GLO-2004 Génie logiciel orienté objet

Livrable 2 : MicrosofTears

présenté à

M. Jonathan Gaudreault

par

Équipe 52 — Équipe 52

matricule	*nom*	*signature*
AAJAL	536 767 347	Adulai Aliu Jalo
KEJOB2	536 777 791	Kevin Jobin
LUNIQ	111 267 600	Lucas Niquet
LYLEL	111 271 294	Lydia Lelièvre

Université Laval

12 octobre 2021

Historique des versions		
version	*date*	*description*
0.0	6 octobre 2021	création du document

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Diagramme de classe de conception

1.1 Diagramme de classe de conception

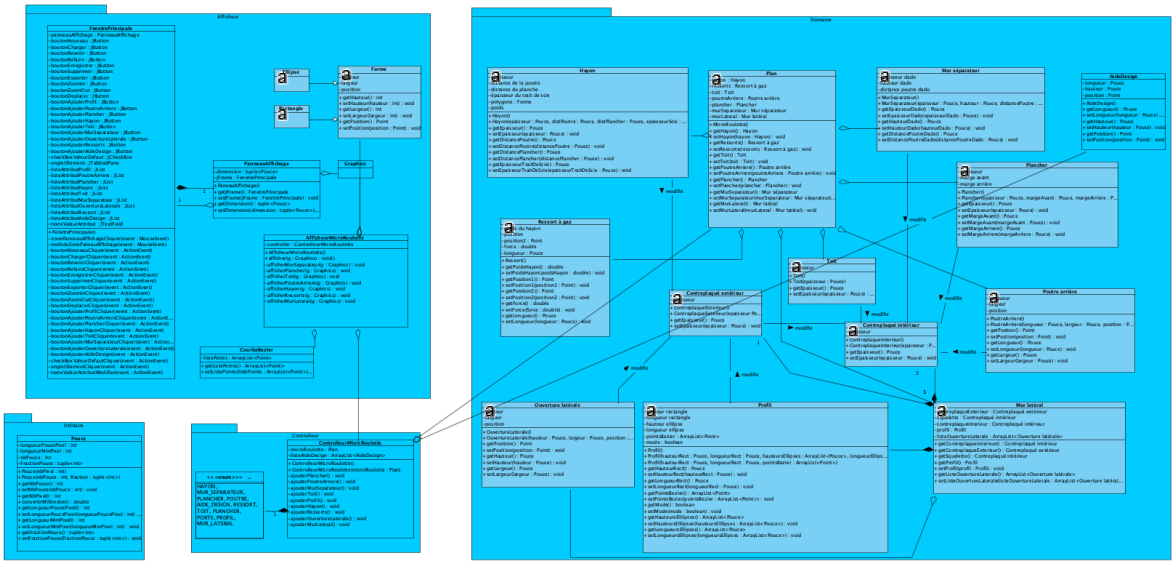

FIGURE 1.1 – Diagramme de classe de conception (complet)

