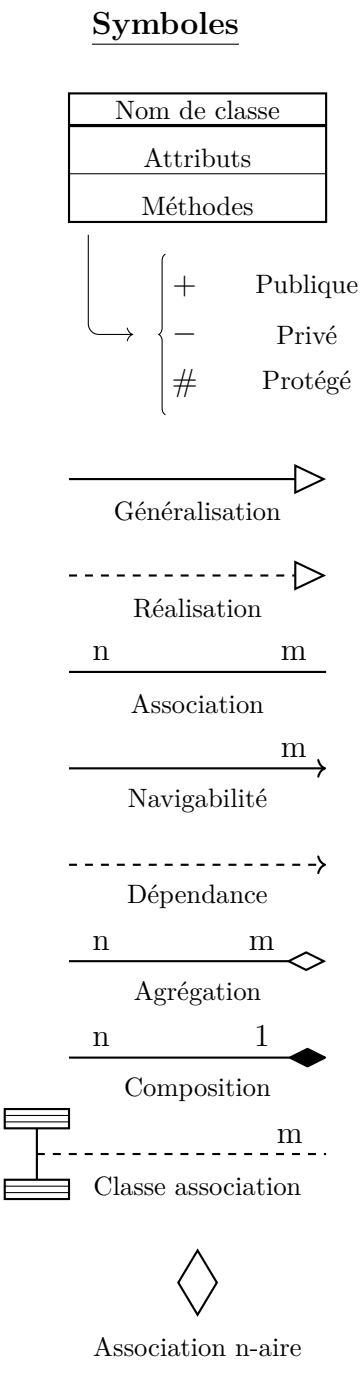
Université Claude Bernard Lyon 1
Polytech Lyon - Département Informatique

4A Apprentis, 2019–2020

ISI-3

TD1. Révisions UML (correction)

Rappel



Le premier encart contient le nom de la classe.
Le second encart contient ses attributs, c'est-à-dire les caractéristiques de la classe.
Le troisième encart contient les méthodes de la classe, c'est-à-dire les fonctions qui peuvent être appelées.
Les attributs comme les méthodes peuvent être publiques, privées ou protégées.
Les attributs/ méthodes publiques sont accessibles par tous, ceux qui sont privés ne le sont que par l'objet qui les contient et ceux qui sont protégés sont accessibles par l'objet et ses descendants.

La classe à l'origine de la flèche hérite de toutes les propriétés (prototypage, attributs, méthodes) de la classe au bout de la flèche.
Une classe ne peut hériter que d'une autre classe.

Une implementation doit pointer sur une interface, c'est-à-dire une classe particulière qui n'a aucun attribut. L'implémentation est la garantie que la classe source contient toutes les méthodes de l'interface. Une classe peut avoir plusieurs implémentations

Une association indique qu'il y a des attributs des classes concernées dans les instances des classes associées.
La cardinalité des extrêmités indique à combien d'instances de la seconde classe, une instance de la première classe doit être liée.

La navigabilité indique que l'association ne peut se lire que dans un seul sens alors que par défaut elle peut se lire dans les deux sens.

La classe source a besoin de la classe cible.

L'agrégation représente une relation d'inclusion structurelle de type tout/partie.

La composition représente une relation de contenance structurelle. Si l'objet composite est détruit, les composantes ne peuvent plus exister.

La classe association indique que l'association entre deux classes se fait par l'intermédiaire d'une troisième classe qui n'existe que dans le cadre de cette association.

L'association n-aire permet de faire des associations pour plus de deux classes.

