ANTROPIUS Simon 29/04/2024

DESMONTEIX Maxence

shapeEditor

Introduction:

Dans ce rapport, nous vous présenterons notre éditeur de formes, en grande partie grâce à une documentation UML.

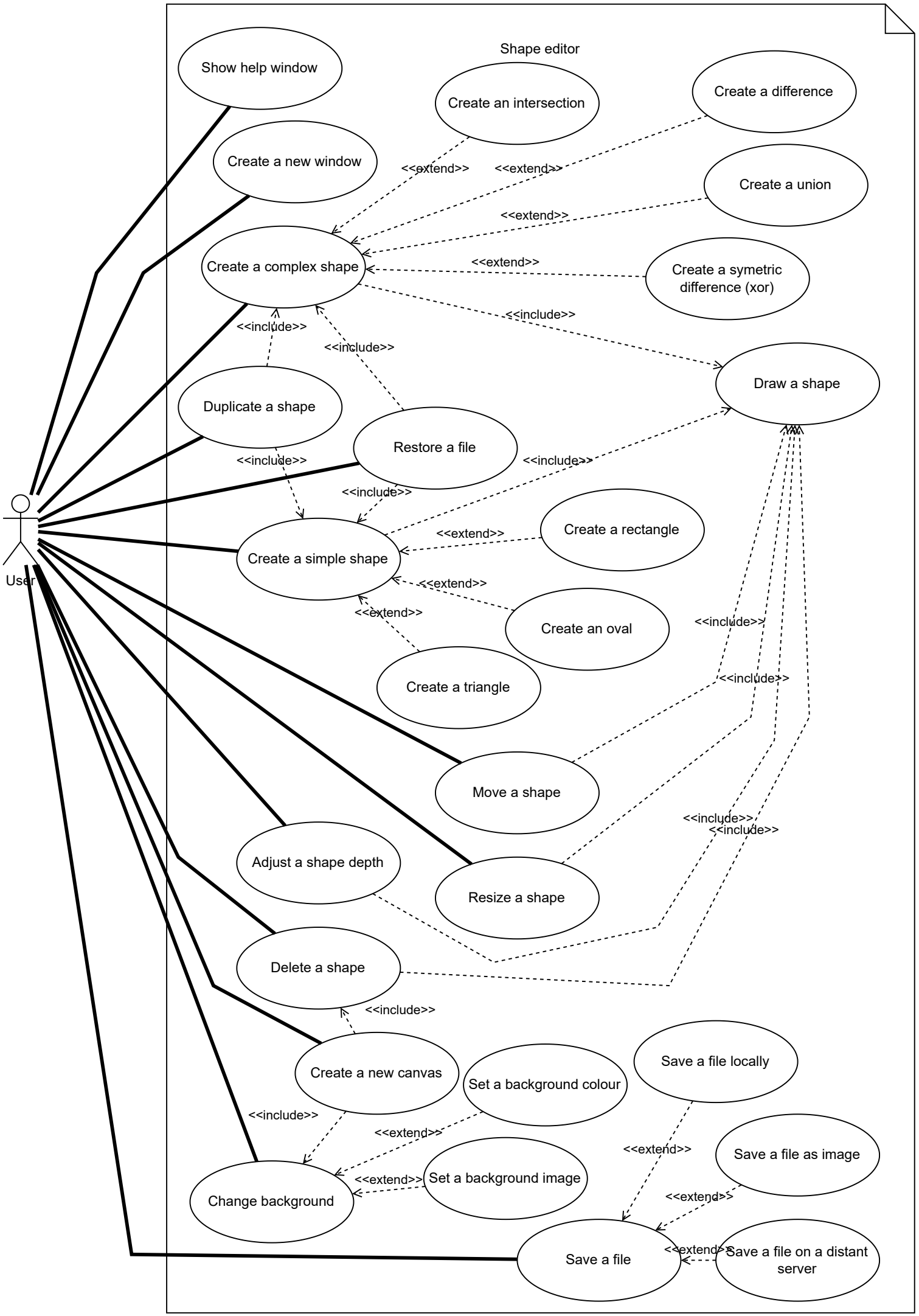
Tout d'abord, un diagramme des cas d'utilisations décrira les fonctionnalités réalisées par notre application. Puis, un diagramme de séquence montrera les étapes réalisées par le logiciel pour créer une différence de 2 rectangles puis pour redimensionner cette différence. Un diagramme d'état présentera ensuite les différentes transitions ayant lieu du déplacement d'une forme. Sur un diagramme d'objets/communication, nous illustrerons les opérations réalisées pour redimensionner une forme et sur un diagramme de classes, nous détaillerons les principaux éléments du logiciel. Enfin, nous expliquerons comment se fait la sauvegarde sur un serveur distant à l'aide d'un diagramme de déploiement.

