Architecture & Design Pattern :

Dans le cadre de la réalisation de la vue liée à notre modèle MVC, nous avons choisi de l’implémenter en plusieurs éléments, tous contenus dans l’unique package vue.

Pour partir de l’utilisateur, nous avons tout d’abord la classe « InterfaceGUI » qui se charge de la création de la fenêtre et du menu avec ses boutons. L’import de la classe du contrôleur permet ensuite de réagir aux actions de l’utilisateur sur ces boutons et de lancer les appels de méthodes correspondantes. Cette classe se charge aussi de gérer la désactivation des boutons grâce à des méthodes que le contrôleur se chargera ensuite d’utiliser. La saisie des différentes informations comme le nombre de livreurs et la durée d’une livraison est aussi mise en place au sein de cette classe qui se chargera de transmettre ces informations au contrôleur.

Ensuite, le reste de l’affichage est divisé entre deux classes héritant toutes deux de « Vue », qui leur permet de manipuler les « Observer » avec une méthode « update » et organise leur structure commune pour contenir une liste de tournées, un objet planification pour l’appel des éléments du modèle, un plan et une demande de livraisons. De ces deux classes, la première, « VueTextuelle » se charge de remplir et de permettre à l’utilisateur d’interagir avec le tableau sur le côté droit de l’écran alors que la seconde, « VueGraphique », comporte l’affichage concret du plan avec la demande de livraisons, les tournées et les modifications de celles-ci. Le dessin des tournées à proprement parlé se fait via les structures de données que nous avons choisies en utilisant des équivalences pour chaque élément.

Les rues seront manipulées par une classe « LineModifiee » qui se chargera d’affecter à une « Line » classique une couleur et un nom. Cette classe permettra également à l’utilisateur de sélectionner une rue via une mécanisme d’évènements ainsi que de changer ou d’obtenir le nom de la rue voulue.

Pour la représentation des livraisons sur l’affichage, nous avons utilisé la classe « CercleLivraison » qui nous permet, avec des attributs tels que l’heure de passage, la livraison associée et l’ordre de passage, de gérer cet aspect de l’application. En effet, des méthodes nous permettent de déterminer l’ordre de passage de cette livraison dans sa tournée et d’obtenir la livraison associée tandis que d’autres nous permettent de fixer la couleur et l’heure de passage correspondant à ce point du plan.

Et enfin, pour les nœuds du graphe et donc plus concrètement les intersections entre les rues, nous avons utilisés la classe « CercleIntersection » qui permet de joindre l’identifiant du nœud au cercle en question. En plus d’une méthode pour obtenir cet identifiant, cette classe dispose aussi d’une méthode pour changer l’affichage d’un nœud quand il est sélectionné au niveau de sa taille et de sa couleur.