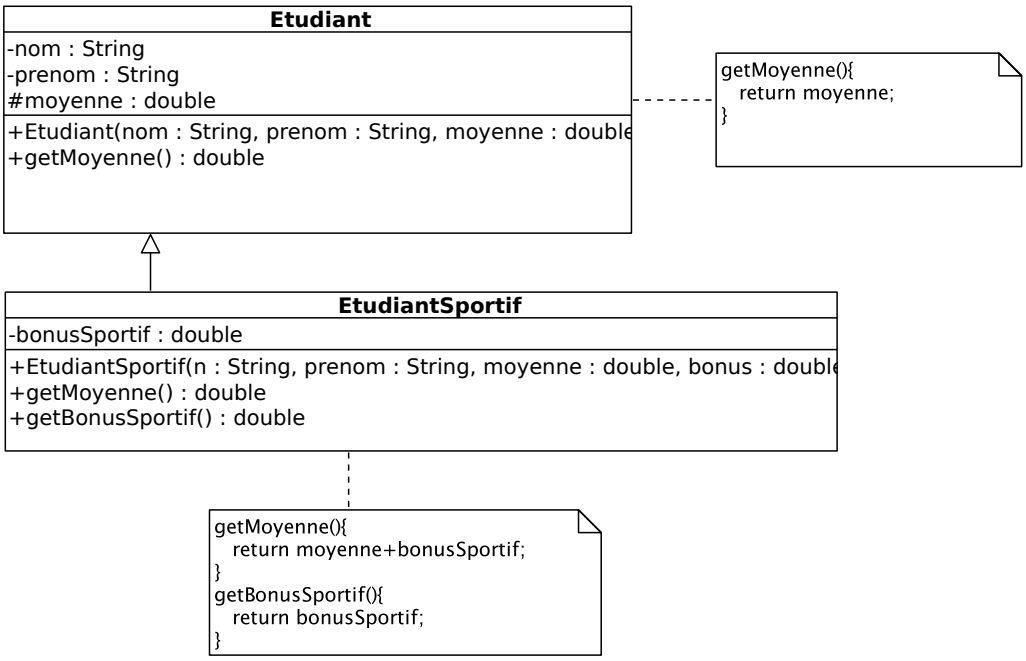


FIGURE 1 –

1 Exercice 1

Soit le diagramme de classes donné en figure 1 avec le code Java de certaines méthodes spécifiées en commentaires UML et les instructions Java suivantes :

```
Etudiant remi = new Etudiant("JACQUES","Remi",15);
Etudiant jean = new EtudiantSportif("DUPONT","Jean",10 , 2);
double d1 = remi.getMoyenne();
double d2 = jean.getMoyenne();
double d3 = jean.getBonusSportif();
```

Question 1 *Identifiez-vous dans ce code de la redéfinition et / ou de la surcharge de fonction ? Si oui, précisez la (ou les) méthode(s) concernée(s).*

`getMoyenne` : redéfinition dans classe fille `EtudiantSportif`.

Question 2 *Identifiez-vous dans ce code du surclassement (upcast) ou sousclassement (downcast) ?*

upcast : mettre dans variable de type déclaré A une instance de type réel B fille de A.
A la ligne 2 : on met dans variable de type déclaré `Etudiant` une instance de type réel `EtudiantSportif` : c'est un surclassement (upcast).

Question 3 *Expliquez ce qui se passe aux lignes 3, 4 et 5 (quelle méthode de quelle classe est appelée, erreur, . . .) et précisez les valeurs des variables d1, d2 et d3 si cela est possible.*

A la compilation le choix de la méthode se fait sur le type déclaré de la variable. A l'exécution, la méthode de la classe du type réel est exécutée.
A la compilation, `remi` et `jean` considérés comme variables de type `Etudiant`.
A la ligne 3, la méthode `getMoyenne` de `Etudiant` est appelés. `d1` vaut 15.
A la ligne 4, la méthode `getMoyenne` de `EtudiantSportif` est appelée. `d2` vaut 12.
A la ligne 5, `getBonusSportif` n'est pas une méthode de `Etudiant` : on a donc une erreur de compilation ! Pour l'éviter, il faut downcaster la variable pour forcer la libération des fonctionnalités cachées par le surclassement fait à la ligne 2. Il faut vérifier avant que le downcast est possible avec `instanceof`.

```
if (jean instanceof EtudiantSportif)
   d3 = ((EtudiantSportif) jean).getBonusSportif();
```

2 Exercice 2

Vous devez concevoir une application permettant de décrire un objet géométrique composé d'éléments géométriques qui peuvent être soit des polylignes, soit des surfaces (voir un exemple en Figure 2). Les polylignes sont des suites ordonnées de points reliés consécutivement par des segments de droites, un point ayant 2 coordonnées x et y (exprimées en mètres par rapport à un point origine). Les surfaces sont soit des cercles soit des polygones. Un cercle est défini par un point, correspondant à son centre, et une valeur numérique positive, correspondant à son rayon. Un polygone est défini par une suite ordonnée de points reliés consécutivement par des segments de droites, le dernier point étant également relié au premier point.