Après l'UML viendront 2 exemples de contraintes OCL pour un invariant et une pré-post condition d'opération.

Nous conclurons sur les possibilités qu'offre notre application, ses limites et quelques perspectives d'évolution possibles.

UML :

1.1 – Diagramme des cas d'utilisation (version cours)



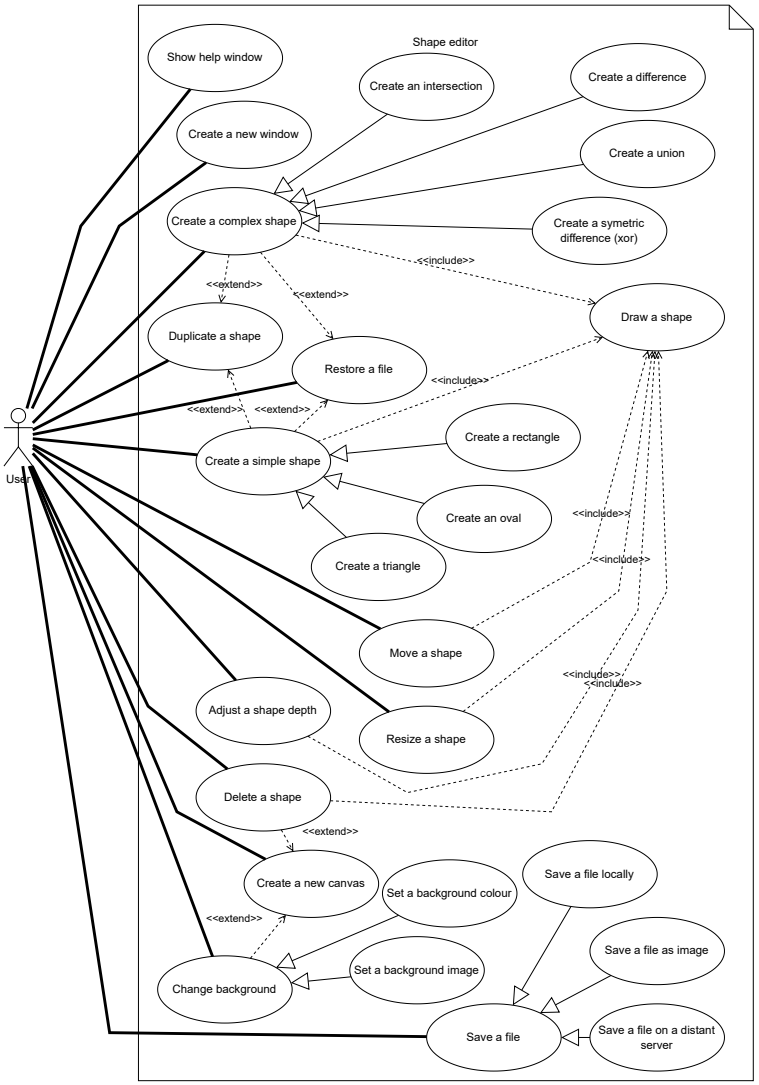
1.2 – Diagramme des cas d’utilisation (en utilisant la documentation officielle)

This diagramm uses the official UML documentation (version 2.5.1)
The relationships are described as the following:

