < listTypesL; i++){
                        Node n = listTypes.item(i);
                        NamedNodeMap attr = n.getAttributes();
                    int idT = Integer.parseInt(attr.getNamedItem("id").getNodeValue());
                    int nombrePersonneMaxParTransport =
Integer.parseInt(attr.getNamedItem("nombrePersonneMaxParTransport").getNodeValue());
                    String nameT = (String)attr.getNamedItem("name").getNodeValue();

                    // Verification de l'ordre
                    if(idT != i){
                                System.out.println("Erreur, les id doivent etre dans l'ordre croissant et se
suivre");
                                return -1;
                    }

                    // Ajout des zones
                    this.typeLine[i] = new TypeLine(idT, nameT, nombrePersonneMaxParTransport);
```

```
                        System.out.println("Ajout d'un nouveau type de ligne : " +
this.typeLine[i].toString());
                    }

                // STOPS
                    System.out.println("-------------------------------------");
                    System.out.println("Chargement de la liste des stop");
                    System.out.println("-------------------------------------");

                    expression = "/data/stop/item";
                    NodeList listStop = (NodeList)path.evaluate(expression, root,
XPathConstants.NODESET);
                    int listStopL = listStop.getLength();

                    this.stops = new Stop[listStopL];

                    //Parcours de la boucle
                    for(int i = 0 ; i < listStopL; i++){
                            Node n = listStop.item(i);
                            NamedNodeMap attr = n.getAttributes();
                        int idT = Integer.parseInt(attr.getNamedItem("id").getNodeValue());
                        String nameT = (String)attr.getNamedItem("name").getNodeValue();
                        int zone = Integer.parseInt(attr.getNamedItem("zone").getNodeValue());
                        int type = Integer.parseInt(attr.getNamedItem("type").getNodeValue());

                        // Verification de l'ordre
                        if(idT != i){
                                System.out.println("Erreur, les id doivent etre dans l'ordre croissant et se
suivre");
                                return -1;
                        }

                        if(zone >= this.zones.length){
                                System.out.println("Erreur, La zone n'Ã©xiste pas");
                                return -1;
                        }

                        if(type >= this.typeLine.length){
                                System.out.println("Erreur, le type de n'existe pas");
                                return -1;
                        }

                        // Ajout des zones
                        this.stops[i] = new Stop(idT, nameT, this.zones[zone], this.typeLine[type]);
                        System.out.println("Ajout d'un nouveau stop : " + this.stops[i].toString());
                    }


                    // LINES // ? non desservi, stop isset, type correspon, type existe
                    System.out.println("-------------------------------------");
                    System.out.println("Chargement de la liste des lignes");
                    System.out.println("-------------------------------------");

                    expression = "/data/line/item";
```

```
                        NodeList listLine = (NodeList)path.evaluate(expression, root,
XPathConstants.NODESET);
                        int listLineL = listLine.getLength();

                        this.lines = new Line[listLineL];

                        //Parcours de la boucle
                        for(int i = 0 ; i < listLineL; i++){
                                Node n = listLine.item(i);
                                NamedNodeMap attr = n.getAttributes();
                            int id = Integer.parseInt(attr.getNamedItem("id").getNodeValue());
                            String name = (String)attr.getNamedItem("name").getNodeValue();
                            int type = Integer.parseInt(attr.getNamedItem("type").getNodeValue());
                            int freq = Integer.parseInt(attr.getNamedItem("freq").getNodeValue());
                            int r = Integer.parseInt(attr.getNamedItem("r").getNodeValue());
                            int g = Integer.parseInt(attr.getNamedItem("g").getNodeValue());
                            int b = Integer.parseInt(attr.getNamedItem("b").getNodeValue());
                            Color color = new Color(r,g,b);

                            // Verification de l'ordre
                            if(id != i){
                                    System.out.println("Erreur, les id doivent etre dans l'ordre croissant et se
suivre");
                                    return -1;
                            }

                            if(type >= this.typeLine.length){
                                    System.out.println("Erreur, le type de n'existe pas");
                                    return -1;
                            }

                            // Verification et ajout des stops

                            String expression2 = "/data/line/item[@id='"+ i +"']/item";
                                NodeList stopLstXml = (NodeList)path.evaluate(expression2, root,
XPathConstants.NODESET);

                            int stopLstL = stopLstXml.getLength();
                            System.out.println(i);
                                    Stop[] stopLst = new Stop[stopLstL];