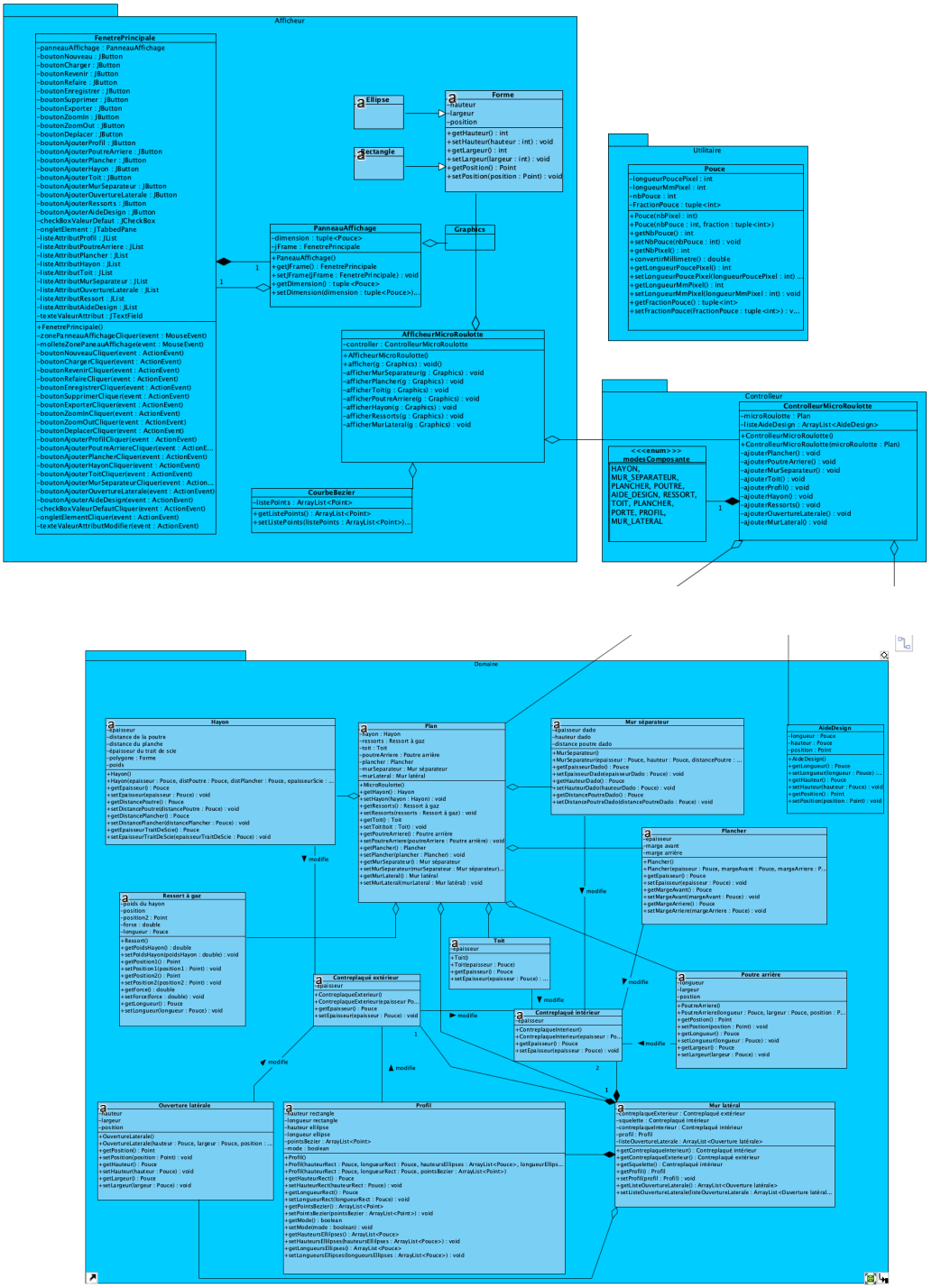

FIGURE 1.2 – Diagramme de classe de conception (détaillé)

1.2 Explication du diagramme de classe de conception

1.2.1 Classes du domaine

MicroRoulotte

Cette classe représente la structure de la microroulotte. La microroulotte a pour attribut un hayon, des ressorts, un toit, une poutre arrière, un plancher, un mur séparateur et un mur latéral. Cette classe hérite des attributs des classes **Hayon**, **Ressorts**, **Toit**, **PoutreArriere**, **Plancher**, **MurSeparateur** et **MurLateral**. Ces classes sont accessibles et modifiables à partir de *getters* et de *setters* qui sont implantés dans la classe **MicroRoulotte**. Le constructeur de cette classe ne prend aucun paramètre car les attributs sont déjà initialisés par défaut.

MurSeparateur

Cette classe représente le mur qui sépare la cuisine de la cabine. Elle a pour attribut l'épaisseur du dado, la hauteur du dado et la distance entre la poutre et le dado. Les mesures sont prises en pouces. Le constructeur de cette classe prend en paramètre l'épaisseur, la hauteur et la distance entre la poutre et le dado. Les attributs sont accessibles à l'aide de *setters* et de *getters*.

Plancher

Cette classe représente la configuration du plancher de la microroulotte. Elle a pour attribut l'épaisseur du plancher, la distance de la marge avant et la distance de la marge arrière. Les mesures sont prises en pouces. Le constructeur de cette classe prend en paramètre l'épaisseur, la marge avant et la marge arrière. Les attributs sont accessibles à l'aide de *setters* et de *getters*.

Toit

Cette classe représente la construction du toit. Elle a pour attribut l'épaisseur du toit en pouces. Le constructeur de la classe prend pour attributs la l'épaisseur qui est accessible à l'aide de *setters* et de *getters*.

ContreplaqueInterieur

Cette classe représente le contreplaqué intérieur de la microroulotte. Elle a pour attribut l'épaisseur du toit en pouces. Le constructeur de la classe prend pour attributs la l'épaisseur qui est accessible à l'aide de *setters* et de *getters*.

AideDesign

Cette classe représente l'aide au design de la microroulotte. Elle permet de positionner des rectangles représentant des objets quelconque. Ces retangles font figure d'illustration et guident le design. Cette classe a pour attribut la longueur, la hauteur et la position de l'objet. Les mesures sont prises en pouces. Le constructeur de la classe prend en parametre la longueur, la hauteur et la position de l'objet. Les attributs sont accessibles à l'aide de *setters* et de *getters*.

PoutreArriere

Cette classe représente la poutre arrière de la microroulotte. Elle a pour attribut la longueur, la largeur et la position. Les mesures sont prises en pouces. Le constructeur de la classe prend en paramètre la longueur, la largeur et la position. Les attributs sont accessibles à l'aide de *setters* et de *getters*.

MurLateral

Cette classe représente le mur latéral de la microroulotte. Elle hérite des classe **ContreplaqueInterieur**, **ContreplaqueExterieur**, **Profil** et **OuvertureLaterale**. Elle a pour attribut le contreplaqué extérieur qui hérite de la classe **ContreplaqueExterieur**, le squelette et le contreplaqué intérieur qui héritent de la classe **ContreplaqueInterieur**, le profil qui hérite de la classe **Profil** et une liste qui hérite de la classe **OuvertureLaterale** . Les attributs sont accessibles à l'aide de *setters* et de *getters*.