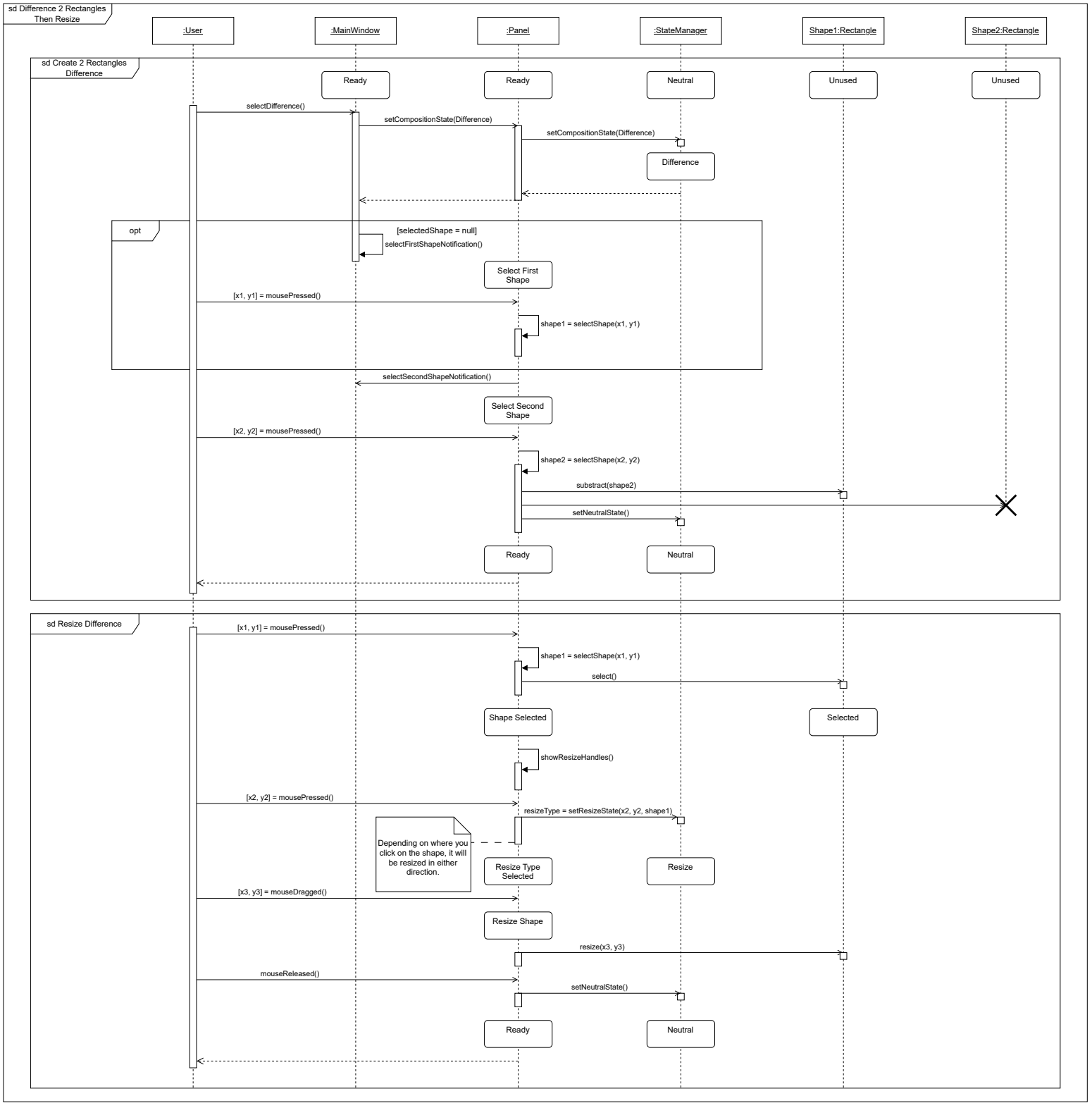
Extends :
An Extend is a relationship from an extending UseCase (the extension) to an extended UseCase (the extendedCase) that specifies how and when the behavior defined in the extending UseCase can be inserted into the behavior defined in the extended UseCase. The extension takes place at one or more specific extension points defined in the extended UseCase. Extend is intended to be used when there is some additional behavior that should be added, possibly conditionally, to the behavior defined in one or more UseCases. The extended UseCase is defined independently of the extending UseCase and is meaningful independently of the extending UseCase. On the other hand, the extending UseCase typically defines behavior that may not necessarily be meaningful by itself. Instead, the extending UseCase defines a set of modular behavior increments that augment an execution of the extended UseCase under specific conditions.