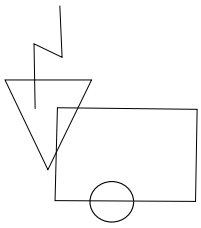
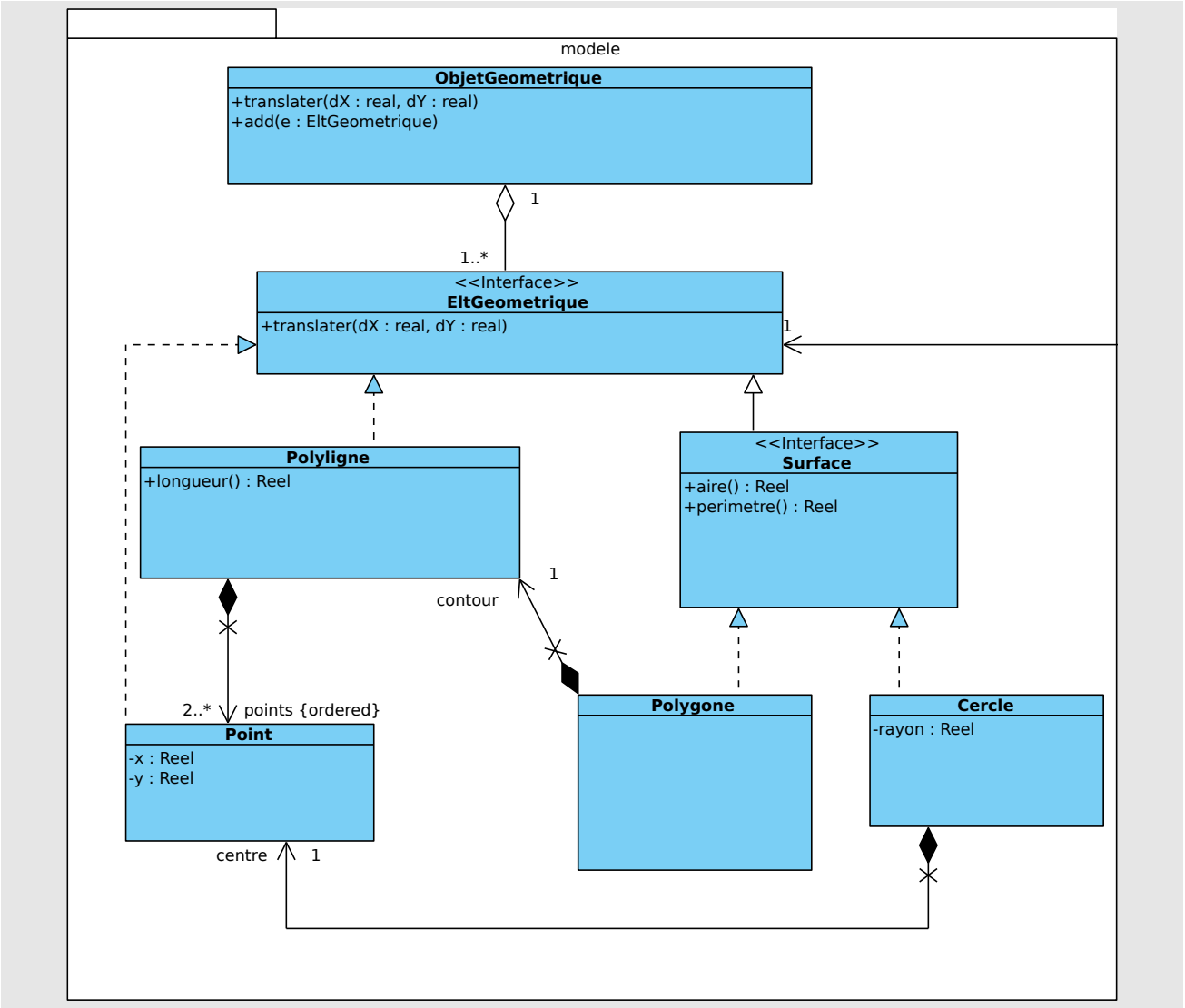