Profil

Cette classe représente le profil de la roulotte. Elle a pour attribut la hauteur du rectangle, la longueur d'un rectangle, la hauteur de l'ellipses, la longueur de l'ellipses, une liste des points de Bézier et mode. Elle a deux constructeurs. Le premier prend en paramètre la hauteur du rectangle, la longueur du rectangle, la hauteur et la longueur de l'ellipse. Le deuxième prend en paramètre la hauteur et la longueur du rectangle et la liste des points de Bézier. Les attributs sont accessibles à l'aide de *setters* et de *getters*.

ContreplaqueExterieur

Cette classe représente le contreplaqué extérieur de la microroulotte. Elle a pour attribut l'épaisseur du toit en pouce. Le constructeur de la classe prend pour attributs la l'épaisseur qui est accessible à l'aide de *setters* et de *getters*.

OuvertureLaterale

Cette classe représente la porte de la microroulotte. Elle a pour attribut la hauteur, la largeur et la position. Les mesures sont prises en pouces. Le constructeur de la classe prend en paramètre la hauteur, la largeur et la position. Les attributs sont accessibles à l'aide de

setters et de *getters*.

Ressorts

Cette classe represente les ressorts à gaz du hayon de la microroulotte. Elle a pour attributs le poids du hayon, position 1, position 2, la force et la longueur en pouce. Les attributs sont accessibles à l'aide de *setters* et de *getters*.

Hayon

Cette classe représente le hayon de la microroulotte. Elle a pour attribut l'épaisseur du hayon, la distance de la poutre, la distance du plancher et l'épaisseur du trait de scie. Les mesures sont prises en pouces. Le constructeur de la classe prend en paramètre l'épaisseur du hayon, la distance de la poutre, la distance du plancher et l'épaisseur du trait de scie. Les attributs sont accessibles à l'aide de *setters* et de *getters*.

1.2.2 Classes de l'afficheur

FenetrePrincipale

La **FenetrePrincipale** (JFrame) contient tous les boutons, les listes, les onglets, les zones de texte et le **PanneauAffichage** (JPanel) que nous allons utiliser sur notre interface.

Il y a d'abords l'attribut panneauAffichage qui hérite de la classe **PanneauAffichage** d'où l'aggregation. Il y a une fonction qui détectent lorsque la souris est cliquer sur le panneau d'affichage et une fonction qui détecte si la mollete est utilisé sur le panneau d'affichage.

Il y a les boutons suivant qui hérite de **JButton** : boutonNouveau, boutonCharger, boutonRevenir, boutonRefaire, boutonEnregistrer, boutonSupprimer, boutonExporter,

boutonZoomIn, boutonZoomOut, boutonDeplacer, boutonAjouterProfil, boutonAjouterPoutreArriere, boutonAjouterPlancher, boutonAjouterHayon, boutonAjouterToit, boutonAjouterMurSeparateur, boutonAjouterOuvertureLaterale, boutonAjouterRessort et boutonAjouterAideDesign. Tous ces boutons ont une fonction qui fait un évènement lorsque qu'ils sont pressé.

Pour chaque composante de la micro-roulotte, il y a une list avec tous les attributs de cette composante qui hérite de **JList** : listeAttributProfil, listeAttributPoutreArriere, listeAttributPlancher, listeAttributHayon, listeAttributToit, listeAttributMurSeparateur, listeAttributOuvertureLaterale, listeAttributRessorts, listeAttributAideDesign.

Il y a également un checkBox pour mettre la valeur par défaut ou non pour chaque attribut des composantes de la micro-roulotte qui hérite de **JCheckBox**.

Il y a un onglet pour choisir de quelle composante de la micro-roulotte, la liste d'attribut sera affiché. Celle-ci hérite de **JTabbedPane**.

Chaque attribut des composantes de la micro-roulotte ont une zone de texte appelé texteValeurAttribut que l'on peut modifier qui hérite de **JTextField**. Il y a un fonction qui fait un évènement lorsque le texte est modifié.

La **FenetrePrincipale** a un contructeur sans paramètre.

PanneauAffichage

Le panneau d'affichage a un attribut dimension qui est un tuple avec sa longueur et hauteur. Cette attribut a un getter et un setter.

Le **PanneauAffichage** doit avoir une **FenetrePrincipal** d'où la composition. Cette attribut a un getter et un setter.

Le **PanneauAffichage** a un constructeur sans paramètre. Cette classe utilise **Gra-**

phics.

CourbeBezier

Cette classe est utilisé pour représenter le profil de la micro-roulotte. Elle a comme attribut une ArrayList<Point>. Cette attribut a un getter et setter.

AfficheurMicroRoulotte

Cette classe permet de représenter graphiquement la micro-roulotte avec la classe **Graphics**. Elle a en attribut un **ControlleurMicroRoulotte** d'où l'aggregation. Cette classe a des fonctions privées pour afficher chaque composante de la micro-roulotte. Tout ces fonctions peuvent être appelées par la fonction public afficher.