                            // Parcour de la liste des stop qu'utilise la ligne
                            for(int u = 0 ; u < stopLstL; u++){
                                        Node item = stopLstXml.item(u);
                                        NamedNodeMap attrStop = item.getAttributes();

                                    int idStop =
Integer.parseInt(attrStop.getNamedItem("id").getNodeValue());

                                    // Verification que le stop existe
                                    if(idStop >= this.stops.length){
                                            System.out.println("Erreur, le stop de n'existe pas ou id
erronee");
                                            return -1;
```

```
                                        }

                                        // Verification que le stop et bien du type de la ligne
                                        if(this.stops[idStop].getType().getId() != type){
                                                System.out.println("Erreur, le type de stop est diffÃ©rent du
type de ligne " + this.stops[idStop].getType().toString() +  " VS " + this.typeLine[type].toString() + " LINE : " +
i);
                                                return -1;
                                        }

                                        System.out.println("new Stop : " + this.stops[idStop].toString());
                                        // Cporrection des data
                                        // ajout de la liste des stop dans la ligne

                                        stopLst[u] = this.stops[idStop];
                                    }

                                // Ajout des lines
                                this.lines[i] = new Line(id, name, this.typeLine[type], stopLst, freq, color);
                                System.out.println("Ajout d'une nouvelle ligne : " + this.lines[i].toString());
                            }


                            // CONTRAINTES == LIEUX
                            System.out.println("--------------------------------------");
                            System.out.println("Chargement de la liste des contraintes");
                            System.out.println("--------------------------------------");

                            expression = "/data/contraintes/item";
                            NodeList listContraintes = (NodeList)path.evaluate(expression, root,
XPathConstants.NODESET);
                            int listContraintesL = listContraintes.getLength();

                            this.contraintes = new Contraintes[listContraintesL];

                            //Parcours de la boucle
                            for(int i = 0 ; i < listContraintesL; i++){
                                    Node n = listContraintes.item(i);
                                    NamedNodeMap attr = n.getAttributes();
                                int idT = Integer.parseInt(attr.getNamedItem("id").getNodeValue());
                                String name = (String)attr.getNamedItem("zone").getNodeValue();
                                int zone = Integer.parseInt(attr.getNamedItem("zone").getNodeValue());
                                long heureDebut =
Long.parseLong(attr.getNamedItem("heureDebut").getNodeValue());
                                long heureFin = Long.parseLong(attr.getNamedItem("heureFin").getNodeValue());
                                int nombreVisiteur =
Integer.parseInt(attr.getNamedItem("nombreVisiteur").getNodeValue());

                                // Verification de l'ordre
                                if(idT != i){
                                        System.out.println("Erreur, les id doivent etre dans l'ordre croissant et se
suivre");
                                        return -1;
                                }
```

```
                        if(zone >= this.zones.length){
                                System.out.println("Erreur, La zone n'Ã©xiste pas");
                                return -1;
                        }

                        // Ajout des zones
                        this.contraintes[i] = new Contraintes(idT, name, this.zones[zone], heureDebut,
heureFin, nombreVisiteur);
                        System.out.println("Ajout d'un nouveau lieu : " + this.contraintes[i].toString());
                     }
                    } catch (ParserConfigurationException e) {
                         System.out.println("Erreur : Forme du fichier XML data");
                      return -1;
                    } catch (SAXException e) {
                         System.out.println("Erreur : ");
                           return -1;
                    } catch (IOException e) {
                         System.out.println("Erreur : Lecture du fichier DATA, url OK ? " + e.toString());
                        return -1;
                         } catch (XPathExpressionException e) {
                         System.out.println("Erreur : Expression xPath invaldie");
                                 return -1;
                         } catch (Exception e) {
                         System.out.println("Erreur non traite !! : " + e.toString());
                         e.printStackTrace();
                                 return -1;
                         }
                        return 0;
                }
}
```

Fichier ‘Main.java’ :

```
package projet;

public class Main{
        /* This is the main method, this instantiate the object Game */
        public static void main (String[] args){
                Game a = new Game();

                // Chargement des donnÃ©es du fichier data.xml
                if(a.loadData() == -1){
                        System.out.println("Exit with faillure");
                }

                // GÃ©nÃ©ration de la population
                a.createPopulation();

                // Lancement du thread game
                try {
```

```
                a.runGame();
            } catch (InterruptedException e) {
                System.out.println("Exit with exception : Game.runGame()");
            }
        }
}
```

Fichier 'Data.xml :