Includes :
Include is a DirectedRelationship between two UseCases, indicating that the behavior of the included UseCase (the addition) is inserted into the behavior of the including UseCase (the includingCase). It is also a kind of NamedElement so that it can have a name in the context of its owning UseCase (the includingCase). The including UseCase may depend on the changes produced by executing the included UseCase. The included UseCase must be available for the behavior of the including UseCase to be completely described. The Include relationship is intended to be used when there are common parts of the behavior of two or more UseCases. This common part is then extracted to a separate UseCase, to be included by all the base UseCases having this part in common. As the primary use of the Include relationship is for reuse of common parts, what is left in a base UseCase is usually not complete in itself but dependent on the included parts to be meaningful. This is reflected in the direction of the relationship, indicating that the base UseCase depends on the addition but not vice versa. All of the behavior of the included UseCase is executed at a single location in the included UseCase before execution of the including UseCase is resumed. The Include relationship allows hierarchical composition of UseCases as well as reuse of UseCases.

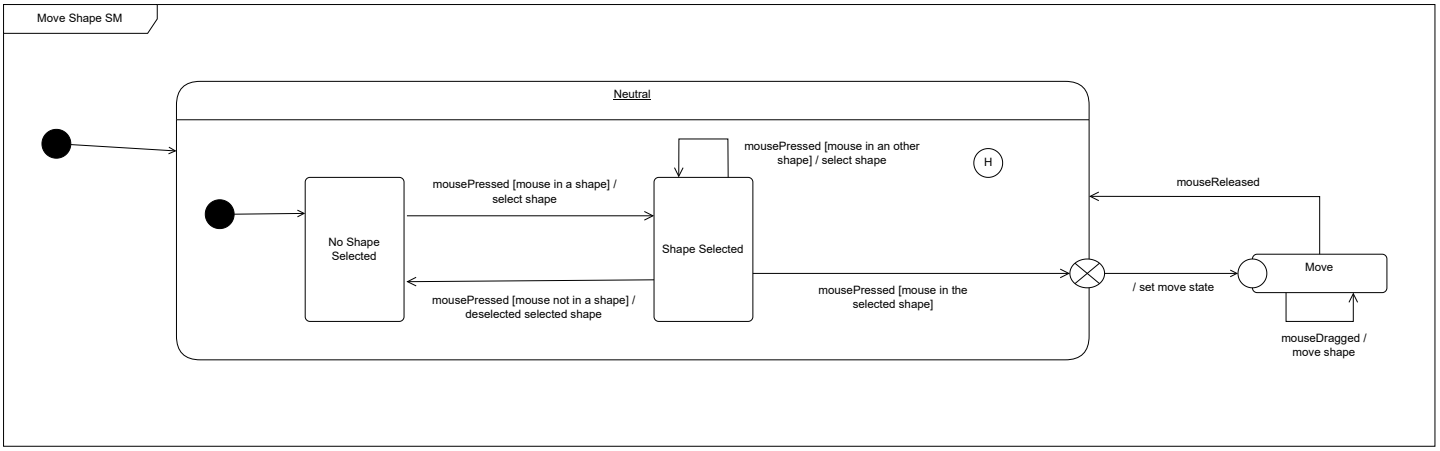
Finally, the generalization relationship establishes an ‘is-a’ connection between two use cases, indicating that one use case is a specialized version of another.