1.2.3 Classes de l'utilitaire

Pouce

Cette classe représente les unités de mesures qui seront utilisées dans le logiciel. Elle contient les attributs du ratio pouce/pixel, le ratio millimetre/pixel, le nombre de pouce et la fraction de pouce. Le constructeur par défaut prend le nombre de pixel en paramètre et le constructeur prend le nombre de pouces et la fraction de pouce en paramètre. Les attributs sont accessibles à l'aide de *setters* et de *getters*. La méthode *convertirMilimetre* sert à la conversion d'une unité à une autre.

1.2.4 Controlleur

ControlleurMicroRoulotte

Cette classe représente le controlleur de Larman. Elle hérite des classes du domaine. La classe contient les attributs microroulotte et une liste aide design. Les méthodes de la classe ajoutent les éléments des classes du domaine. Le constructeur de la classe prend en paramètre microroulotte.

Chapitre 2

Architecture logique

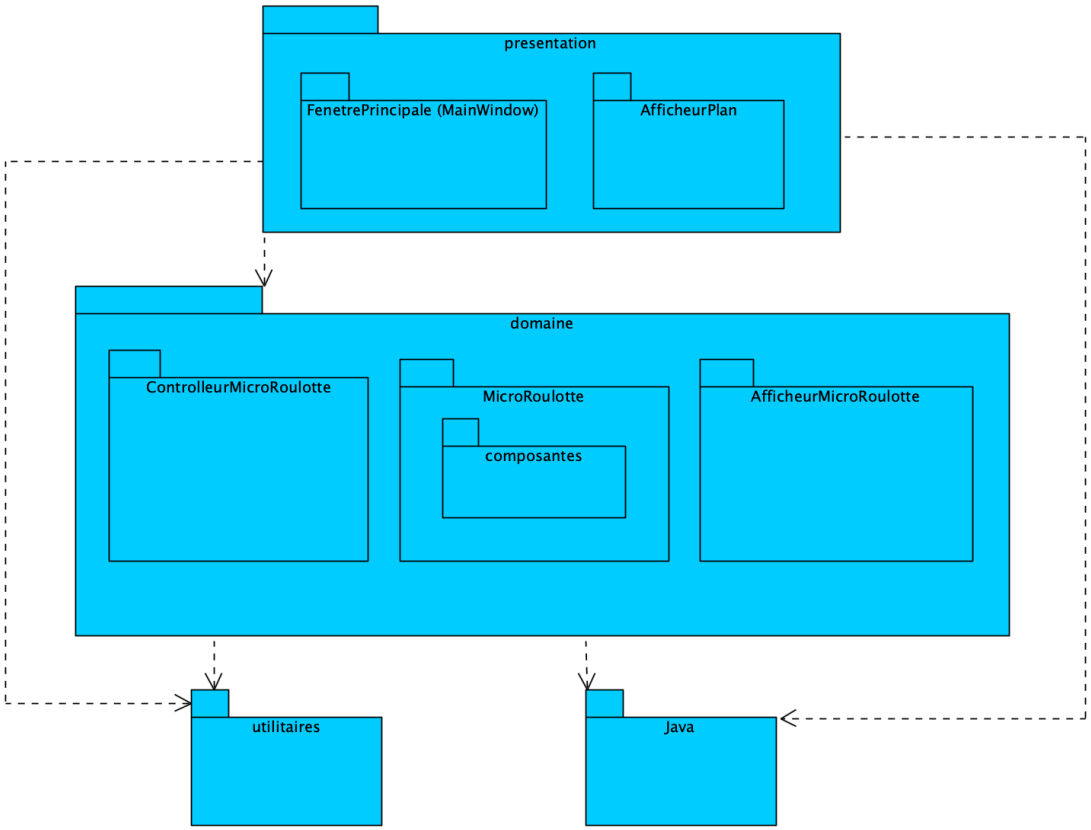

FIGURE 2.1 – Diagramme des packages

2.1 Explications

Notre architecture logique est décomposée en deux couches principales : la couche de présentation et la couche du domaine tel que décrit par Larmann. Ces deux couches utilisent les classes de la librarie Java (Swing et AWT) ainsi que des classes utilitaires (qui servent à la conversion des mesures, entre autres).

2.1.1 La couche presentation

La couche de présentation se charge de l'affichage. Elle contient la fenêtre principale (FenetrePrincipale) qui s'occupe de générer les différents éléments de l'interface (par exempel : boutons, textes, etc.). Tous ces éléments sont des instances des classes appartenant à la librairie "Swing" de JavaX. L'ensemble des évènements sont gérés d'abord par la fenêtre principale qui s'occupe d'acheminer l'information au niveau de la couche domaine ou d'appeler l'Afficheur pour rafraichir l'affichage suite à un événement.

2.1.2 La couche domaine

La couche du domaine contient l'ensemble de la logique de l'application. Elle commence par notre ControlleurMicroRoulotte (contrôleur de Larman), dont la tâche principale est de recevoir les appels provenant de la couche de présentation. Le contrôleur achemine par la suite ces appels vers les classes et entitées du domaine (les objects réels de notre modèle-objet). Finalement, il a aussi la tâche de retourner les information s(et non l'objet complet) nécessaires à la couche de présentation.

