# Stap 1

Bij stap 1 hebben wij een applicatie gemaakt in C# (winforms). Hierbij is er een panel ter grootte van het hele scherm. De form registreert wanneer de gebruiker de muis indrukt(mousedown event) en slaat dan de positie van de muis op. Zodra de gebruiker de muis weer loslaat wordt wederom de positie opgeslagen, deze twee punten maken de x- en y-coördinaten van het begin en eindpunt van het vlak dat de gebruiker net geselecteerd heeft.

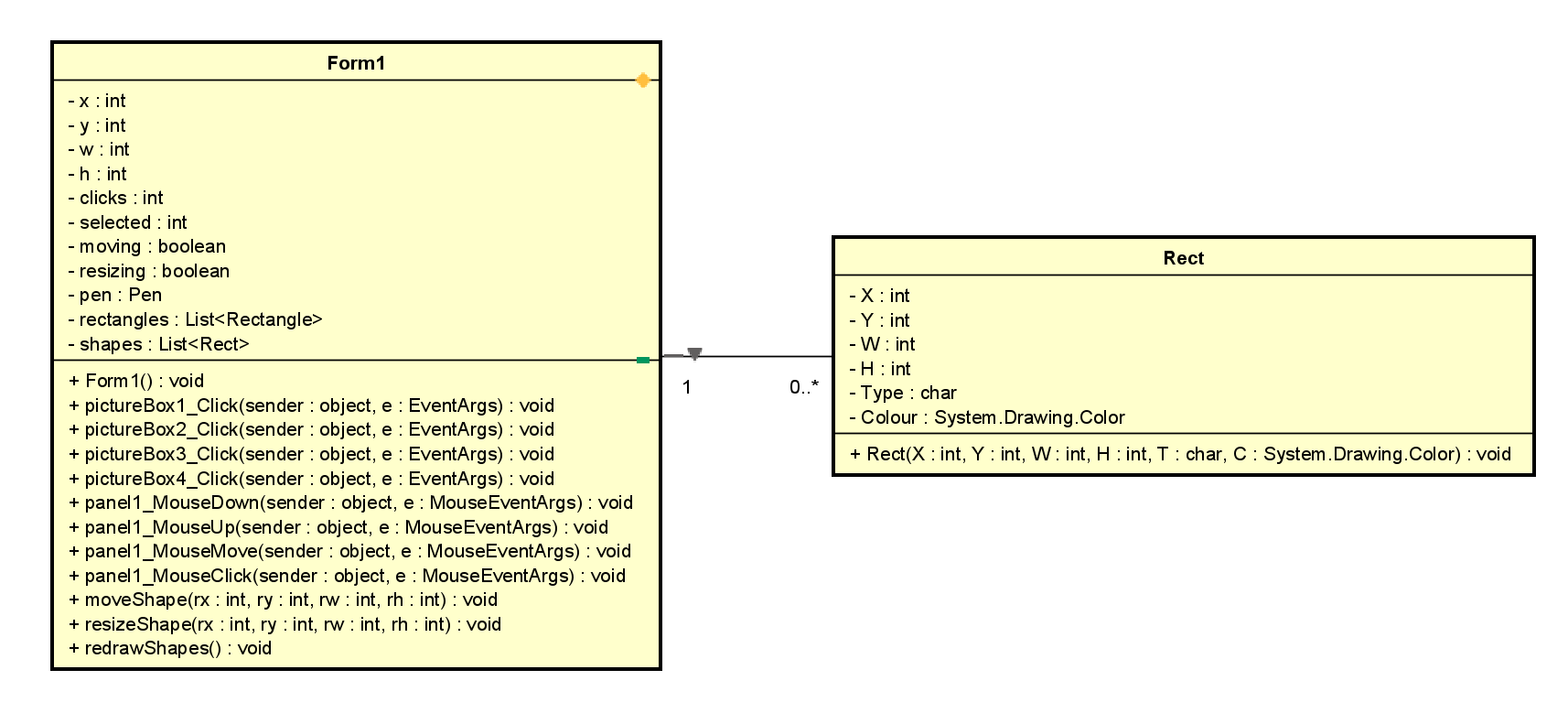
In het mouseup event wordt dan nagegaan welke radio button geselecteerd is. Als dit ellis is, wordt er een ellips getekend ter grootte van dit vlak. Als de rechthoek radiobutton geslecteerd is, wordt dit vlak een rechthoek. Als de cursor radiobutton geselecteerd was, en resizing is true(dit gebeurd bij het mousemove event als de cursor is geselecteerd), wordt het door de gebruiker gemaakte vlak de nieuwe grootte van de geselcteerde vorm.

Zodra een vorm getekent is, wordt deze opgeslagen in 2 lijsten, één van rectangles omdat met dit type vormen getekent kunnen worden, en één lijst van het type Rect, omdat het type(ellips of rechthoek) en de kleur in opgeslagen kunnen worden. Als bij mouseup de cursor was geselecteerd, wordt het vlak niet toegevoegd aan de lijsten, er is dan immers geen vlak getekent.

