Algorithmes :

**Classe Thermometre**

*Propriétés privées :*

CouleurLigne  : Chaine

CouleurPoleEchange  : Chaine

LesArrets : List<Arret>  //Liste d’objets de type Arret

LesLignes : List<Ligne>  //Liste d’objets de type Ligne

LongueurSommet 🡨 15 : double

HauteurSommet 🡨 15 : double

EpaisseurTraitSommet 🡨 2 : double

tpEtiquetteNomArret 🡨 15 : double //Taille de police pour les étiquettes des noms d’arrêt

tpEtiquetteCorrespondance 🡨 12 : double //Taille de police pour les étiquettes de correspondances

*Méthodes publiques :*

//Constructeur

*Thermometre(couleurLigne : string, couleurPoleEchange : string, lesArrets : List<Arret>, lesLignes : List<Ligne>)*

Fonction GetCouleurLigne() : chaine

//accesseur sur la propriété CouleurLigne

Fonction GetCouleurPoleEchange() : chaine

//accesseur sur la propriété CouleurPoleEchange

Fonction GetLesArrets() : List<Arret>

//accesseur sur la propriété LesArrets

Fonction GetLesLignes() : List<Ligne>

//accesseur sur la propriété LesLignes

**Fonction Construire()**

//Etape 1 :

**arcPrincipal** 🡨 **CreerArc**(0, 0, 250, 0, CouleurLigne, 2)

canvas.Children.Add(**arcPrincipal**);

**abscisseDebut** 🡨 **arcPrincipal**.X1

**ordonneeDebut** 🡨 **arcPrincipal**.Y1

**rotationEtiquetteNomArretTerminus** : RotateTransform(0);

//Etape 2 :

**Si** **LesArrets**.Count > 0 **Alors**

**Pour** chaque *arret* dans **LesArrets** **Faire**

**Si** *arret* = **LesArrets**.First() **Alors**

**Si** *arret*.GetEstCorrespondance est Vrai **Alors**

**pointArret** 🡨 **CreerSommet**(LongueurSommet, HauteurSommet**,** CouleurLigne**,** EpaisseurTraitSommet**,** *arret*.GetNom**)**

**etiquetteNomArret** 🡨 **CreerEtiquetteNomArret(***arret*.GetNom**,** tpEtiquetteNomArret**,** *arret*.GetEstPoleEchange**)**

**etiquetteCorrespondanceArret** 🡨 **CreerEtiquetteCorrespondance(**tpEtiquetteCorrespondance**,** *arret*.GetEstPoleEchange**)**

**etiquetteNomArret**.RenderTransform 🡨 rotationEtiquetteNomArretTerminus

// Positionnement du point d’arrêt en correspondance

Canvas.SetLeft(**pointArret**, **abscisseDebut**)

Canvas.SetTop(**pointArret**, **ordonneeDebut**)

// Positionnement de l’étiquette du nom du pôle d’échange

Canvas.SetLeft(**etiquetteNomArret**, **ordonneeDebut** + 5)

Canvas.SetTop(**etiquetteNomArret**, **ordonneeDebut** - 230)

// Positionnement de l’étiquette de correspondance du pôle d’échange

Canvas.SetLeft(**etiquetteCorrespondanceArret**, **ordonneeDebut** - 2)

Canvas.SetTop(**etiquetteCorrespondanceArret**, **ordonneeDebut** - 6)

canvas.Children.Add(**pointArret**);

canvas.Children.Add(**etiquetteNomArret**);

canvas.Children.Add(**etiquetteCorrespondanceArret**);

**FinSi**

**Si** *arret*.GetEstPoleEchange est Vrai **Alors**

**pointPoleEchange** 🡨 **CreerSommet**(LongueurSommet, HauteurSommet**,** CouleurPoleEchange**,** EpaisseurTraitSommet**,** *arret*.GetNom**)**

**etiquetteNomPoleEchange** 🡨 **CreerEtiquetteNomArret(***arret*.GetNom**,** tpEtiquetteNomArret**,** *arret*.GetEstPoleEchange**)**

**etiquetteCorrespondancePoleEchange** 🡨 **CreerEtiquetteCorrespondance(**tpEtiquetteCorrespondance**,** *arret*.GetEstPoleEchange**)**

**etiquetteNomPoleEchange**.RenderTransform 🡨 rotationEtiquetteNomArretTerminus

// Positionnement du point de pôle d’échange

Canvas.SetLeft(**pointPoleEchange**, **abscisseDebut**)

Canvas.SetTop(**pointPoleEchange**, **ordonneeDebut**)

// Positionnement de l’étiquette du nom du pôle d’échange

Canvas.SetLeft(**etiquetteNomPoleEchange**, **ordonneeDebut** + 5)

Canvas.SetTop(**etiquetteNomPoleEchange**, **ordonneeDebut** - 230)