Chapitre 3

Diagrammes de séquences de conception

3.1 Diagrammes de séquences de conception

3.1.1 Déterminer l'élément sélectionné lors d'un clic de souris dans la vue en 2D (3.1.1)

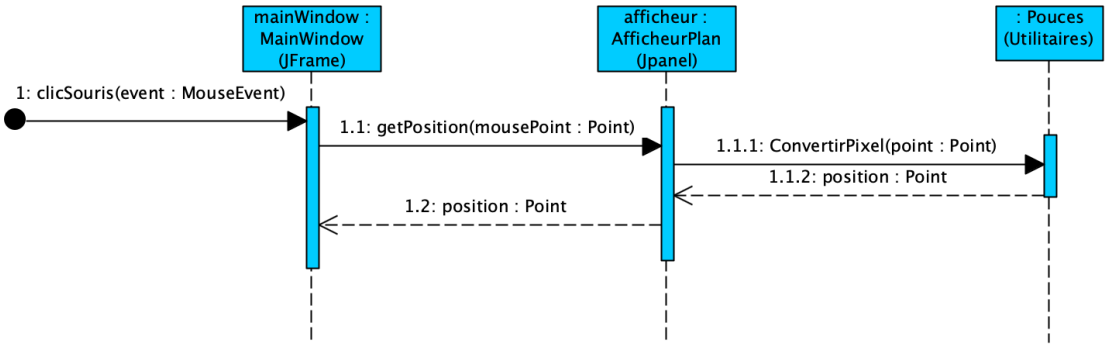

FIGURE 3.1 – Diagramme 3.1.1

Explication (3.1.1)

Déterminer l'élément sélectionné lors d'un clic de souris dans la vue en 2D

Cette séquence démarre lorsque la fenêtre principale (MainWindow) reçoit l'évènement MousePressed (clicSouris).Ensuite la fenêtre principale fait appel à l'afficheur (afficheurPlan) par l'entremise de la méthode getPosition, l'AfficheurPlan se chargera d'appeler la classe Pouces par la méthode ConvertirPixel, cette classe est chargée de convertir les coordonnés du clic de la souris en coordonnées du plan (en unités de mesure selon l'unité de mesure choisie par l'utilisateur). Celle-ci est stockée à l'interne dans l'afficheur qui le passe en paramètre à la classe Pouce.

3.1.2 Déterminer l'élément sélectionné lors d'un clic de souris dans la vue en 2D (3.1.2)

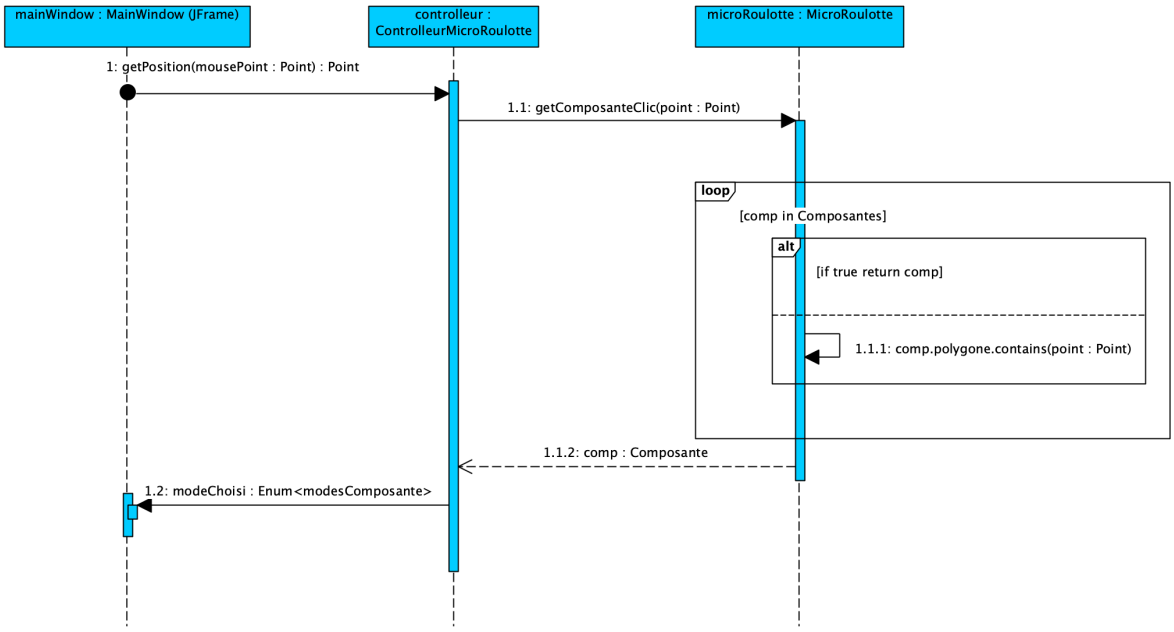

FIGURE 3.2 – Diagramme 3.1.2

Explication (3.1.2)