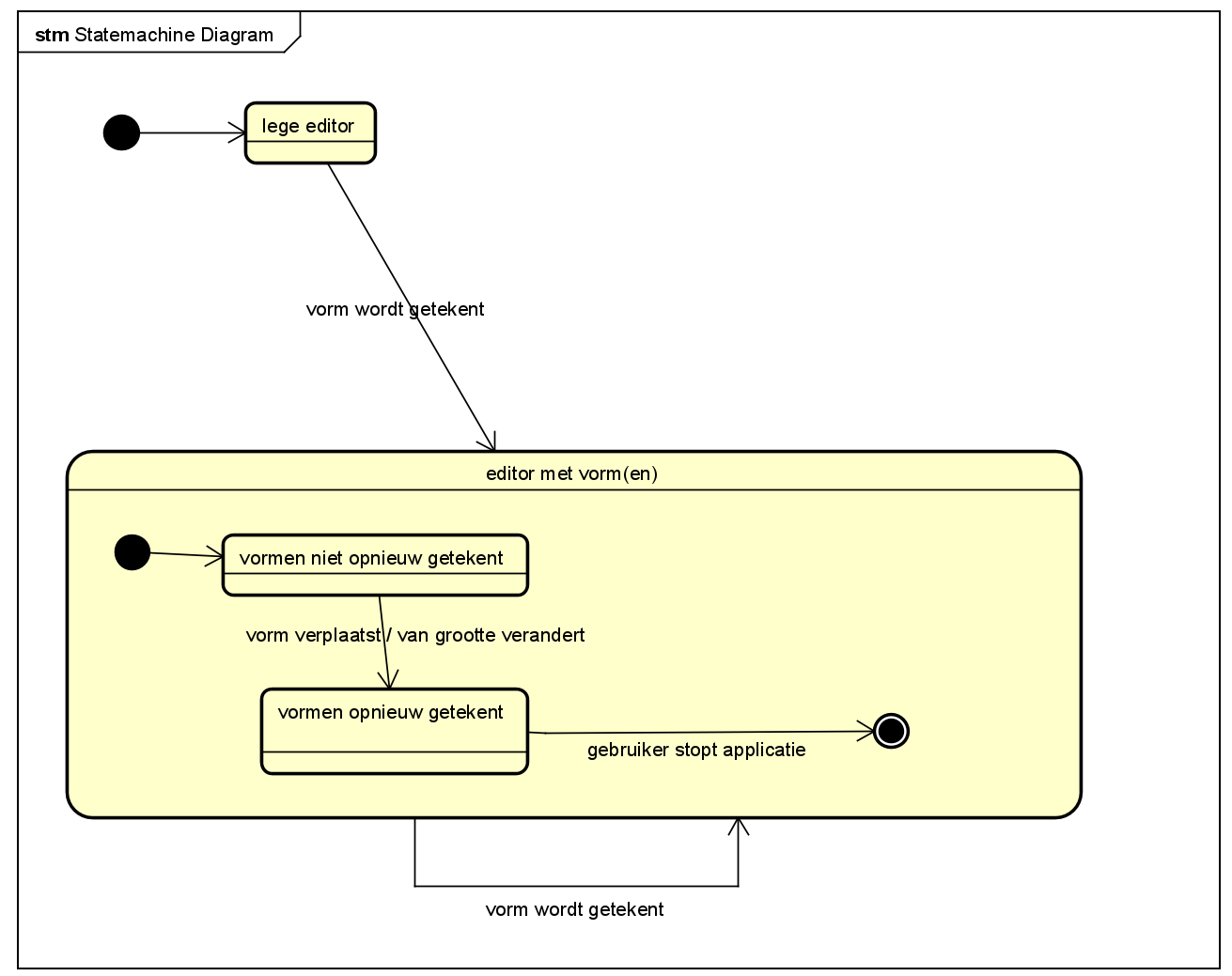
Ook is er een mouseclick event, als dit gebeurdt terwijl de cursor(radiobutton) geselecteerd is wordt er gekeken of op die locatie op het scherm een vorm ligt, zo ja dan wordt een tweede check uitgevoerd om de waarde van clicked te achterhalen, als dat 0 is, gaat clicked met 1 omhoog (click == 1 betekent dat een vorm geselecteerd is). Als clicked al 1 was, dan wordt de positie waar geklikt is de nieuwe positie van de vorm.

Als een vorm is verplaatst of van grootte is veranderd, wordt het scherm gewist en volgens de lijst van Rectangles opnieuw getekend. Dit gebeurt omdat je bij een panel niet één enkele vorm kan verplaatsten/veranderen.

