TP 5 : Visualiser le modèle d'usine robotisée grâce aux interfaces

Dominique Blouin, Télécom Paris, Institut Polytechnique de paris
dominique.blouin@telecom-paris.fr

Au cours du TP précédent, nous avons modélisé la **structure** de l'usine robotisée pour laquelle nous voulons développer un simulateur. Nous avons donc créé un **modèle objet** de l'usine. Au cours de ce TP, nous allons modifier ce modèle afin de permettre de le visualiser grâce à une interface graphique qui vous est fournie. Pour cela, il faudra que votre modèle implémente les **interfaces Java** fournies par cette interface graphique.

Cette interface graphique est très simple ; la seule chose qu'elle sait faire, c'est d'afficher des **figures géométriques** de deux dimensions (2D) de forme rectangulaire, ovale ou polygonale, de différentes couleurs, et de différents types de traits (tirets et épaisseur) dans un canevas de dessin.

Tel que vu en classe, une interface définie un **point de vue** ou un **descriptif de propriétés** sur les classes qui l'implémentent. Ainsi, l'interface graphique fournie des interfaces Java spécifiant un simple point de vue **figures 2D**. Chaque composant de l'usine robotisée peut être en effet représenté par une figure 2D dans un canevas. Ainsi, il suffira que les classes de votre modèle **implémente** les interfaces figures 2D fournies par l'interface graphique pour que celle-ci puisse les afficher.

Ce jeu d'interfaces fournies par l'interface graphique constitue donc un **contrat** entre le modèle et l'interface graphique devant afficher le modèle. Cette manière d'intégrer différents composants logiciels est classique en programmation.

L'interface graphique de visualisation de figures

Télécharger la bibliothèque de l'interface graphique de visualisation de figures (canvas viewer) qui se trouve ici. Cette bibliothèque, nommée *canvas-viewer.jar*, n'est en réalité qu'un ensemble de classes Java compilées (fichiers *.class*) qui peuvent être utilisées par d'autres applications Java.

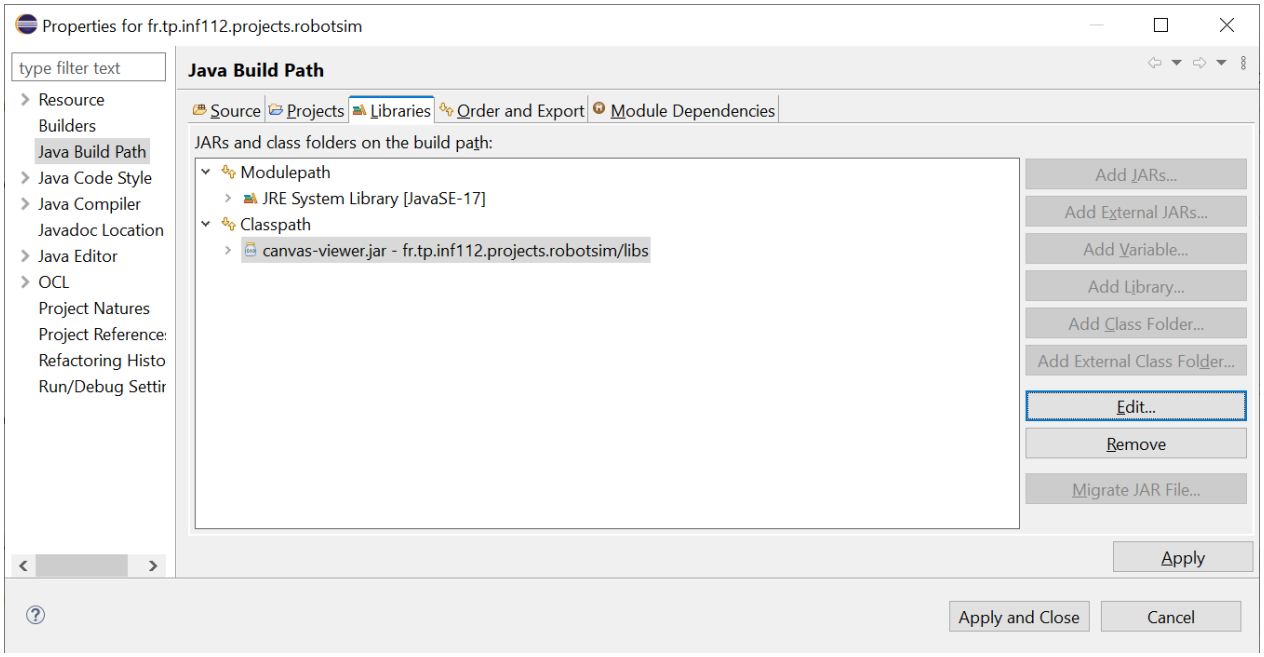
Ouvrir le fichier *canvas-viewer.jar* avec un utilitaire de compression de fichier tel que 7-Zip. Vous constaterez qu'il ne s'agit que d'une simple archive contenant les fichiers **.class* dans la même arborescence de répertoire que celle des packages des classes. Vous y trouverez également des fichiers **.java*, ce qui vous permettra de voir le code source de l'interface graphique, et même d'y mettre des points d'arrêt si vous souhaitez comprendre ce qui s'y passe en utilisant un débogueur…