Déterminer l'élément sélectionné lors d'un clic de souris dans la vue en 2D

La seconde séquence débute par un appel au contrôleur auquel on passe les coordonnées (en unités de mesures) du point sur lequel l'utilisateur a cliqué. Par la suite, le contrôleur fait appel à la classe MicroRoulotte par l'entremmise de la méthode getComposanteClic afin que celle-ci boucle sur sa liste de composantes et teste, pour chacune des composantes présentes dans le plan, si le polygone représentant l'élément contient le point. Si oui, la boucle s'arrête et retourne l'élément. Le contrôlleur pourra ensuite renvoyé à la fenêtre principale le modeChoisi (Enum représentant la composante sur laquelle l'utilisateur a cliqué).

3.1.3 Création d'une porte

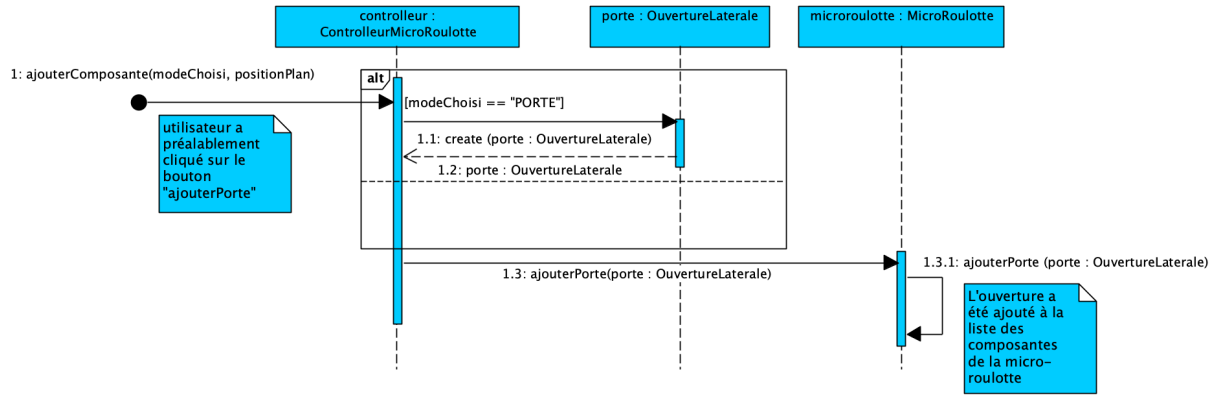

FIGURE 3.3 – Diagramme 3.2

Explication (3.2)

Création d'une porte

La création d'une porte débute lorsque la fenêtre principale (MainWindow) reçoit un clique sur le bouton *ajouterPorte*, puis appelle le ControlleurMicroRoulotte en lui fournissant le modeComposante choisi (la composante choisie) ainsi que la position réelle (déjà convertie en unité de mesure). Par la suite le contrôleur fait appel à la classe OuvertureLaterale pour qu'elle crée un objet OuvertureLaterale (la porte) avec les paramètres par défaut. Par la suite, le contrôleur reçoit cet objet et appelle la méthode *ajouterPorte* de la classe MicroRoulotte pour qu'elle l'ajoute à sa liste de composantes.

3.1.4 Réalisation de l'affichage de la vue 2D

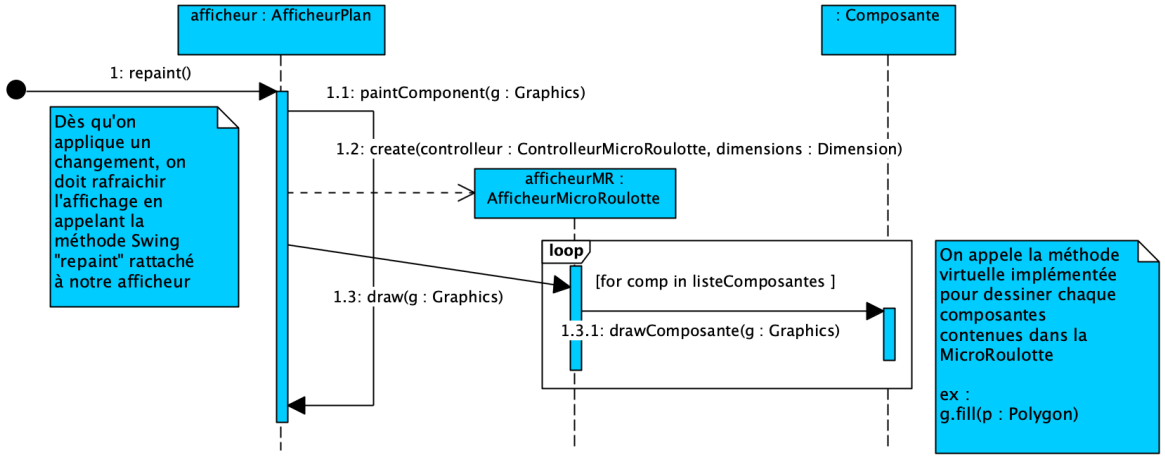

FIGURE 3.4 – Diagramme 3.2

Explication (3.2)