```
<data>
    <zone>
        <item id="0" name="Gare Routiere A1" posX="100" posY="50"></item>
        <item id="1" name="Arret BUS A1" posX="100" posY="200"></item>
        <item id="2" name="Arret BUS A2" posX="100" posY="350"></item>
        <item id="3" name="Arret BUS A3" posX="100" posY="500"></item>
        <item id="4" name="Arret BUS A4" posX="200" posY="600"></item>
        <item id="5" name="Arret BUS A5" posX="350" posY="600"></item>
        <item id="6" name="Arret BUS A6" posX="500" posY="600"></item>
        <item id="7" name="Arret BUS A7" posX="650" posY="600"></item>
        <item id="8" name="Arret BUS A8" posX="800" posY="500"></item>
        <item id="9" name="Arret BUS A9" posX="800" posY="350"></item>
        <item id="10" name="Arret BUS A10" posX="800" posY="200"></item>
        <item id="11" name="Arret BUS A11" posX="700" posY="50"></item>
        <item id="12" name="Arret BUS A12" posX="550" posY="50"></item>
        <item id="13" name="Arret BUS A13" posX="400" posY="50"></item>
        <item id="14" name="Gare Routiere A2" posX="250" posY="50"></item>

        <item id="15" name="Arrive TRAM B1" posX="850" posY="100"></item>
        <item id="16" name="Arret TRAM B1" posX="675" posY="200"></item>
        <item id="17" name="Arret TRAM B2" posX="550" posY="275"></item>
        <item id="18" name="Arret TRAM B3" posX="550" posY="400"></item>
        <item id="19" name="Arret TRAM B4" posX="350" posY="400"></item>
        <item id="20" name="Arrive TRAM B2" posX="200" posY="400"></item>

        <item id="21" name="Arrive TRAM A1" posX="350" posY="300"></item>
        <item id="22" name="Arret TRAM A1" posX="350" posY="150"></item>
        <item id="23" name="Arret TRAM A2" posX="500" posY="175"></item>
        <item id="24" name="Arret TRAM A3" posX="675" posY="350"></item>
        <item id="25" name="Arret TRAM A4" posX="650" posY="500"></item>
        <item id="26" name="Arret TRAM A5" posX="500" posY="500"></item>
        <item id="27" name="Arrive TRAM A2" posX="300" posY="500"></item>

        <item id="28" name="Grande Gare TER" posX="225" posY="250"></item>
        <item id="29" name="Sortie" posX="200" posY="-100"></item>
    </zone>
    <types>
        <item id="0" name="Bus" nombrePersonneMaxParTransport="40"></item>
```

```
        <item id="1" name="Tramway" nombrePersonneMaxParTransport="150"></item>
        <item id="2" name="TER" nombrePersonneMaxParTransport="650"></item>
</types>
<line>
        <item id="0" name="Bus 1" type="1" r="0" g="255" b="0" freq="11000">
                <item id="0" />
                <item id="1" />
                <item id="2" />
                <item id="3" />
                <item id="4" />
                <item id="5" />
                <item id="6" />
                <item id="7" />
                <item id="8" />
                <item id="9" />
                <item id="10" />
                <item id="11" />
                <item id="12" />
                <item id="13" />
                <item id="14" />
        </item>
        <item id="1" name="Bus 2" type="1" r="0" g="255" b="0" freq="7000">
                <item id="14" />
                <item id="13" />
                <item id="12" />
                <item id="11" />
                <item id="10" />
                <item id="9" />
                <item id="8" />
                <item id="7" />
                <item id="6" />
                <item id="5" />
                <item id="4" />
                <item id="3" />
                <item id="2" />
                <item id="1" />
                <item id="0" />
        </item>
        <item id="2" name="Tramway 1" type="0" r="215" g="67" b="161" freq="4000">
                <item id="15" />
                <item id="16" />
                <item id="17" />
                <item id="18" />
                <item id="19" />
                <item id="20" />
        </item>
        <item id="3" name="Tramway 2" type="0" r="215" g="67" b="161" freq="8000">
                <item id="20" />
```

