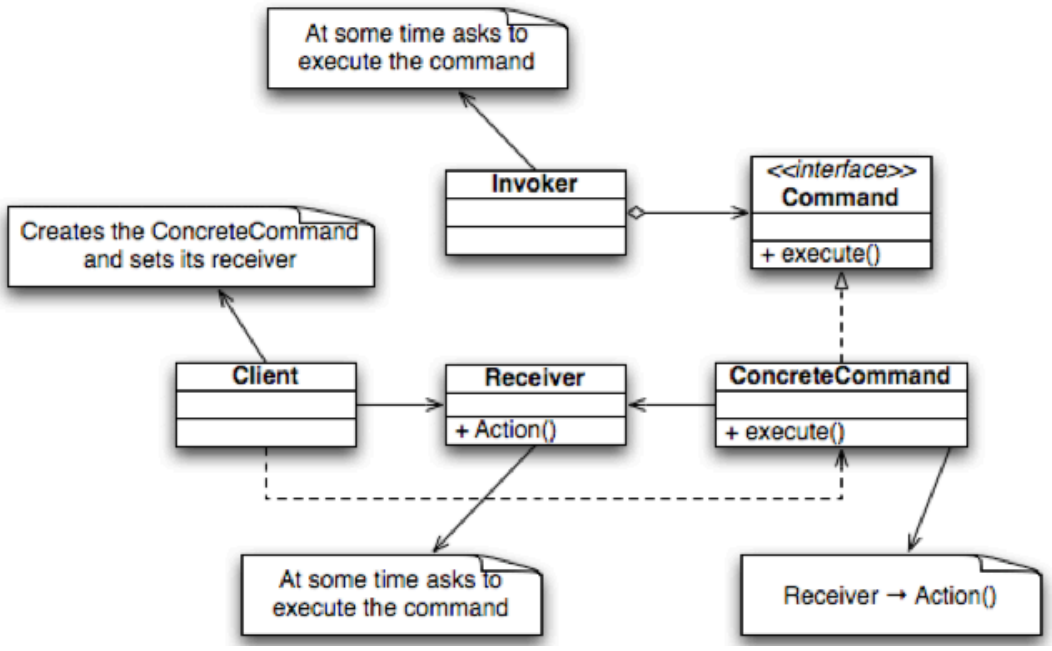
Vous devez compléter le système modélisé lors des deux premières séances de TD afin de prendre en compte les événements provenant du clavier :
- Quand l'utilisateur appuie sur les touches h, b, g, ou d : si un élément est sélectionné, alors cet élément est translaté de 1 cm vers le haut, le bas, la gauche ou la droite en fonction de la touche appuyée. Si aucun élément n'est sélectionné, alors tous les éléments composant l'objet géométrique sont translatés. Dans tous les cas, la vue est redessinée.
- Quand l'utilisateur appuie sur la touche a : la dernière translation effectuée est annulée et la vue est redessinée.

Votre travail : Lors de la séance d'aujourd'hui, vous devez :
- dessiner le diagramme de séquence associé à la frappe d'un caractère au clavier ;
- compléter en conséquence le diagramme de classes élaboré lors des séances précédentes (cf page suivante) ;
- Dessiner le diagramme de paquetages définissant l'architecture logique du système.

Vous utiliserez l'API Java *Swing* pour détecter les événements en provenance du clavier. Vous utiliserez le design pattern *Command* pour mémoriser les translations effectuées et permettre leur annulation.

Détection d'événements en Java : Pour détecter les événements en provenance du clavier, il faut implémenter des méthodes déclarées dans l'interface *KeyListener* de l'API *Swing*, et notamment la méthode *keyPressed(k : KeyEvent)* qui est appelée à chaque fois que l'utilisateur appuie sur une touche. La méthode *getKeyChar()* de la classe *KeyEvent* retourne le caractère appuyé.

Design pattern *command* : Afin de pouvoir annuler la dernière translation effectuée, vous mémoriserez les translations en utilisant le design pattern *Command* décrit de la façon suivante sur Wikipedia :



> *« The client instantiates the command object and provides the information required to call the method at a later time. The invoker decides when the method should be called. The receiver is an instance of the class that contains the method's code. If all user actions in a program are implemented as command objects, the program can keep a stack of the most recently executed commands. When the user wants to undo a command, the program simply pops the most recent command object and executes its undo() method. »*

Dans notre cas, les rôles de *Invoker* et *Client* sont joués par la classe *Contrôleur*, tandis que le rôle de *Receiver* est joué par la classe *EltGéométrique* (s'il y a un élément sélectionné) ou par la classe *ObjetGéométrique* (sinon). Plus précisément, pour effectuer une translation le contrôleur créera une commande dont le destinataire sera soit l'élément géométrique sélectionné soit l'objet géométrique. Cette commande sera mémorisée puis exécutée. Quand l'utilisateur appuiera sur la touche a, la dernière commande mémorisée sera défaite.