// Positionnement de l’étiquette de correspondance du pôle d’échange

Canvas.SetLeft(**etiquetteCorrespondancePoleEchange**, **ordonneeDebut** - 2)

Canvas.SetTop(**etiquetteCorrespondancePoleEchange**, **ordonneeDebut** - 6)

canvas.Children.Add(**pointPoleEchange**);

canvas.Children.Add(**etiquetteNomPoleEchange**);

canvas.Children.Add(**etiquetteCorrespondancePoleEchange**);

**FinSi**

**Si** *arret*.GetEstCorrespondance est Faux **ET** *arret*.GetEstPoleEchange est Faux **Alors**

**pointArret** 🡨 **CreerSommet**(LongueurSommet, HauteurSommet**,** CouleurLigne**,** EpaisseurTraitSommet**,** *arret*.GetNom**)**

**etiquetteNomArret** 🡨 **CreerEtiquetteNomArret(***arret*.GetNom**,** tpEtiquetteNomArret**,** *arret*.GetEstPoleEchange**)**

**etiquetteNomArret**.RenderTransform 🡨 rotationEtiquetteNomArretTerminus

// Positionnement du point d’arrêt en correspondance

Canvas.SetLeft(**pointArret**, **abscisseDebut**)

Canvas.SetTop(**pointArret**, **ordonneeDebut**)

// Positionnement de l’étiquette du nom du pôle d’échange

Canvas.SetLeft(**etiquetteNomArret**, **ordonneeDebut** + 5)

Canvas.SetTop(**etiquetteNomArret**, **ordonneeDebut** - 230)

canvas.Children.Add(**pointArret**);

canvas.Children.Add(**etiquetteNomArret**);

**FinSi**

**FinSi**

**FinPour**

**FinSi**

**CreerArc**(*x1* : double, *y1* : double, *x2* : double, *y2* : double, *couleur* : chaîne, *epaisseurTrait* : double) : Line

{

retourner : Line

{

X1 🡨 *x1*

Y1 🡨 *y1*

X2 🡨 *x2*

Y2 🡨 *y2*

Stroke 🡨 new SolidColorBrush((Color)ColorConverter.ConvertFromString(*couleur*))

StrokeThickness 🡨 *epaisseurTrait*

}

}

**CreerSommet**(*longueur* : double, *hauteur* : double, *couleur* : chaine, *epaisseurTrait* : double, *etiquette* : chaine) : Ellipse

{

retourner : Ellipse

{

Width 🡨 *longueur*

Height 🡨 *hauteur*

Fill 🡨 new SolidColorBrush((Color)ColorConverter.ConvertFromString(*couleur*))

Stroke 🡨 new SolidColorBrush((Color)ColorConverter.ConvertFromString(*couleur*))

StrokeThickness 🡨 *epaisseurTrait*

Tag 🡨 *etiquette*

}

}

**CreerEtiquetteNomArret**(*nom* : chaine, *taillePolice* : double, *estPoleEchange* : booléen) : Label

{

**Si** *estPoleEchange* est Vrai **Alors**

{

retourner : Label

{

Content 🡨 *nom*

FontSize 🡨 *taillePolice*

HorizontalContentAlignment 🡨 HorizontalAlignment.Center,

VerticalContentAlignment 🡨 VerticalAlignment.Center

FontWeight 🡨 FontWeights.Bold

}

}

**Sinon**

{

retourner : Label

{

Content 🡨 *nom*

FontSize 🡨 *taillePolice*

HorizontalContentAlignment 🡨 HorizontalAlignment.Center,

VerticalContentAlignment 🡨 VerticalAlignment.Center

}

}

**FinSi**

}

**CreerEtiquetteCorrespondance**(*taillePolice* : double, *estPoleEchange* : booléen) : Label

{

**Si** *estPoleEchange* est Vrai **Alors**

{

retourner : Label

{

Content 🡨 "C"

FontSize 🡨 *taillePolice*

FontWeight 🡨 FontWeights.Bold

Foreground 🡨 Brushes.Black

}

}

**Sinon**

{

retourner : Label

{

Content 🡨 "C"

FontSize 🡨 *taillePolice*

FontWeight 🡨 FontWeights.Bold

Foreground 🡨 Brushes.White

}

}

**FinSi**

}

**CreerRectangleNoLigne**(*longueur* : double, *hauteur* : double, *couleur* : chaine, *bordureX* : double, *bordureY* : double, *etiquette* : chaine) : Rectangle

{

retourner : Rectangle

{

Width 🡨 *longueur*

Height 🡨 *hauteur*

Stroke 🡨 new SolidColorBrush((Color)ColorConverter.ConvertFromString(*couleur*))

Fill 🡨 new SolidColorBrush((Color)ColorConverter.ConvertFromString(*couleur*))

RadiusX 🡨 *bordureX*

RadiusY 🡨 *bordureY*

Tag 🡨 *etiquette*

HorizontalAlignment 🡨 HorizontalAlignment.Center

VerticalAlignment 🡨 VerticalAlignment.Center

}

}

**CreerEtiquetteNoLigne**(*longueur* : double, *hauteur* : double, *contenu* : chaine, *taillePolice* : double, *etiquette* : chaine) : Label