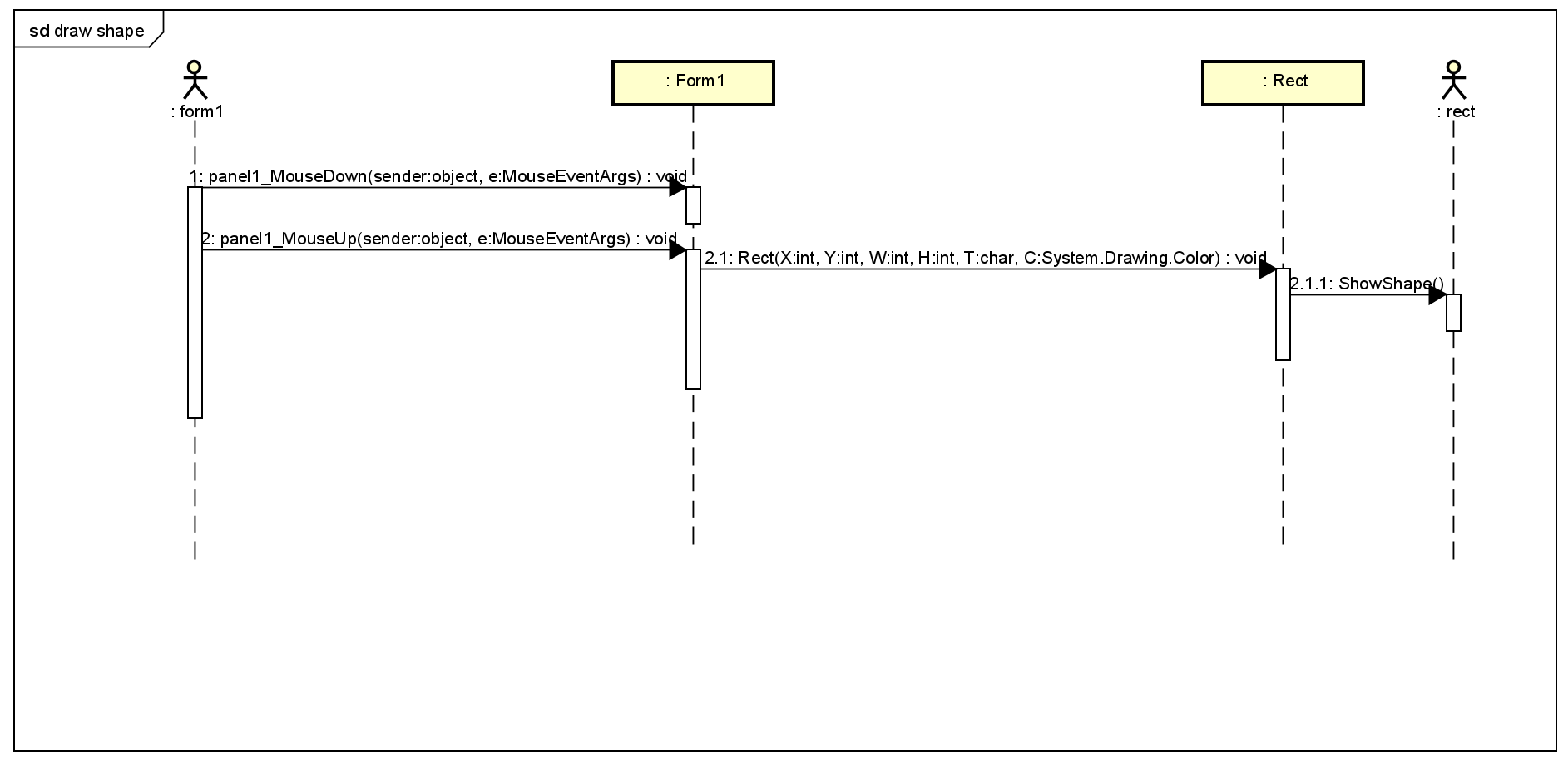
## Klassediagram



Toestandsdiagram



## Tijdsvolgordediagram



# Stap 2

Bij stap 2 is de functionaliteit om een actie te *redo* / *undo* ‘en toegevoegd volgens de *command design pattern*. Hierbij werkt de code van form1 als *client*. De client roept een *invoker* aan. De *invoker* roept een *concrete command* aan. Een *concrete command* is een specifiek commando voor 1 actie, hij erft over van de *interface* *Command*. De commando interface bevat een methode die ervoor zorgt dat de receiver een operatie *execute*. De receiver zit als instantie in de *concrete command*. Op deze manier kan de *draw command* de *receiver* de opdracht geven om zichzelf te tekenen. De receiver is ook een interface, deze wordt geïmplementeerd door een rechthoek class en een ellips class.

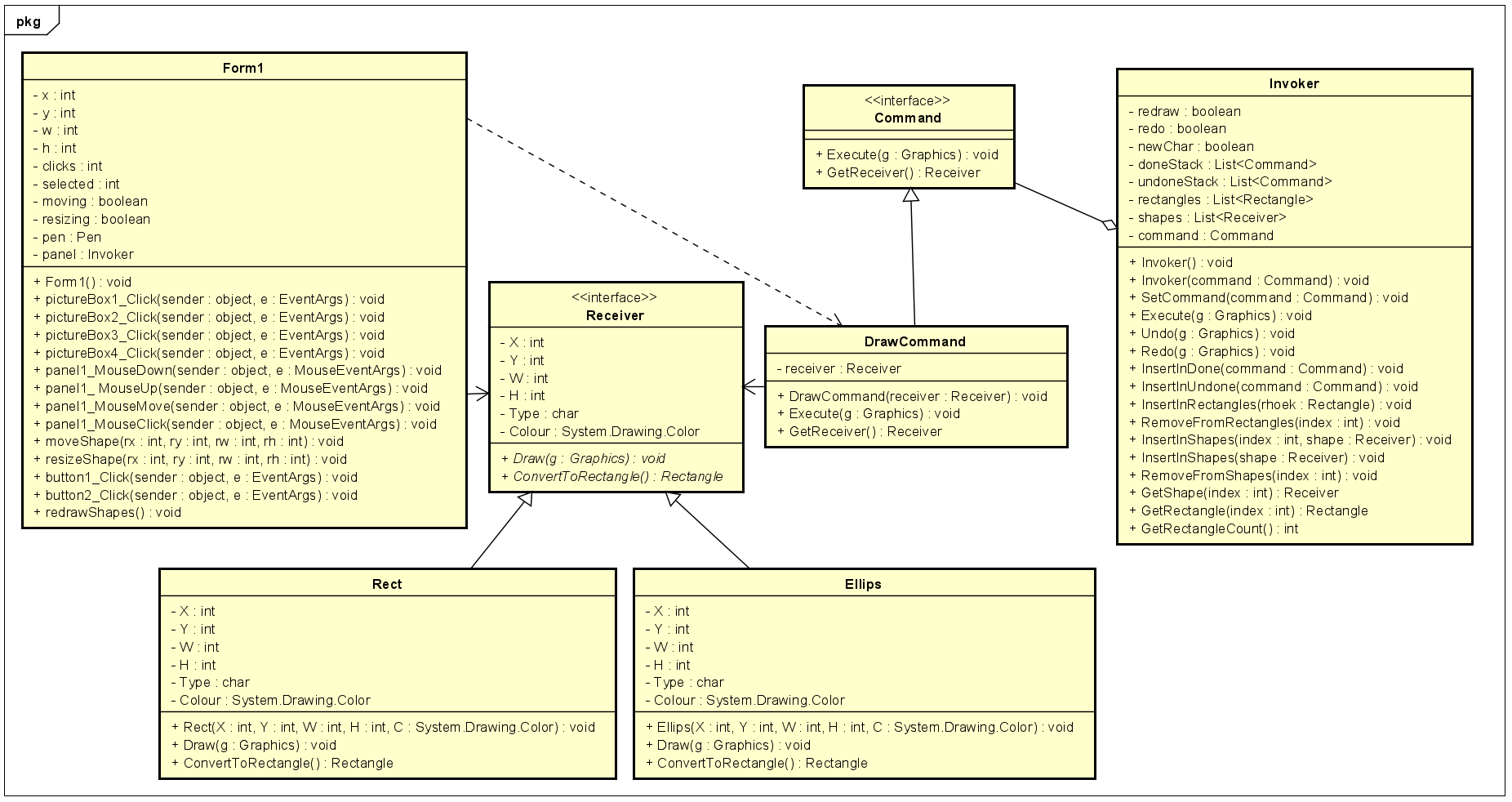
Wat er dus gebeurt in de *client*, is dat de *client* een en *shape* en een *command* aanmaakt. Hij geeft deze aan de *invoker*, en deze bepaalt wanneer deze wordt ge-*execute*.

In ons geval is de implementatie niet volledig gelukt, dit komt door de uitwerking in stap 1. Hierbij zijn resize en move geen apperte commando’s. Dit komt omdat je geen figuur kan selecteren en veranderen, net als bij echte verf zit het figuur op het canvas en kun je het niet meer verplaatsen. Wat er dus bij move gebeurt is dat de positie wordt geupdate en dan worden alle vormen opnieuw getekent door meerdere draw-commando’s. Hierdoor werkt de design pattern wel correct, maar de functionaliteit van het programma werkt niet helemaal naar behoren.