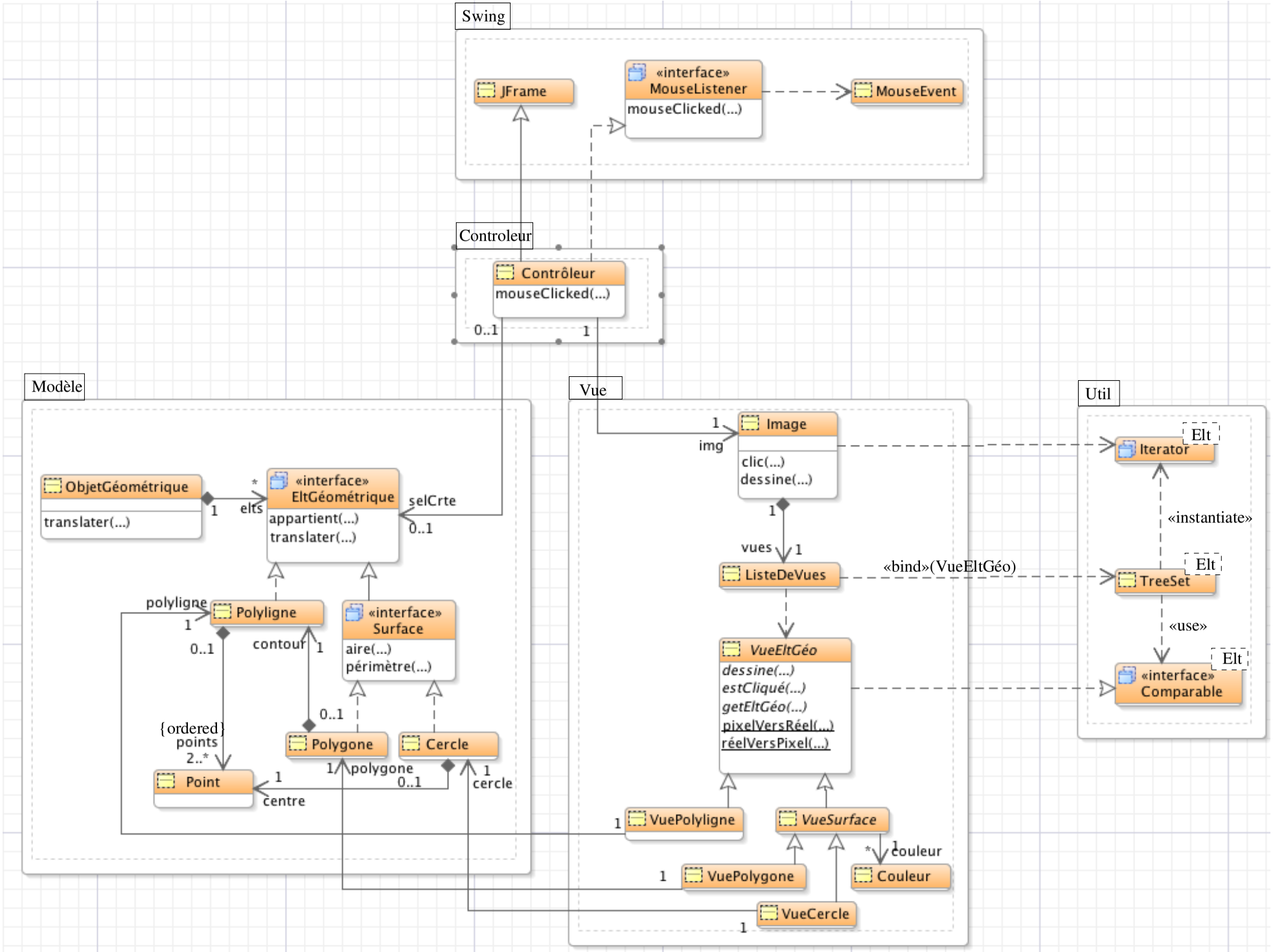

Swing
JFrame
«interface»
MouseListener
mouseClicked(...)
MouseEvent
Controleur
Contrôleur
mouseClicked(...)
0..1
1
Modèle
ObjetGéométrique
translater(...)
1
elts
*
«interface»
EltGéométrique
appartient(...)
translater(...)
selCrte
0..1
polyligne
1
0..1
Polyligne
contour
1
«interface»
Surface
aire(...)
périmètre(...)
0..1
{ordered}
points
2..*
Polygone
Cercle
1
polygone
0..1
1
cercle
Point
1
centre
Vue
1
img
Image
clic(...)
dessine(...)
1
vues
1
ListeDeVues
«bind»(VueEltGéo)
VueEltGéo
dessine(...)
estCliqué(...)
getEltGéo(...)
pixelVersRéel(...)
réelVersPixel(...)
1
VuePolyligne
VueSurface
*
1
couleur
1
VuePolygone
Couleur
VueCercle
1
Util
Elt
Iterator
«instantiate»
Elt
TreeSet
«use»
Elt
«interface»
Comparable

Diagramme de séquence associé au clic de souris