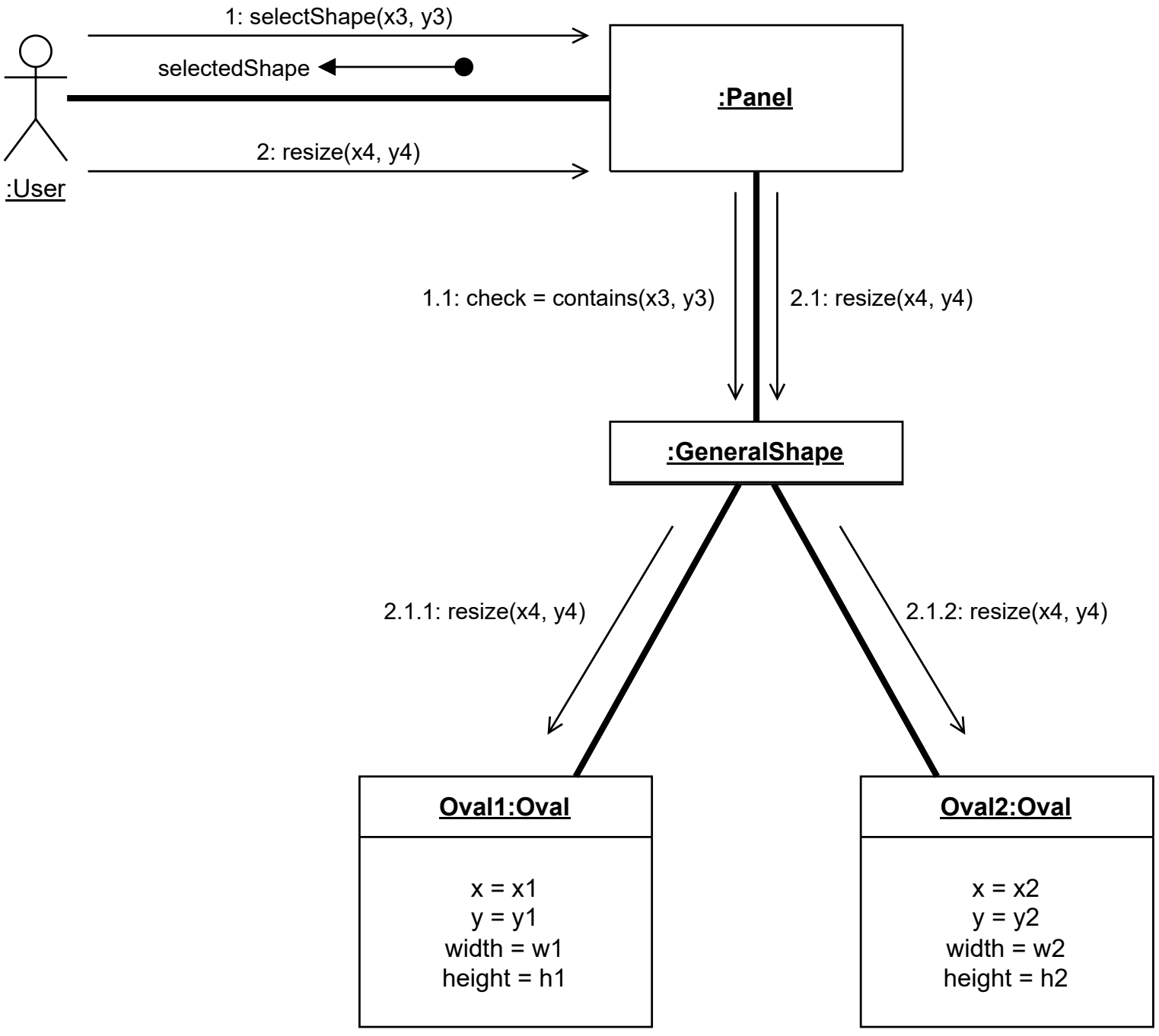
2 – Diagramme de séquence



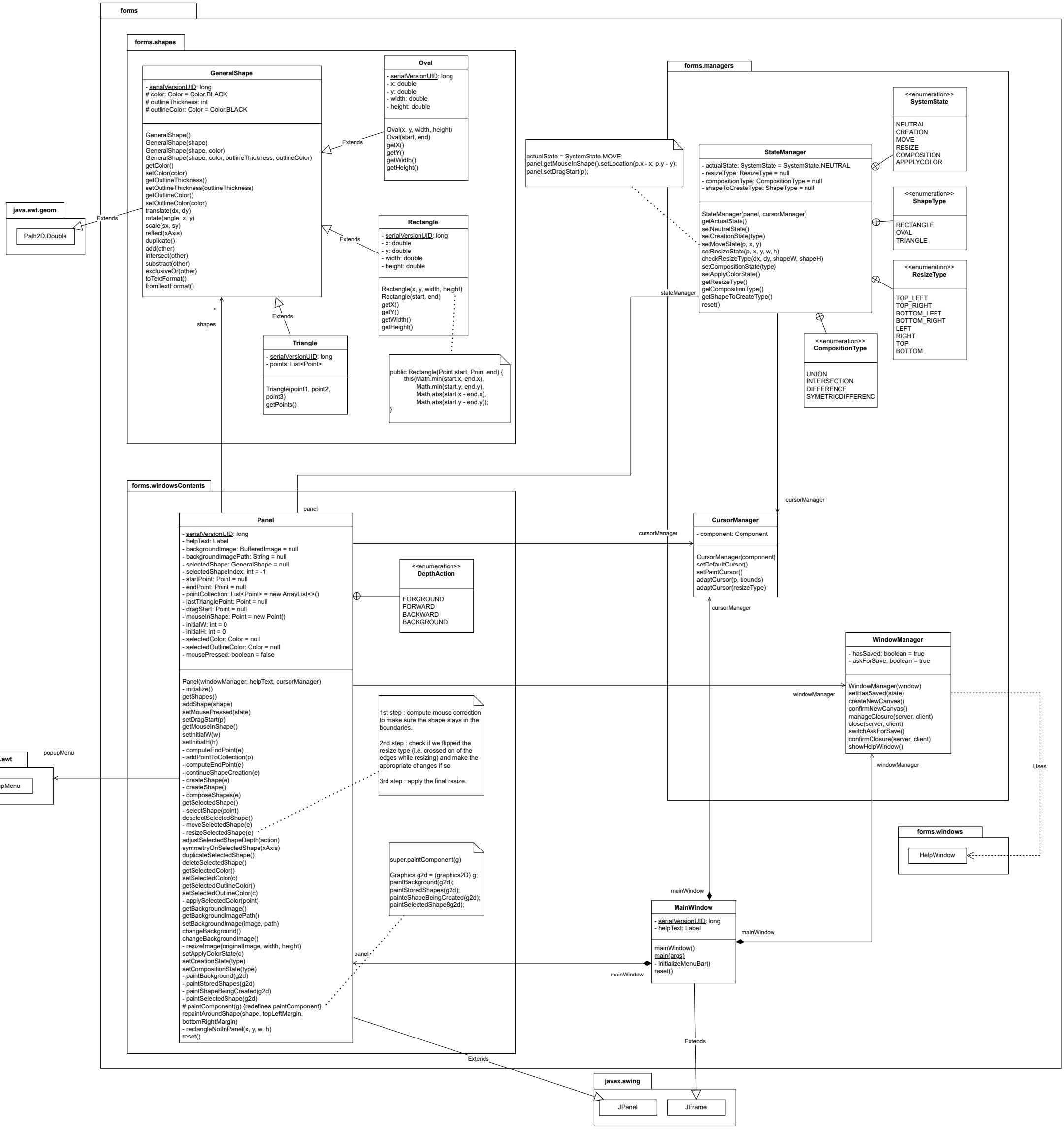
3 – Diagramme d’état



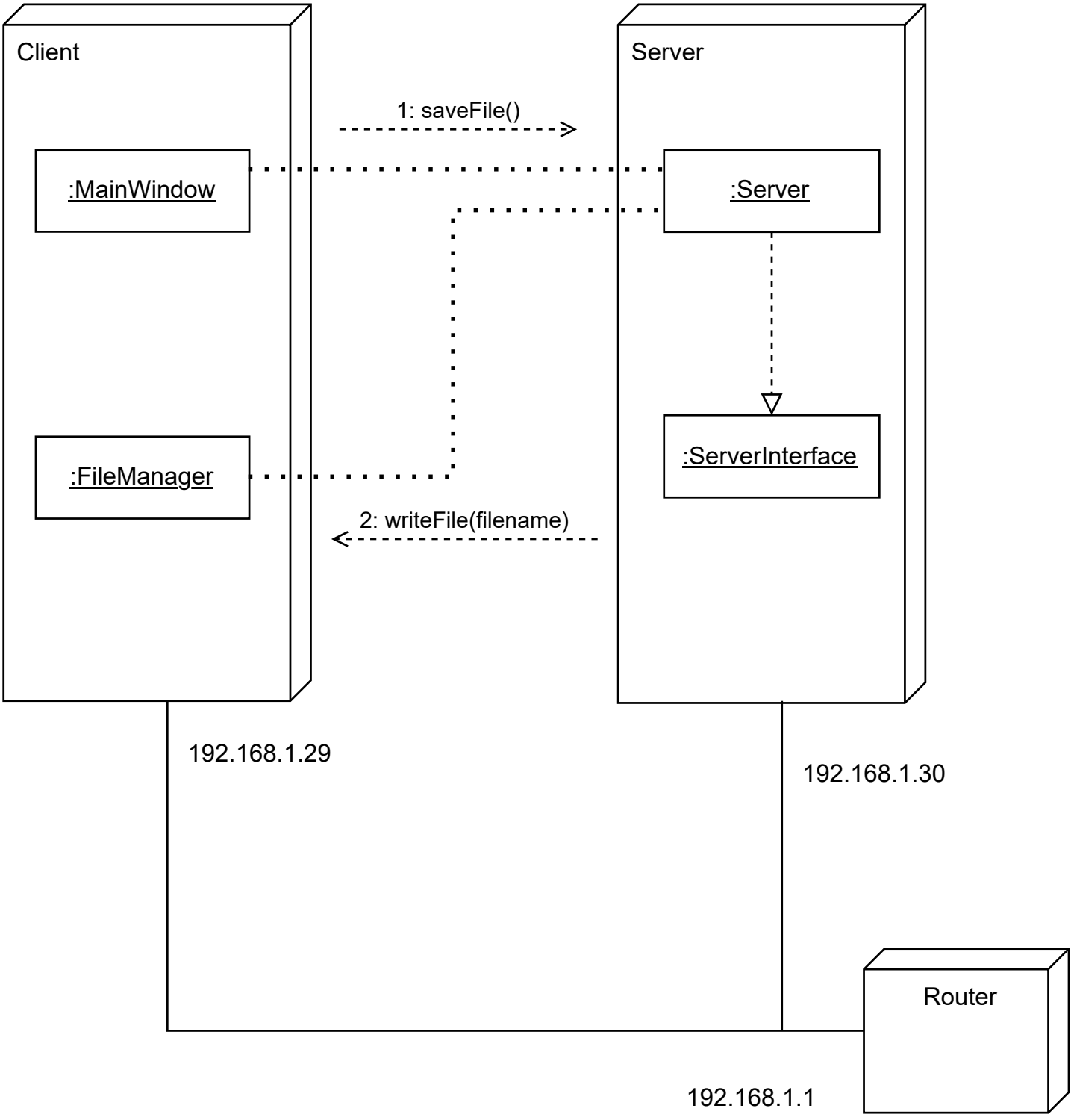
4 – Diagramme d'objets/communication



5 – Diagramme de classe



6 - Diagramme de déploiement



1) Lorsqu'on clique sur le bouton pour sauvegarder un fichier sur le serveur (géré dans MainWindow) ça appelle la méthode saveFile() de la classe Server.
2) La méthode saveFile() de Server appelle ensuite la méthode writeFile(filename) de FileManager.
3) La méthode writeFile(filename) de FileManager écrit le fichier dans un répertoire précis avec le nom filename.