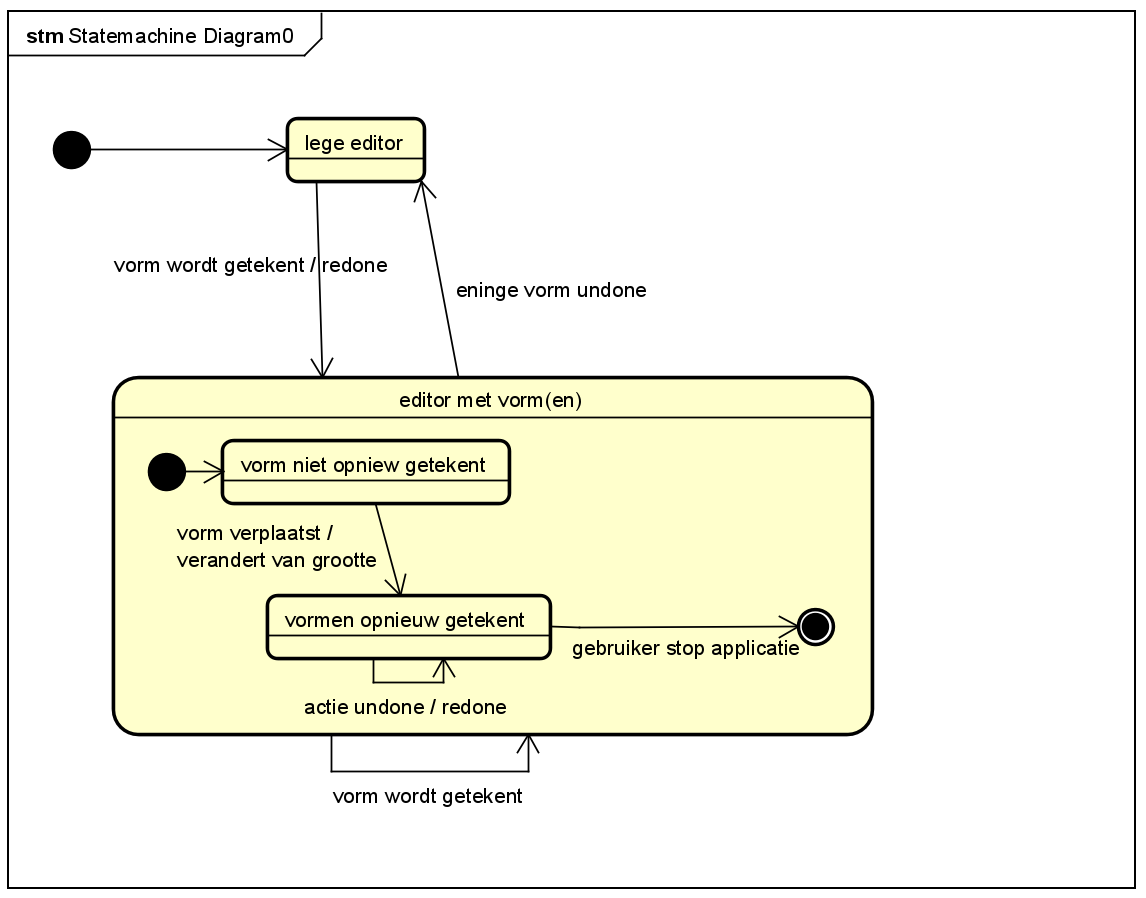
In het programma zijn geen move- en resize-commands, want dit zijn zoals net uitgelegd eigenlijk ook gewoon draw-commando’s.

Achter de schermen wordt er gekeken wat voor soort commando is uitgevoerd, is dit simpelweg het maken van een nieuwe vorm, dan komt er enkel 1 commando op de done-stack. Dit commando is gemakkelijk ongedaan te maken, of opnieuw uit te voeren. Wanneer een vorm verschuift of vergroot wordt, gaat de boolean redraw in de invoker op true. De invoker weet dan dat de figuren opnieuw getekent zijn en verwijdert dan de onderste commando’s van de done-stack, omdat dit, op de vorm die verplaatst is na, duplicaten zijn.