Réalisation de l'affichage de la vue 2D

L'affichage de la vue 2D débute après avoir appliqué un changement qui nécessite un rafraichissement (par exemple l'ajout d'une composante). L'afficheur principal (AfficheurPlan) par l'entremise de la méthode *repaint()* appelle automatiquement sa méthode héritée du JPanel *paintComponent*. C'est dans celle-ci qu'on fait appel au constructeur de l'AfficheurMicroRoulette afin de redessiner la micro-roulotte et ses composantes. Ainsi, l'AfficheurMicroRoulotte va boucler sur les composantes de la liste de composantes de la microroulotte et se charger d'appeler leurs méthodes de dessin implémentées *drawComposante*. C'est une méthode que chaque composante doit implémenter (méthode virtuelle pure).

Chapitre 4

Pseudo-code d'un algorithme

4.1 Algorithme 1

1. Creer le polygone P à l'intersection des deux formes

2. Ajouter tous les points de l'ellipse qui est à l'intérieur du rectangle.

3. Ajouter tous les points du rectangle à l'intérieur de l'éllipse

4. Mettre ces points dans l'ordre antihoraire

```
    // Trouver les points du contour de l'ellipse
debut fonction(ellipse, rectangle)
listeRectangle = rectangle.getPoints
for (int i = 0 ; i <= 360 ; i++)
double x, y ;
x = ellipse.getHeight/2 * Math.round(Math.sin(Math.toRadians(i)))+ellipse.getCenterX ;
y = ellipse.getWidth/2 * Math.round(Math.cos(Math.toRadians(i)))+ellipse.getCenterY ;
listPoint.add(x,y) ;
```

```
// Exemple coin haut-gauche, trouver l'intersection avec la ligne horizontale
for point in listPoint
if(point.getY==-rectangle.getHeigth/2+getCenterY)
if(listPoint2.isEmpty)
listPoint2.add(point)
elsif(point.getX<listPoints2.get(0).getX)
listPoint2.remove(0)
listPoint2.add(point)

    // Trouver le points d'intersection avec la ligne vertical gauche du rectangle le plus haut
for point in listPoint
if(point.getY==-rectangle.getWidth/2+getCenterX)
if(listPoint2.isEmpty)
listPoint2.add(point)
elsif(point.getY<listPoints2.get(1).getY)
listPoint2.remove(1)
listPoint2.add(point)

    // On a donc les 2 points que le coin gauche est coupé
position1=listPoint.indexOf(listPoint2.get(0))
position2=listPoint.indexOf(listPoint2.get(1))

    for(int i = position1+1 ; i<position2 ; i++)
listCouper.add(listPoint(i)) ;
```

position=ListeRectangle.remove(Point entre listPoint2.get(0) AND listPoint2.get(1))
ListeRectangle.add(position,ListeCouper)