{

retourner : Label

{

Width 🡨 *longueur*

Height 🡨 *hauteur*

Content 🡨 *contenu*

FontSize 🡨 *taillePolice*

FontWeight 🡨FontWeights.Bold

Foreground 🡨Brushes.White

HorizontalAlignment 🡨 HorizontalAlignment.Center

VerticalAlignment 🡨 VerticalAlignment.Center

Tag 🡨 *etiquette*

}

}

**ModeliserPoleEchange**(

*nomArret* : chaine,

*estPoleEchange* : booléen,

*abscissePointPoleEchange* : double,

*ordonneePointPoleEchange* : double,

*abscisseEtiquetteNomPoleEchange* : double,

*ordonneeEtiquetteNomPoleEchange* : double,

*abscisseEtiquetteCorrespondancePoleEchange* : double,

*ordonneeEtiquetteCorrespondancePoleEchange*: double)

{

**Si** *estPoleEchange* est Vrai **Alors**

{

**pointPoleEchange** 🡨 **CreerSommet**(LongueurSommet, HauteurSommet**,** CouleurPoleEchange**,** EpaisseurTraitSommet**,** *nomArret***)**

**etiquetteNomPoleEchange** 🡨 **CreerEtiquetteNomArret(***nomArret***,** tpEtiquetteNomArret**,** *estPoleEchange***)**

**etiquetteCorrespondancePoleEchange** 🡨 **CreerEtiquetteCorrespondance(**tpEtiquetteCorrespondance**,** *estPoleEchange***)**

// Positionnement du point de pôle d’échange

Canvas.SetLeft(**pointPoleEchange**, *abscissePointPoleEchange*)

Canvas.SetTop(**pointPoleEchange**, *ordonneePointPoleEchange*)

// Positionnement de l’étiquette du nom du pôle d’échange

Canvas.SetLeft(**etiquetteNomPoleEchange**, *abscisseEtiquetteNomPoleEchange*)

Canvas.SetTop(**etiquetteNomPoleEchange**, *ordonneeEtiquetteNomPoleEchange*)

// Positionnement de l’étiquette de correspondance du pôle d’échange

Canvas.SetLeft(**etiquetteCorrespondancePoleEchange**, *abscisseEtiquetteNomPoleEchange*)

Canvas.SetTop(**etiquetteCorrespondancePoleEchange**, *ordonneeEtiquetteCorrespondancePoleEchange*)

canvas.Children.Add(**pointPoleEchange**);

canvas.Children.Add(**etiquetteNomPoleEchange**);

canvas.Children.Add(**etiquetteCorrespondancePoleEchange**);

}

**FinSi**

}

**ModeliserArret**(

*nomArret* : chaine,

*abscissePointArret* : double,

*ordonneePointArret* : double,

*abscisseEtiquetteNomArret* : double,

*ordonneeEtiquetteNomArret* : double,

*abscisseEtiquetteCorrespondanceArret* : double,

*ordonneeEtiquetteCorrespondanceArret* : double,

*estCorrespondance* : booléen)

{

**Si** *estPoleEchange* est Faux **Alors**

{

**pointArret** 🡨 **CreerSommet**(LongueurSommet, HauteurSommet**,** CouleurLigne**,** EpaisseurTraitSommet**,** *nomArret***)**

**etiquetteNomArret** 🡨 **CreerEtiquetteNomArret(***nomArret***,** tpEtiquetteNomArret**,** *estPoleEchange***)**

// Positionnement du point de pôle d’échange

Canvas.SetLeft(**pointArret**, *abscissePointArret*)

Canvas.SetTop(**pointArret**, *ordonneePointArret*)

// Positionnement de l’étiquette du nom du pôle d’échange

Canvas.SetLeft(**etiquetteNomArret**, *abscisseEtiquetteNomArret*)

Canvas.SetTop(**etiquetteNomArret**, *ordonneeEtiquetteNomArret*)

canvas.Children.Add(**pointArret**);

canvas.Children.Add(**etiquetteNomArret**);

**Si** *estCorrespondance* est Vrai **Alors**

{

**etiquetteCorrespondanceArret** 🡨 **CreerEtiquetteCorrespondance(**tpEtiquetteCorrespondance**,** *estPoleEchange***)**

// Positionnement de l’étiquette de correspondance du pôle d’échange

Canvas.SetLeft(**etiquetteCorrespondanceArret**, *abscisseEtiquetteCorrespondanceArret*)

Canvas.SetTop(**etiquetteCorrespondanceArret**, *ordonneeEtiquetteCorrespondanceArret*)

canvas.Children.Add(**etiquetteCorrespondanceArret**);

}

}

}

**ModeliserCorrespondance**(estPoleEchange : booléen,