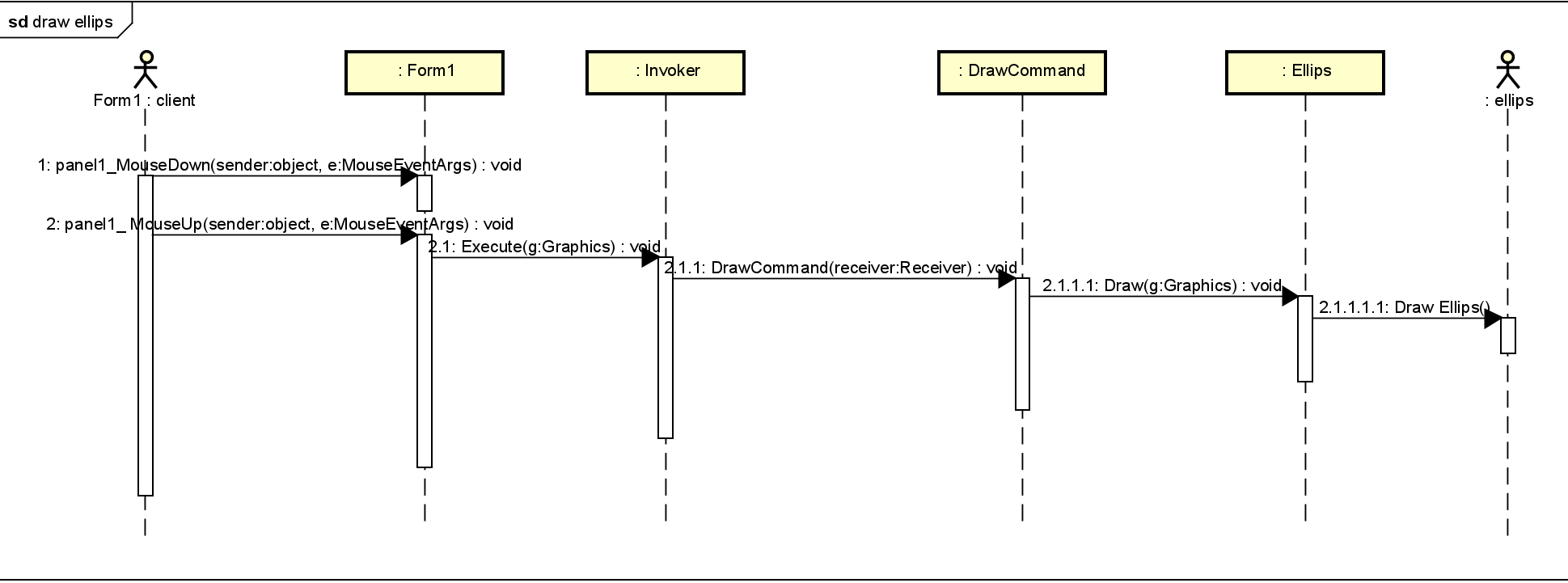
## Klassediagram

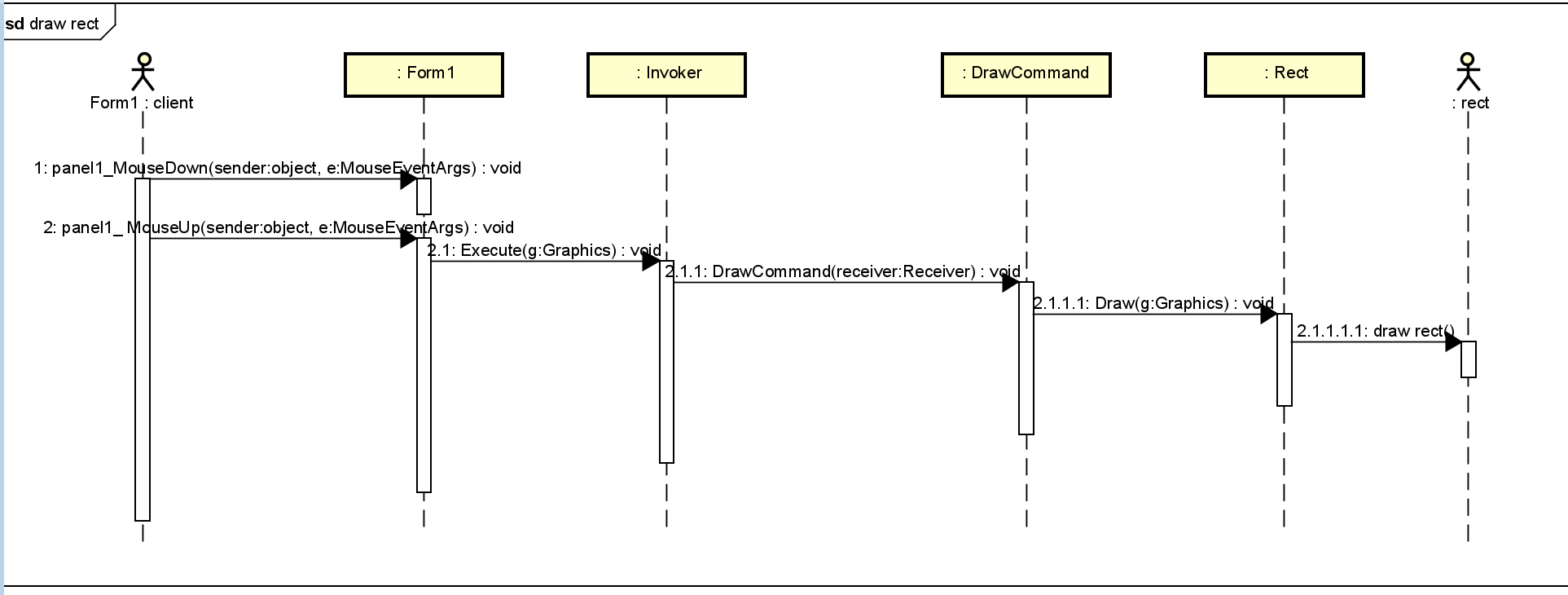


## Toestandsdiagram



## Tijdsvolgordediagram



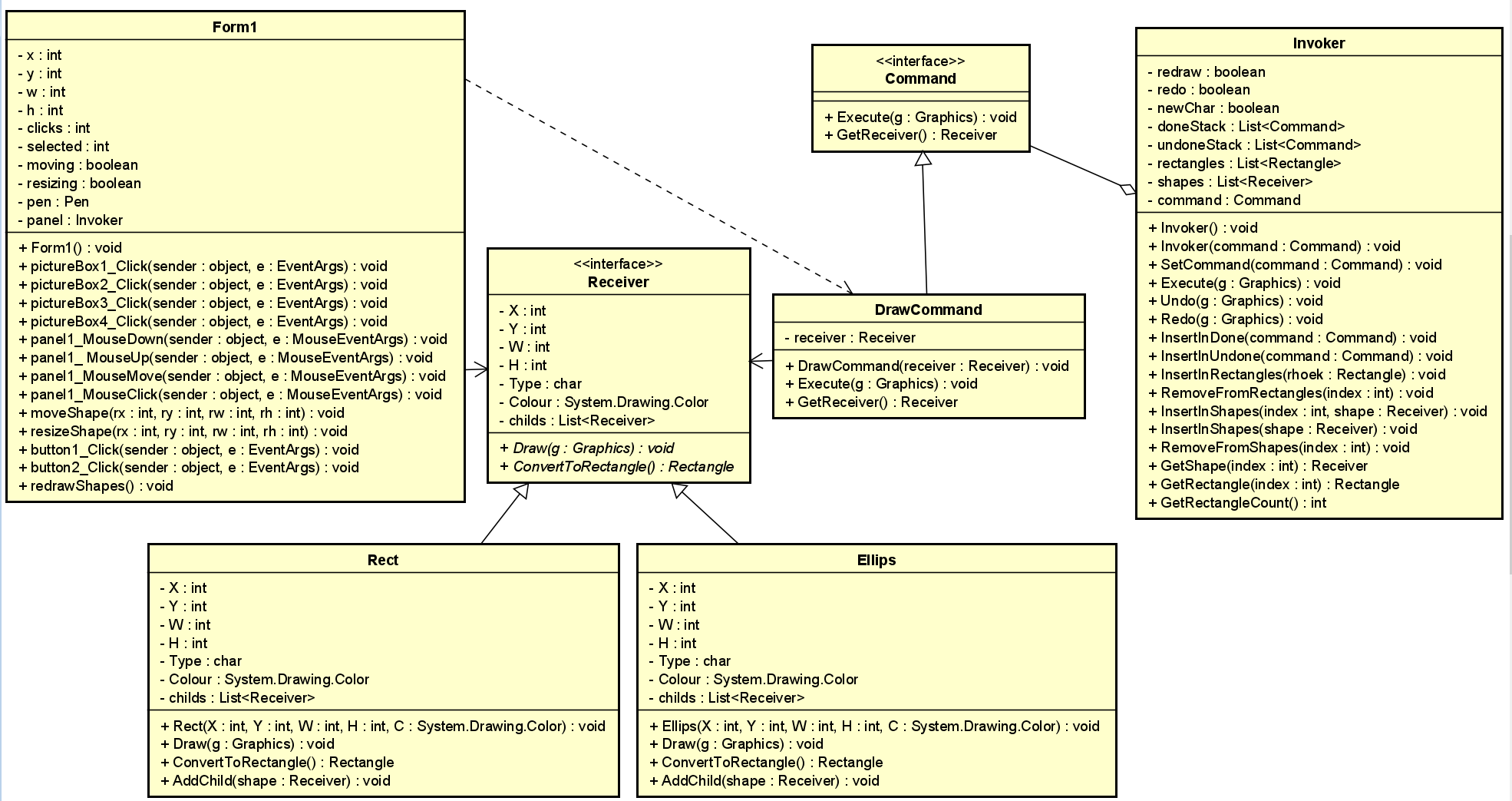


# Stap 3

Bij de transitie van stap 2 naar stap 3 is er niet veel gebeurd om compositie toe te voegen. Wij hebben net als in het voorbeeld in de les een List<Receiver> toegevoegd in de Receiver class, deze wordt dus ook geïmplementeerd door de classes: Rect en Ellips. Hiernaast is ook de methode AddChild toegevoegd, om ondergeschikte vormen toe te voegen. Bij de mousUp event in form1 wordt gecheckt of het getekende vlak binnen een bestaande vorm zit, als dit het geval is wordt de net getekende vorm toegevoegd aan de “childs” list van de vorm waarin hij zich begeeft. Door de manier waarop wij de undo functionaliteit hebben gemaakt, is het niet nodig om een RemoveChild functie te maken/gebruiken.

Bij de tijdsvolgorde diagram is te zien hoe het programma loopt als je een nieuwe rechthoek tekent. Hiernaast is een diagram gemaakt die weergeeft wat er gebeurd als de getekende rechthoek binnen een bestaande vorm ligt.