fin fonction

Chapitre 5

Diagramme de Gantt modifié

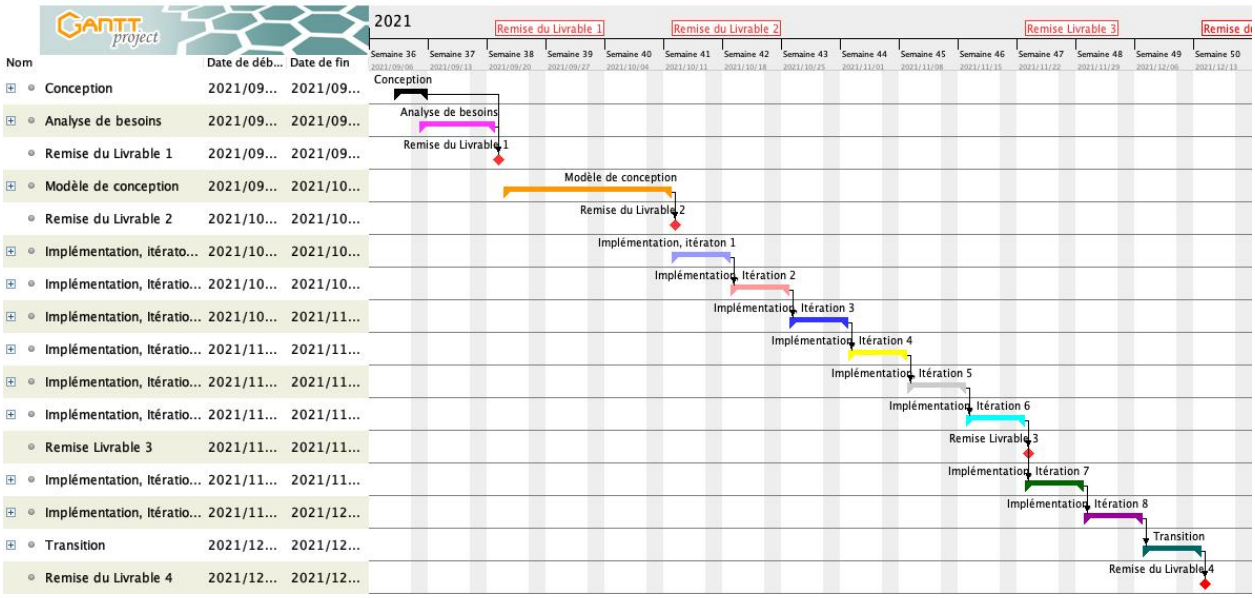

FIGURE 5.1 – Diagramme de Gantt modifié

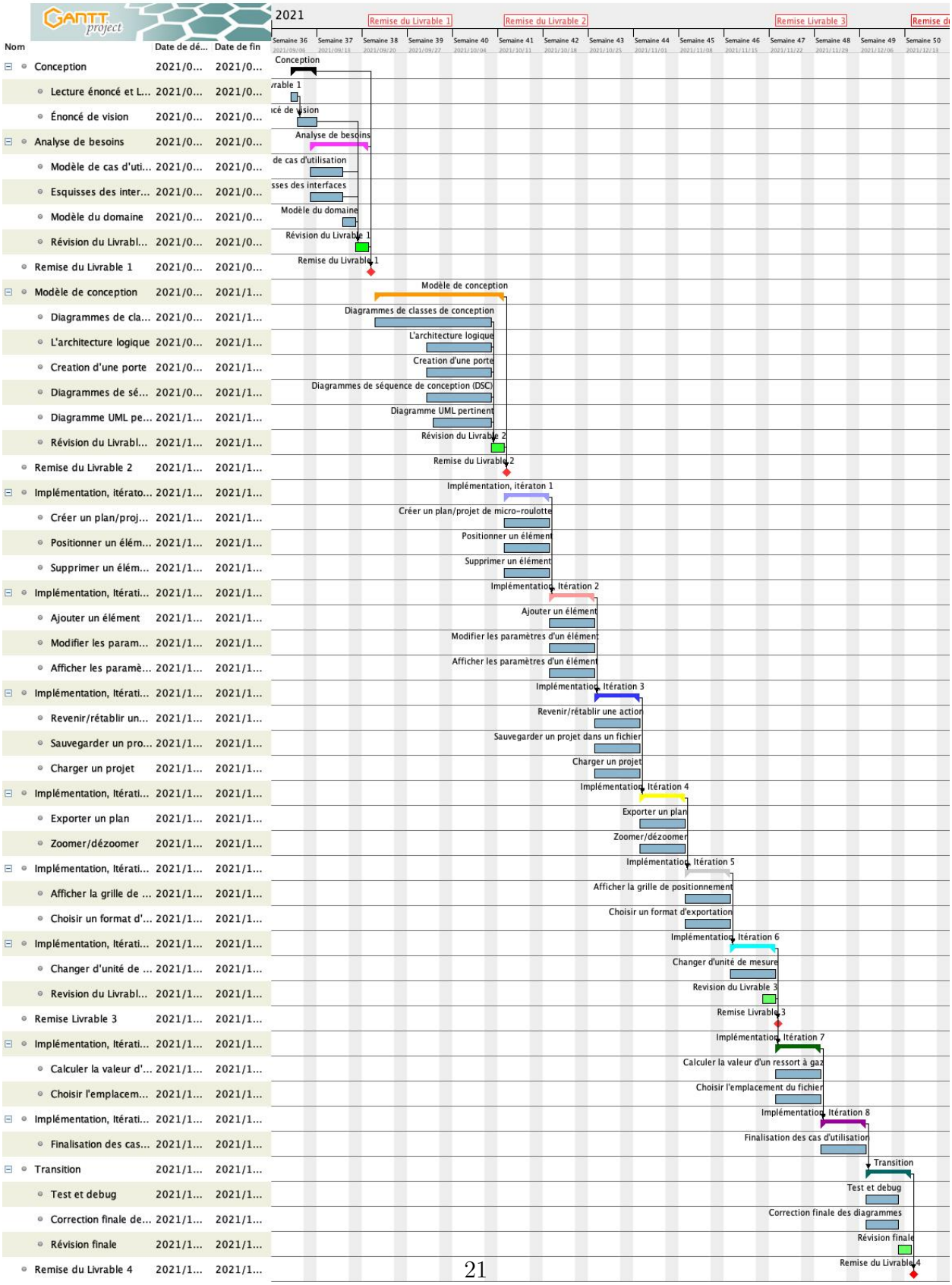

FIGURE 5.2 – Diagramme de Gantt modifié version longue

Chapitre 6

Contribution des membres de l'équipe

Adulai : text explicatif pour les diagrammes de séquences et pseudo-code

Lydia : Text explicatif pour les classes du domaine des classes de conception et le diagramme de Gantt

Lucas : J'ai travaillé sur le diagramme de classe de conception, le texte des classes de l'interface, le pseudo code du profil et j'ai participé aux rencontres d'équipe.

Kevin : Interface graphique (code), diagrammes de sequences de conception, texte explicatif des diagramme de séquence de conception, mise en page du rapport, participation au diagramme de classe de conceptin, participation au pseudo-code d'algorithme