Configurer le projet du simulateur pour utiliser la bibliothèque de l'interface graphique

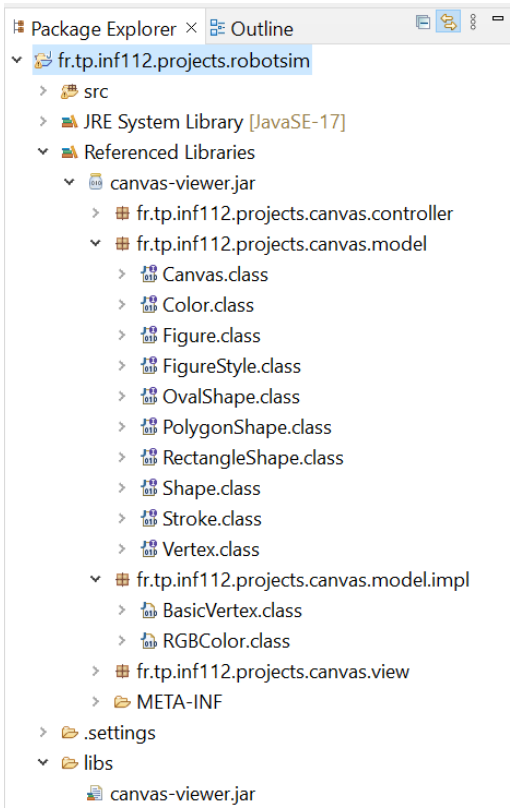
Dans le navigateur de projet ou de package dans Eclipse, sélectionner le projet de votre simulateur, puis faire un clic droit et cliquer sur le menu *New>>Folder* et créer un répertoire nomme *libs*. Déplacez le fichier *canvas-viewer.jar* dans ce répertoire (vous pouvez d'ailleurs

directement déplacer le fichier à partir d'un navigateur de fichier de votre système d'exploitation vers le répertoire du navigateur de projet dans Eclipse).

Sélectionner le projet de votre simulateur, puis faire un clic droit et cliquer sur le menu *Properties*. Dans la boîte de dialogue qui s'affiche, sélectionner la branche *Java Build Path* puis l'onglet *Libraries* dans la partie de droite de la fenêtre. Sélectionner *Classpath* puis cliquer sur le bouton *Add Jars…* et naviguer vers le fichier *canvas-viewer.jar* que vous avez ajouté dans le répertoire *libs*.



Dans le package explorer (ou le navigateur de projet), déplier la branche *Referenced Libraries*. Vous y verrez la bibliothèque *canvas-viewer.jar* que vous venez d'ajouter au projet. Vous verrez également les packages fournis par cette bibliothèque. Ouvrir les package *fr.tp.inf112.projects.canvas.model* et *fr.tp.inf112.projects.canvas.model.impl*.



Examiner le contenu de ces packages. Dans *fr.tp.inf112.projects.canvas.model*, vous trouverez des interfaces telles que *Canvas, Figure, Style* et *Shape*.

L'interface *Canvas* décrit les propriétés du canevas telles que son nom, sa largeur et sa hauteur, ainsi qu'une collection de figures à afficher dans le canevas.

L'interface *Figure* décrit les propriétés d'une figure 2D telles que son nom, sa position dans le cannevas (coordonnées x et y), son style, qui caractérise la couleur et le contour de la figure, et la forme de la figure telle que décrite par l'interface *Shape*. L'interface graphique ne peut afficher que trois types de formes dont les propriétés sont décrites par les sous interfaces *RectangleShape*, *OvalShape* et *PolygonShape*.

Afin de faciliter leur utilisation, toutes ces interfaces contiennent de la documentation sous format Javadoc qui explique les différentes méthodes requises par les interfaces.