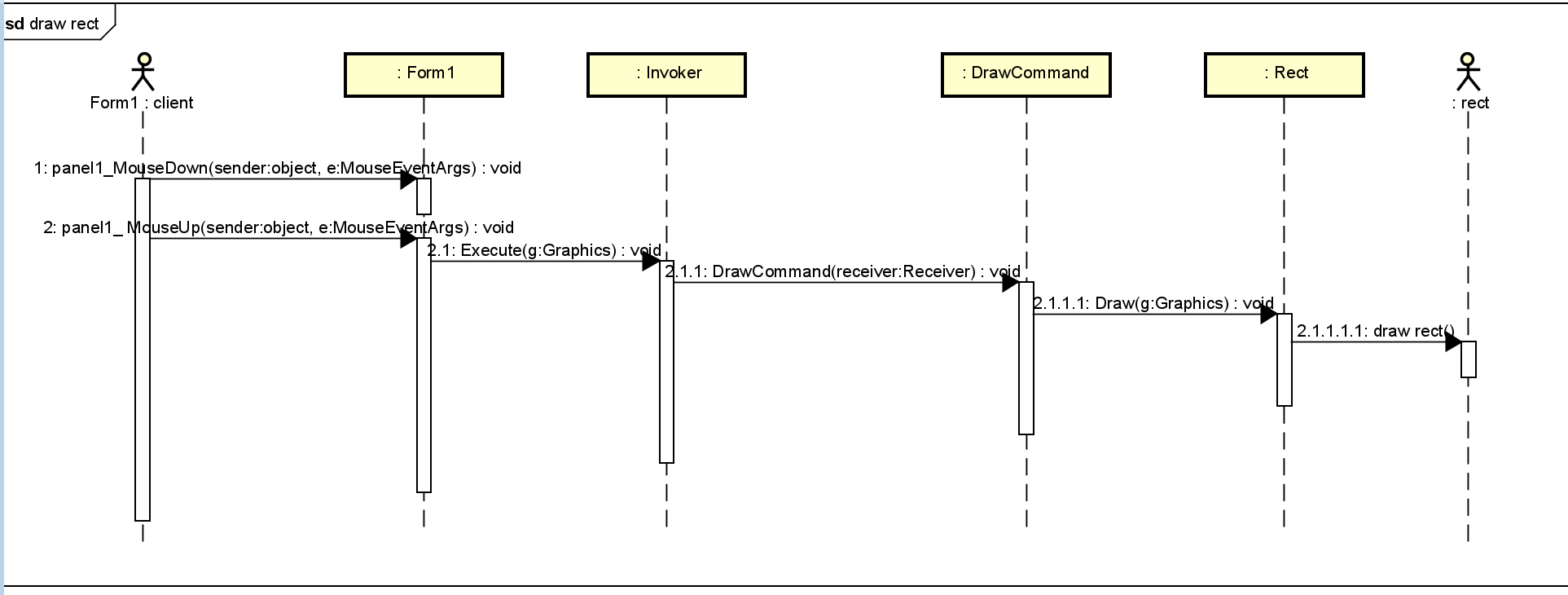
## Klassediagram

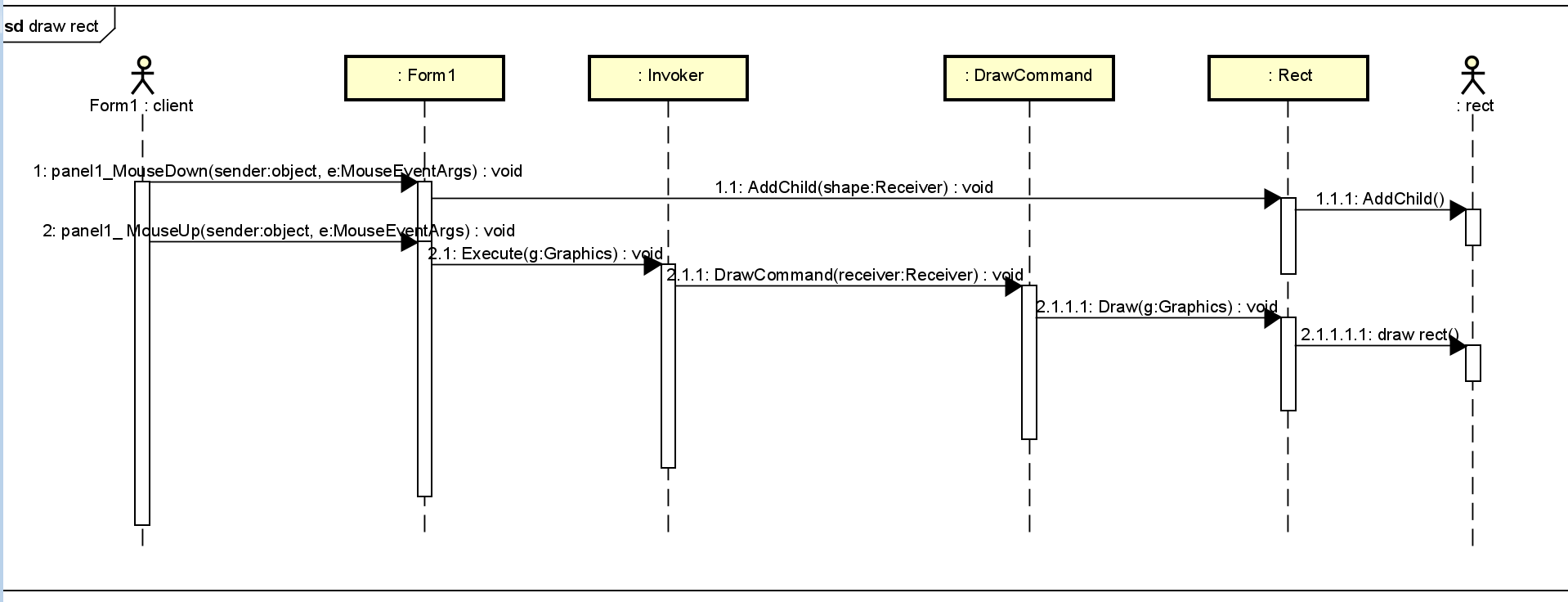


## Toestandsdiagram

Toestand is niet veranderd in tegenstelling tot stap 2. Voor diagram zie toestandsdiagram stap 2.

## Tijdsvolgordediagram





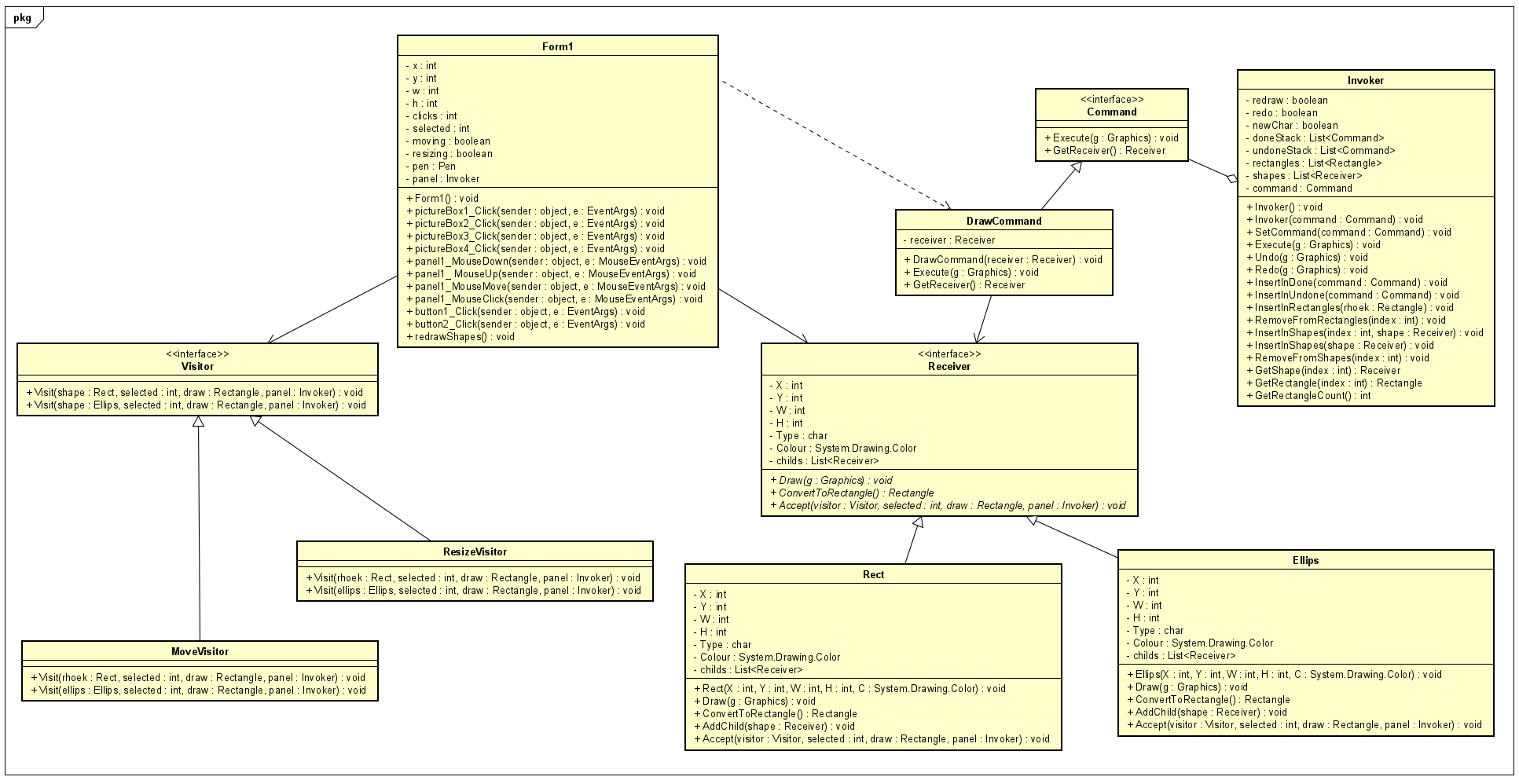
# Stap 4

Bij stap 4 is de *interface* “Visitor” aangemaakt, deze wordt door de klassen “MoveVisitor” en “ResizeVisitor” overgeërfd. De functionaliteit in deze klassen regelt het verplaatsen en veranderen van grootte van de vormen. Hierdoor zijn de functies “MoveShape” en “ResizeShape” uit Form1 gehaald, deze zijn niet meer nodig door de *visitors*.

De klassen “Rect” en “Ellips” hebben de functie *Accept* gekregen, wanneer deze wordt aangeroepen accepteert de vorm dat er een visitor komt en functionaliteit over de form uitvoert.