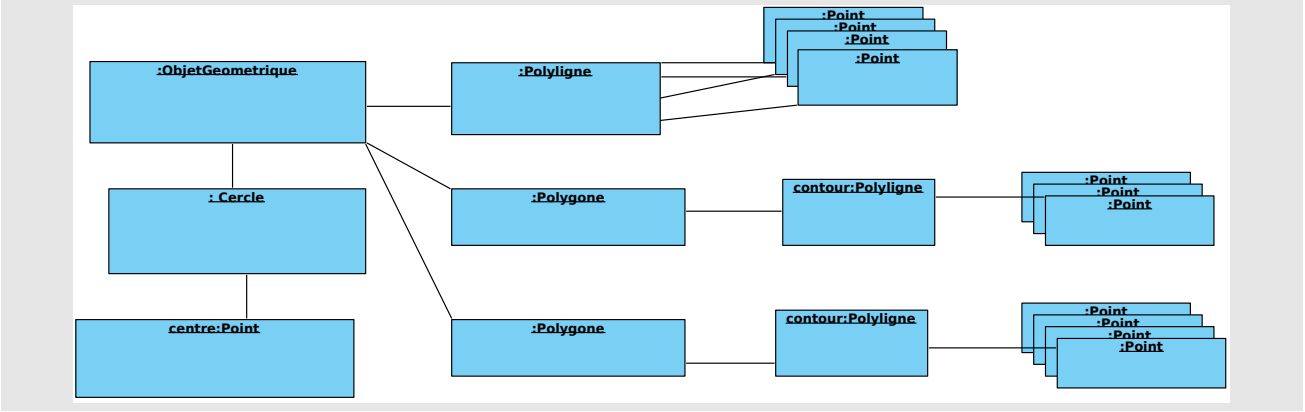
FIGURE 2 – Exemple d'objet géométrique composé d'une polyligne à 4 points, de 2 polygones (ayant 3 et 4 points respectivement) et d'un cercle.

L'application doit permettre de :
- calculer la longueur d'une polyligne ;
- calculer le périmètre et l'aire d'une surface ;
- translater un élément géométrique ou l'objet géométrique dans sa totalité par rapport à un vecteur spécifié par 2 valeurs δx et δy.

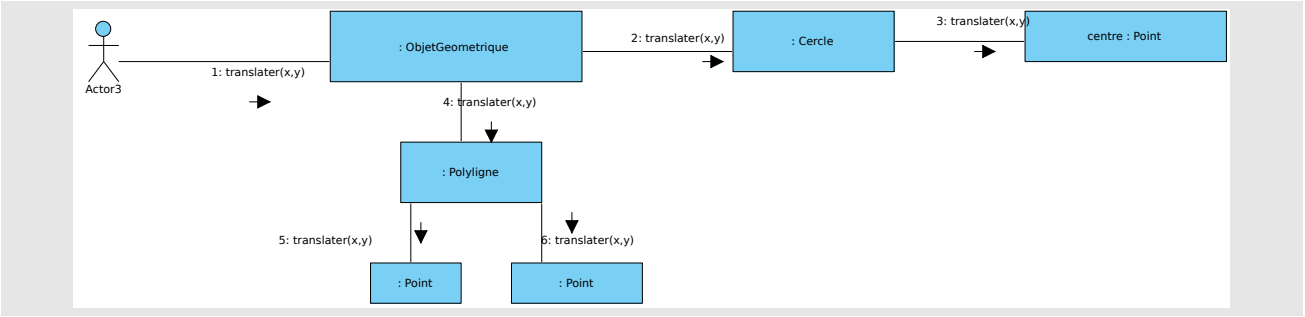
Question 1 *Dessinez le diagramme de classes correspondant à cette application en précisant les attributs et méthodes de chaque classe, ainsi que les multiplicités sur les associations entre classes.*