Dans le package *fr.tp.inf112.projects.canvas.view* se trouvent les classes qui réalisent l'interface graphique qui servira à visualiser votre modèle d'usine robotisée. Il n'est pas nécessaire de comprendre ces classes, les interfaces graphiques n'étant pas au programme du cours.

Implémenter les interfaces du canevas avec le modèle d'usine robotisée

Pour afficher votre modèle dans l'interface graphique, vous devez déterminer les interfaces que devront implémenter les classes de votre modèle d'usine robotisée. Par exemple, l'usine, qui consiste en un plan dans lequel sont positionnés les composants de l'usine peut être vue comme un *canvas*. De la même manière, tout composant de l'usine robotisée peut être

représenté par une figure géométrique 2D. Ainsi, votre classe abstraite *Component* devra implémenter l'interface *Figure* :

```
public abstract class Component implements Figure
```

Nom et position de la figure dans le canevas

L'interface *Figure* demande de connaître un nom (méthode *getName()*) pour la figure, qui sera affiché sur le coin gauche en haut de la figure dans le canevas. Elle demande également les coordonnées de la figure afin de savoir où la dessiner dans le canevas.

Style de la figure

L'interface *Figure* nécessite également de connaitre le style d'affichage de la figure. Celui-ci est caractérisé par une couleur de fond pour la figure (méthode *getBackgroundColor()*), et les caractéristiques du trait du contour de la figure (épaisseur, pointillé, couleur, tels que décrits par l'interface *Stroke*).

Forme de la figure

Tel que mentionné précédemment, l'interface graphique ne sait dessiner que des figures de formes rectangulaire (*RectangleShape*), ovale (*OvalShape*) ou polygonale (*PolygonShape*). Pour chaque sous-classe concrète de la classe *Component*, il faut donc déterminer la forme souhaitée et définir des classes d'implémentation pour ces formes. Puis, chaque composant devra définir la méthode *getShape()* de l'interface *Figure* pour retourner une instance de l'implémentation correspondant à la forme choisie pour le composant.

Par exemple, puisque les robots de l'usine sont de forme circulaire, votre classe Robot retournera une instance d'une classe d'implémentation de *OvalShape*, tandis que les classes *Room* ou *Area*, correspondant à des composants de forme rectangulaires, retourneront des instances de classe d'implémentation de *RectangleShape*.

Faites de même pour chaque classe composant de votre modèle devant être visualisé par l'interface graphique à l'exception de la classe représentant l'usine robotisée, qui elle sera considérée comme étant de forme rectangulaire.

Le canevas

En effet, cette classe devra plutôt implémenter l'interface *Canvas*, qui représente le conteneur des figures. Notez que l'interface *Canvas* spécifie une méthode *Collection<Figure> getFigures()*. Puisque tous les composants de l'usine implémentent l'interface *Figure*, il sera possible de directement retourner l'attribut de votre classe Factory qui contient les composants. Cependant, il se peut que vous ayez à transtyper cette référence comme suit :

```
public Collection<Figure> getFigures() {
    return (Collection) components;
}
```

Lancement de l'interface graphique

Examiner la classe *CanvasViewer* du package *fr.tp.inf112.projects.canvas.view* de la bibliothèque de l'interface graphique (fichier *canvas-viewer.jar*). L'un des constructeurs de

cette classe prend comme unique paramètre une instance de classe implémentant l'interface *Canvas*. C'est ce constructeur que vous utiliserez pour lancer l'interface graphique.

A cette fin, créer une classe nommée *SimulatorApplication* afin de représenter l'application du simulateur reliant le modèle et l'interface graphique. Dans cette classe, créer une méthode *main*. Dans cette méthode, instancier un modèle d'usine tel que celui du TP précédent (contenant 3 salles ayant chacune une aire de travail, contenant chacune une machine de production, ainsi que 3 robots et une station de recharge).

Puis, instancier la classe *CanvasViewer* fournie par l'interface graphique en lui fournissant l'usine que vous venez de créer. Lancer la méthode main. Vérifier que votre modèle s'affiche correctement dans l'interface graphique tel qu'illustré par la figure suivante.

Noter qu'aucune des actions définies par les menus de l'interface graphique ne fonctionnera (elles seront désactivées), mis à part l'action de quitter l'application. Ces actions seront définies au prochain TP, lorsque vous implémenterez l'interface contrôleur du design pattern modèle-vue-contrôleur, que nous verrons en classe.