```
                <item id="19" />
                <item id="18" />
                <item id="17" />
                <item id="16" />
                <item id="15" />
        </item>
        <item id="4" name="Tramway 4" type="0" r="215" g="67" b="161" freq="10000">
                <item id="21" />
                <item id="22" />
                <item id="23" />
                <item id="17" />
                <item id="24" />
                <item id="25" />
                <item id="26" />
                <item id="27" />
        </item>
        <item id="5" name="Tramway 5" type="0" r="215" g="67" b="161" freq="7000">
                <item id="27" />
                <item id="26" />
                <item id="25" />
                <item id="24" />
                <item id="17" />
                <item id="23" />
                <item id="22" />
                <item id="21" />
        </item>
        <item id="6" name="TER" type="2" r="0" g="0" b="255" freq="9000">
                <item id="28" />
                <item id="29" />
        </item>
</line>
<stop>
        <item id="0" name="Gare Routiere A1" zone="0" type="1"/>
        <item id="1" name="Arret BUS A1" zone="1" type="1"/>
        <item id="2" name="Arret BUS A2" zone="2" type="1"/>
        <item id="3" name="Arret BUS A3" zone="3" type="1"/>
        <item id="4" name="Arret BUS A4" zone="4" type="1"/>
        <item id="5" name="Arret BUS A5" zone="5" type="1"/>
        <item id="6" name="Arret BUS A6" zone="6" type="1"/>
        <item id="7" name="Arret BUS A7" zone="7" type="1"/>
        <item id="8" name="Arret BUS A8" zone="8" type="1"/>
        <item id="9" name="Arret BUS A9" zone="9" type="1"/>
        <item id="10" name="Arret BUS A10" zone="10" type="1"/>
        <item id="11" name="Arret BUS A11" zone="11" type="1"/>
        <item id="12" name="Arret BUS A12" zone="12" type="1"/>
        <item id="13" name="Arret BUS A13" zone="13" type="1"/>
        <item id="14" name="Gare Routiere A2" zone="14" type="1"/>
```

```
                <item id="15" name="Arrive TRAM B1" zone="15" type="0"/>
                <item id="16" name="Arret TRAM B1" zone="16" type="0"/>
                <item id="17" name="Arret TRAM B2" zone="17" type="0"/>
                <item id="18" name="Arret TRAM B3" zone="18" type="0"/>
                <item id="19" name="Arret TRAM B4" zone="19" type="0"/>
                <item id="20" name="Arrive TRAM B2" zone="20" type="0"/>

                <item id="21" name="Arrive TRAM A1" zone="21" type="0"/>
                <item id="22" name="Arret TRAM A1" zone="22" type="0"/>
                <item id="23" name="Arret TRAM A2" zone="23" type="0"/>
                <item id="24" name="Arret TRAM A3" zone="24" type="0"/>
                <item id="25" name="Arret TRAM A4" zone="25" type="0"/>
                <item id="26" name="Arret TRAM A5" zone="26" type="0"/>
                <item id="27" name="Arrive TRAM A2" zone="27" type="0"/>

                <item id="28" name="Grande Gare TER" zone="28" type="2"/>
                <item id="29" name="Sortie" zone="29" type="2"/>
        </stop>
        <config>
                <mapX>1000</mapX>
                <mapY>700</mapY>
        </config>
        <contraintes>
                <item id="0" zone="28" name="Sortie de travail" heureDebut="400"
heureFin="57600" nombreVisiteur="500"/>
                <item id="1" zone="2" name="Sortie de travail" heureDebut="400" heureFin="57600"
nombreVisiteur="50"/>
                <item id="2" zone="5" name="Sortie de travail" heureDebut="400" heureFin="57600"
nombreVisiteur="50"/>
                <item id="3" zone="8" name="Sortie de travail" heureDebut="400" heureFin="57600"
nombreVisiteur="50"/>
                <item id="4" zone="17" name="Sortie de travail" heureDebut="400"
heureFin="57600" nombreVisiteur="500"/>
                <item id="5" zone="28" name="Vacances" heureDebut="24000" heureFin="576000"
nombreVisiteur="50"/>
                <item id="6" zone="12" name="Broquante" heureDebut="24000" heureFin="576000"
nombreVisiteur="50"/>
                <item id="7" zone="14" name="Sortie ecole" heureDebut="24000"
heureFin="576000" nombreVisiteur="50"/>
                <item id="8" zone="19" name="Sortie ecole" heureDebut="24000"
heureFin="576000" nombreVisiteur="50"/>
                <item id="9" zone="20" name="Sortie ecole" heureDebut="24000"
heureFin="576000" nombreVisiteur="50"/>
        </contraintes>
</data>
```