Question 2 *Instanciez ce diagramme de classes en un diagramme d'objet correspondant au schéma de la Figure 2.*



Question 3 *Proposer un diagramme de communication illustrant la translation d'un objet géométrique composé d'un cercle et d'une polyligne à 2 points.*



Vous devez maintenant compléter le diagramme de classes proposé en question 1 pour permettre de visualiser l'objet géométrique sous la forme d'une image. Vous disposez pour cela d'un ratio permettant de traduire les coordonnées en mètres de l'objet géométrique en coordonnées en pixels dans l'image.

La représentation graphique d'une polyligne a une épaisseur et une profondeur. La représentation graphique d'une surface a une couleur, une épaisseur de contour et une profondeur. La profondeur permet de déterminer l'ordre dans lequel les éléments sont dessinés, de sorte que les éléments les plus profonds sont partiellement masqués par les éléments moins profonds qui les intersectent (voir un exemple en figure 3).

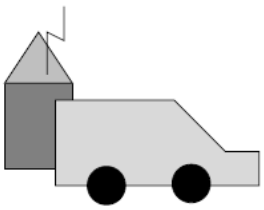
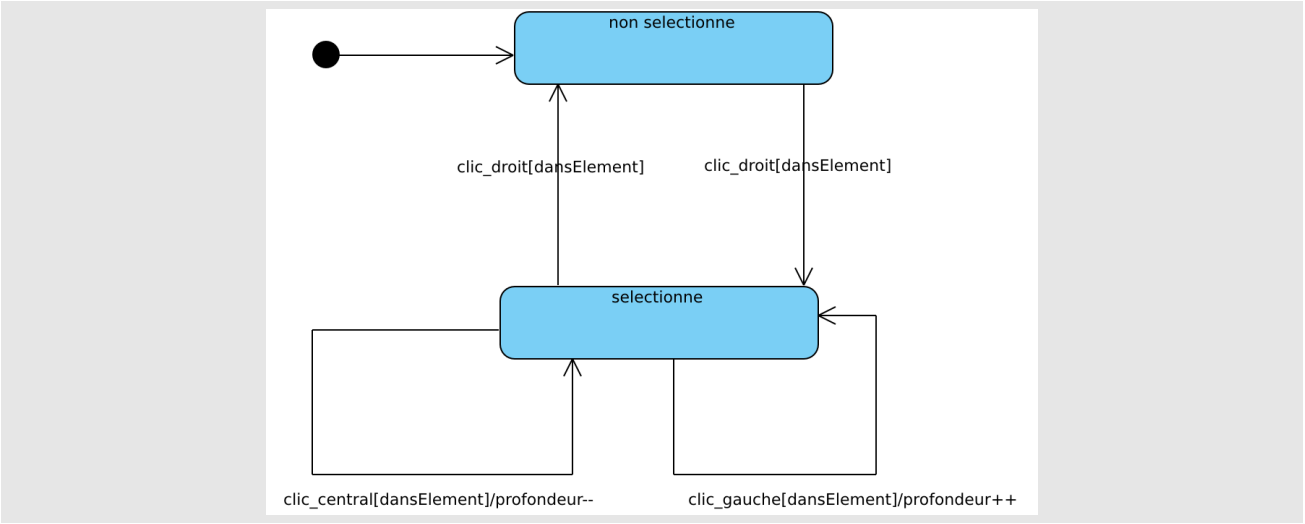


FIGURE 3 – Exemple d'image : la profondeur des cercles et de la polyligne est 1, celle des polygones à 3 points et 6 points est 2, celle du polygone à 4 points est 3.

Un clic droit de souris sur un élément géométrique permet de le sélectionner. Un clic gauche sur un élément sélectionné augmente son niveau de profondeur. Un clic sur le bouton central sur un élément sélectionné diminue son niveau de profondeur. Un clic droit sur un élément sélectionné le dé-sélectionne.