Contraintes OCL :

Invariant	Pre-post condition
class Rectangle attributes x : Integer y : Integer width : Integer height : Integer operations getX() getY() getWidth() getHeight() end constraints context r:Rectangle invariant: (r.x>0) and (r.y>0) and (r.width>0) and (r.height>0)	class GeneralShape end class Panel attributes selectedShape : GeneralShape hasSaved : Boolean operations moveSelectedShape() end constraints context Panel::moveSelectedShape() pre: selectedShape != null post: hasSaved = false

Possibilités de l'application, limites et évolutions possibles :

L'application offre les possibilités suivantes :

- Reset du canvas
- Création d'une nouvelle fenêtre
- Ouverture d'un fichier local
- Ouverture d'un fichier sur machine distante
- Sauvegarder d'un fichier en local
- Sauvegarde d'un fichier sur machine distante
- Sauvegarde d'un fichier en tant qu'image
- Changement du fond de la zone de dessin (couleur ou image)
- Création de forme simple (rectangle, oval, triangle)
- Création de forme composée (union, intersection, différence, différence symétrique)
- Changement de couleur des formes
- Possibilité de mettre un contour aux formes (épaisseur + couleur)
- Modification de la profondeur d'une forme
- Application d'une symétrie à une forme
- Duplication d'une forme
- Suppression d'une forme
- Possibilité de choisir si l'application demande avant de fermer en cas de fichier non sauvegardé
- Ouverture d'une fenêtre d'aide

Nous avons identifiés les limites suivantes :

- Pour désélectionner une forme, il faut soit cliquer sur une autre forme, soit cliquer sur une zone du canvas où il n'y a pas de forme, soit sélectionner certaines opérations à l'aide du menu (comme par exemple, la création de forme). Si jamais le canvas est totalement rempli et qu'il ne reste plus aucun pixel vide, il n'existe plus de possibilité intuitive pour désélectionner une forme. Toutefois, nous pensons que ceci ne pose pas réellement problème dans la mesure où toutes les opérations sont toujours fonctionnelles et nous n'avons pas identifié de cas où il serait indispensable de pouvoir désélectionner une forme.
- Si l'utilisateur agrandi la fenêtre, dessine une forme au bord de celle-ci puis diminue la taille de la fenêtre, alors celui-ci peut faire sortir une forme du cadre. Nous avons réfléchis à une solution en « poussant » par exemple, les formes concernées en m1eme temps que la fenêtre diminue mais il nous a semblé qu'il était préférable de ne rien faire pour la raison suivante : si l'utilisateur a pu agrandir la fenêtre, c'est que son écran est assez grand pour accueillir la forme actuellement en dehors des limites. De plus, cela ne pose pas réellement de problème et ça laisse même une possibilité à l'utilisateur pour mettre des formes en bord de fenêtre, rognant sur l'extérieur. Enfin, si nous avions fait en sorte que la forme soit poussée vers l'intérieur si l'utilisateur diminue trop la taille de la fenêtre, cela aurait pu être plus dérangeant qu'autre chose car ça pourrait désorganiser tout ce que l'utilisateur avait dessiné auparavant.
- Le réseau ne permet pas encore de restaurer le background si celui-ci est une image (et pas une couleur) et que la connexion se fait entre 2 ordinateurs différents. Cela fonctionne si serveur et client sont sur le même ordinateur.

Nous avons pensé aux améliorations suivantes pour notre application :

- Possibilité de pouvoir sélectionner plusieurs formes d'un coup pour effectuer des opérations communes (redimensionnement, duplication, mouvement etc...)
- Rotation de formes
- Possibilité de zoom/dézoom pour avoir une taille de dessin plus adaptable
- Ajouter d'autres formes (dessin libre, polygone avec nombre de côtés au choix etc...)
- Possibilité de revenir en arrière (CTRL + Z)
- Possibilité de mettre des zones de texte