Dit wordt aangestuurd in Form1, hier wordt een visitor aangemaakt een als parameter meegegeven aan een vorm in de “Accept” functie. De Accept functie roept dan de Visit functie van de visitor aan. Hierin zit de functionaliteit voor verplaatsen, of resizen.

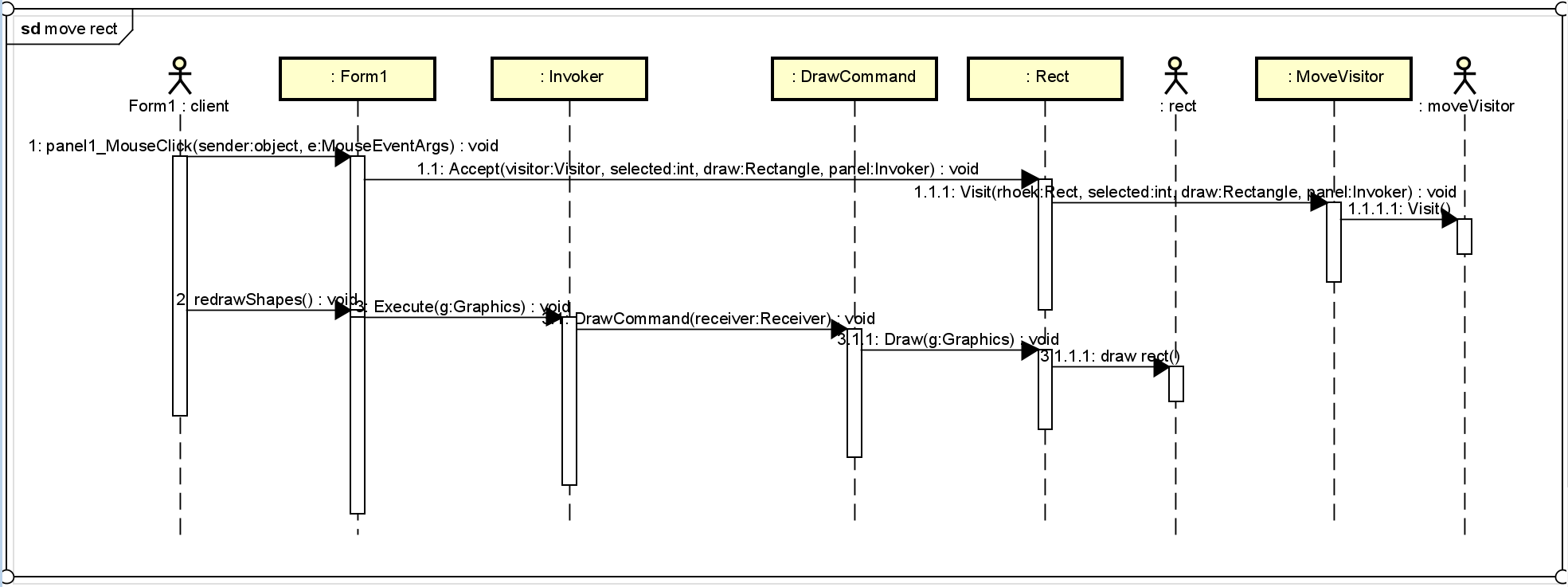
## Klassediagram

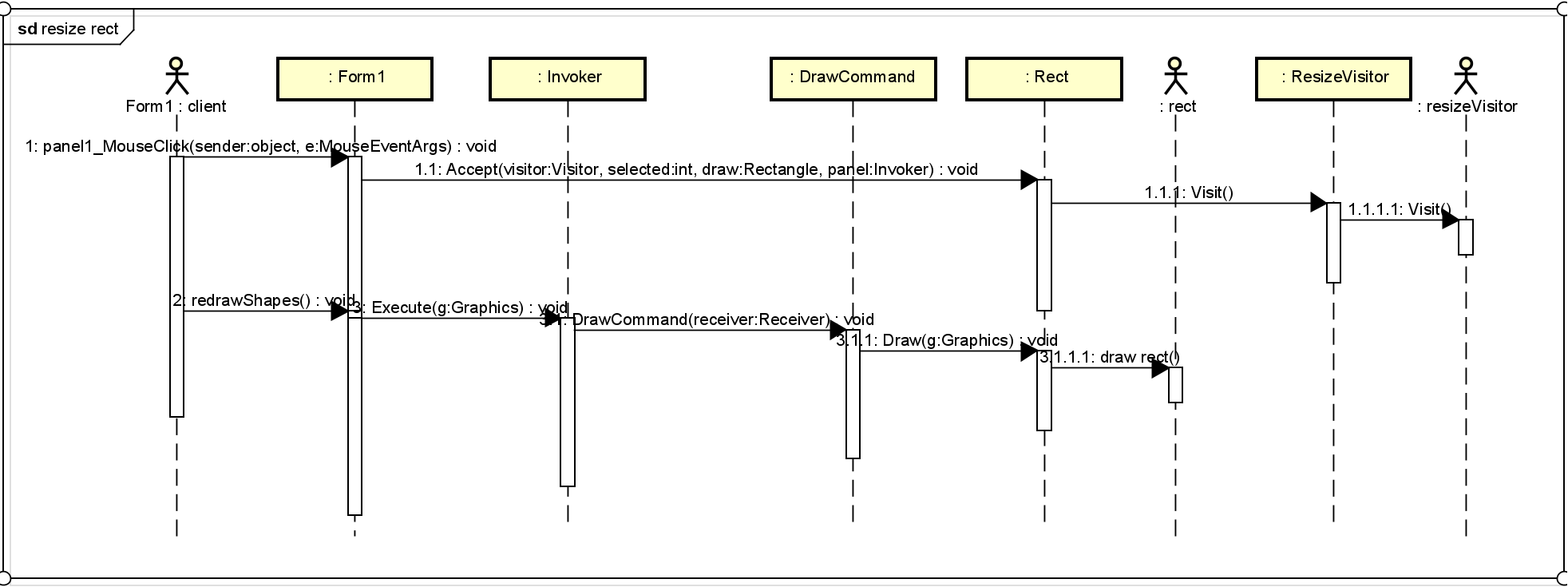


## Toestandsdiagram

Toestand is niet veranderd in tegenstelling tot stap 3. Voor diagram zie toestandsdiagram stap 2.

## Tijdsvolgordediagram





# Stap 5

## Klassediagram

## Toestandsdiagram

## Tijdsvolgordediagram

# Stap 6

## Klassediagram

## Toestandsdiagram

## Tijdsvolgordediagram