Question 4 *Donner le diagramme d'états-transitions décrivant le comportement d'une représentation graphique d'un élément géométrique.*

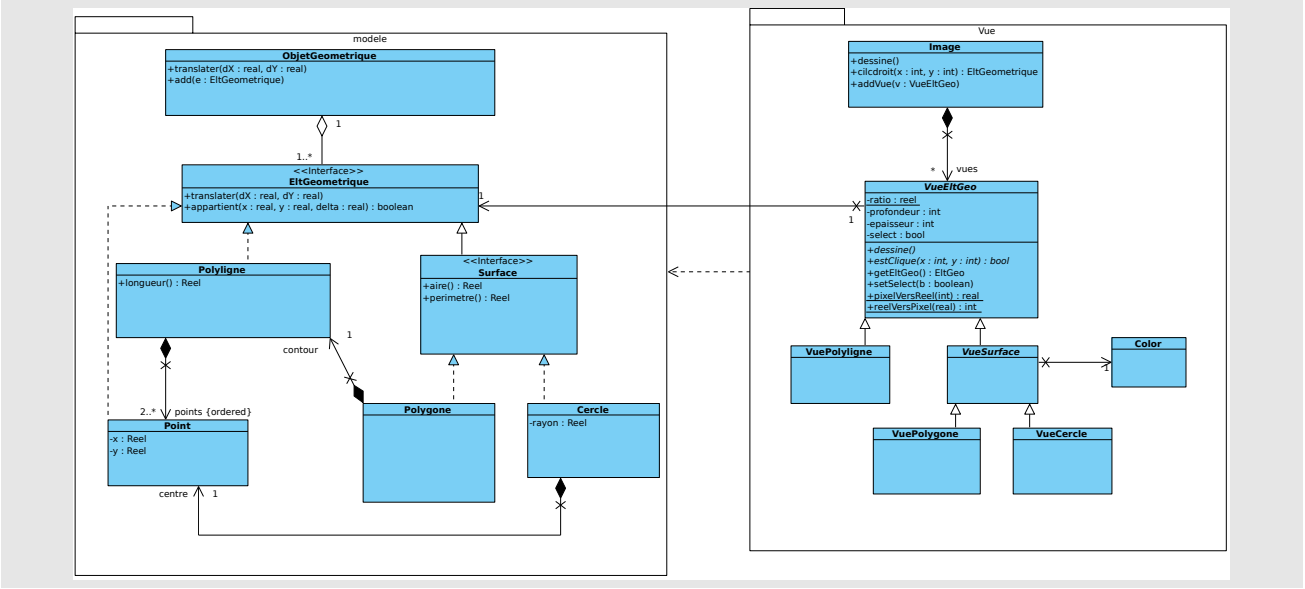


L'application devra notamment permettre de :

