Interfaces Graphiques
TPs 4, 5, 6 et 7 - Architecture MVC - Twisk

TP5

L'objectif de cette séance est d'ajouter les points de contrôle sur les activités et de donner la possibilité à l'utilisateur de relier deux points de contrôle par un arc.

Au fur et à mesure de la réalisation des questions ci-dessous, vous devez compléter le diagramme de classes fourni à la séance précédente. Cela vous sera utile pour conserver une vue d'ensemble du projet.

1- Définir et afficher les points de contrôle

a) Dans le modèle, la classe **twisk.mondeIG.PointDeControleIG** décrit un point de contrôle, défini par la position de son centre, un identifiant unique (une chaîne de caractères) et l'étape à laquelle il est rattaché. Lors de la création d'une activité, quatre points de contrôle lui sont attribués, mémorisés (dans une collection) dans **EtapeIG**. Ces points de contrôle sont placés sur les milieux des quatre côtés.

La classe **EtapeIG** devient itérable : l'itérateur permet de parcourir tous les points de contrôle.

Écrivez et testez la classe **PointDeControleIG**.

b) La classe **VuePointDeControleIG** est une vue de **PointDeControleIG**, elle hérite de **Circle**. Écrivez cette classe.

Les points de contrôle sont affichés lors de l'exécution de la fonction **reagir** de **VueMondeIG.**

Complétez la fonction **reagir** de **VueMondeIG** pour afficher les points de contrôle de toutes les étapes.

2- Définir et afficher les arcs

a) Dans le modèle, la classe **twisk.mondeIG.ArcIG** décrit un arc défini par deux points de contrôle. Écrivez et testez la classe **ArcIG**.

La classe **MondeIG** mémorise tous les arcs. Dans cette classe, ajoutez une fonction **ajouter(PointDeControleIG pt1, PointDeControleIG pt2)** qui permet d'ajouter un arc, créé à partir des deux points de contrôle passés en paramètre. Ajoutez également un nouvel itérateur dans la classe **MondeIG**, pour consulter tous les arcs.

b) Pour créer un chemin entre deux activités, l'utilisateur doit cliquer sur deux points de contrôle.

Ajoutez un écouteur sur **VuePointDeControleIG** qui prévient le monde qu'un point de contrôle est sélectionné. Le monde ajoute un arc, dès lors que deux points ont été sélectionnés. Dans un premier temps, vous pouvez vous contenter de créer tous les arcs demandés par l'utilisateur.

c) Sur le dessin, un arc est une ligne droite (instance de **Line**) qui se termine par un triangle (instance de **PolyLine**) pour dessiner la tête de la flèche. Si vous avez quelques lacunes en géométrie, un peu de recherche sur Internet vous permettra de trouver le calcul des coordonnées des trois points du triangle.

La classe **VueArcIG** est une vue de **ArcIG**, elle hérite de **Pane**, pour pouvoir y intégrer deux composants (une **Line** et une **PolyLine**). Écrivez cette classe. Complétez la fonction **reagir** de **VueMondeIG** pour afficher les arcs. Attention, il faut impérativement afficher les arcs avant les activités ; dans le cas contraire, les écouteurs ne sont plus toujours opérationnels. Pensez à mettre du style sur les arcs tracés.

3- Vérifier la validité des arcs

Certains arcs ne sont pas valides. Faites la liste des contraintes qui doivent être respectées et vérifiez ces contraintes dans la fonction qui ajoute un arc (dans le monde). Si l'une d'elles n'est pas satisfaite, l'arc n'est pas ajouté et un message est affiché sur la sortie standard uniquement.

Dans le contexte de ce tp IG, vous pouvez ignorer la détection de circuit, qui sera faite dans le cadre du projet Twisk.

Complétez les tests pour vérifier que les contraintes sont correctement traitées.

4- Affichage de messages d'erreurs

Lorsque les contraintes ne sont pas respectées rendant impossible l'ajout d'un arc, il serait profitable pour l'utilisateur que s'affiche une fenêtre avec un message d'erreur approprié (au lieu d'afficher un message d'erreur sur la sortie standard).

Une solution consiste à déclencher une exception dans le modèle lors d'un ajout d'arc impossible ; cette exception est capturée dans l'écouteur à l'origine de la demande ; l'écouteur peut alors afficher une boîte de dialogue pour prévenir l'utilisateur. Pour gérer simplement les différentes erreurs, il est pratique de faire des exceptions spécifiques à Twisk, c'est-à-dire des sous-classes de **twisk.exceptions.TwiskException**, elle-même sous-classe de **Exception**.

Ce type d'affichage utilise une boîte de dialogue modale (**Alert**, par exemple) instanciée dans l'écouteur à l'origine du problème. Attention, on n'instancie jamais ces composants dans le modèle, qui doit rester indépendant de l'interface graphique. Une boîte modale nécessite une action de la part de l'utilisateur (par exemple, cliquer sur Ok). Toutefois, il pratique de fermer automatiquement cette boîte au bout d'un certain temps, en utilisant **PauseTransition** ; c'est un chronomètre à qui on peut attacher un écouteur qui est réveillé lorsque le temps imparti est écoulé.

5- (option pour ceux qui avancent vite et bien) Deux types d'arcs

Dans certains cas, le tracé en ligne droite entre deux points de contrôle ne donne pas un résultat visuel satisfaisant. Il faut donner à l'utilisateur la possibilité de dessiner soit des lignes (instances de **Line**), soit des courbes (instances de **CubicCurve**). Ces courbes sont construites à partir de plusieurs points de contrôle intermédiaires entre le point de départ et le point d'arrivée : ces points sont sélectionnés par l'utilisateur, mais pas affichés ; ils ne seront pas modifiables, même si l'étape est déplacée.

Pour intégrer cette nouvelle fonctionnalité, il faut rendre la classe **ArcIG** abstraite, en faire deux sous-classes **LigneDroiteIG** (avec le code de **ArcIG** de la question précédente) et **CourbeIG**. De même il faut rendre la classe **VueArcIG** abstraite, en faire deux sous-classes **VueLigneDroiteIG** et **VueCourbeIG**.

Par ailleurs, l'écouteur de clic de **VuePointDeControleIG** n'est plus valide, puisqu'il est maintenant possible de cliquer n'importe où sur le dessin. Il faut donc enlever cet écouteur et le déplacer dans **VueMondeIG**. Cet écouteur transmet au monde les coordonnées de tous les points cliqués ; si le premier et le deuxième points sélectionnés sont des points de contrôle, le monde crée une **LigneDroiteIG** ; si le premier et le quatrième sont des points de contrôle, le monde crée une **CourbeIG**.