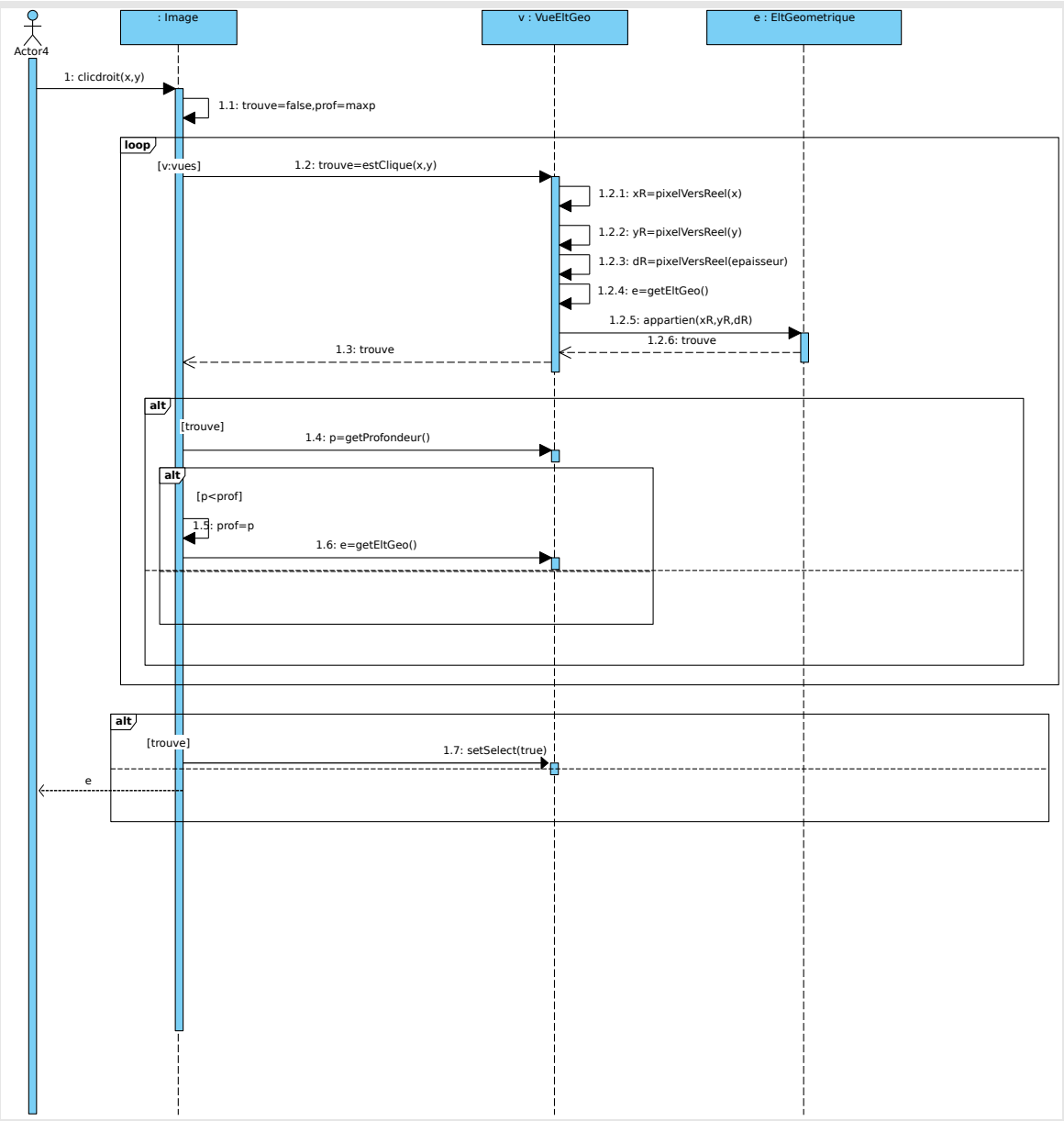
- dessiner l'image correspondant à un objet géométrique ;
- déterminer l'élément géométrique sélectionné par un clic droit de souris en fonction des coordonnées en pixels (x, y) du clic.

Question 5 *Complétez le diagramme de classes de la question 1 du TD6 (classes du modèle)* ***en modifiant au minimum les classes déjà conçues****. Il faudra préserver la séparation du modèle et de l'affichage (vue), en particulier, les classes du modèle devront être indépendante de la vue choisie.*

Vous préciserez les attributs et méthodes de chaque classe, ainsi que les multiplicités sur les associations entre classes.



Question 6 *Proposer un diagramme de séquences pour le scénario suivant : un clic droit sélectionne un élément géométrique.*


Actor4
: Image
v : VueEltGeo
e : EltGeometrique
1: clicdroit(x,y)
1.1: trouve=false,prof=maxp
loop
[v:vues]
1.2: trouve=estClique(x,y)
1.2.1: xR=pixelVersReel(x)
1.2.2: yR=pixelVersReel(y)
1.2.3: dR=pixelVersReel(epaisseur)
1.2.4: e=getEltGeo()
1.2.5: appartien(xR,yR,dR)
1.2.6: trouve
1.3: trouve
alt
[trouve]
1.4: p=getProfondeur()
alt
[p<prof]
1.5: prof=p
1.6: e=getEltGeo()
alt
[trouve]
1.7: setSelect(true)
e