

Design Interactr

Group 26
Iteration 2nd
By: Afraz Salim

Contents

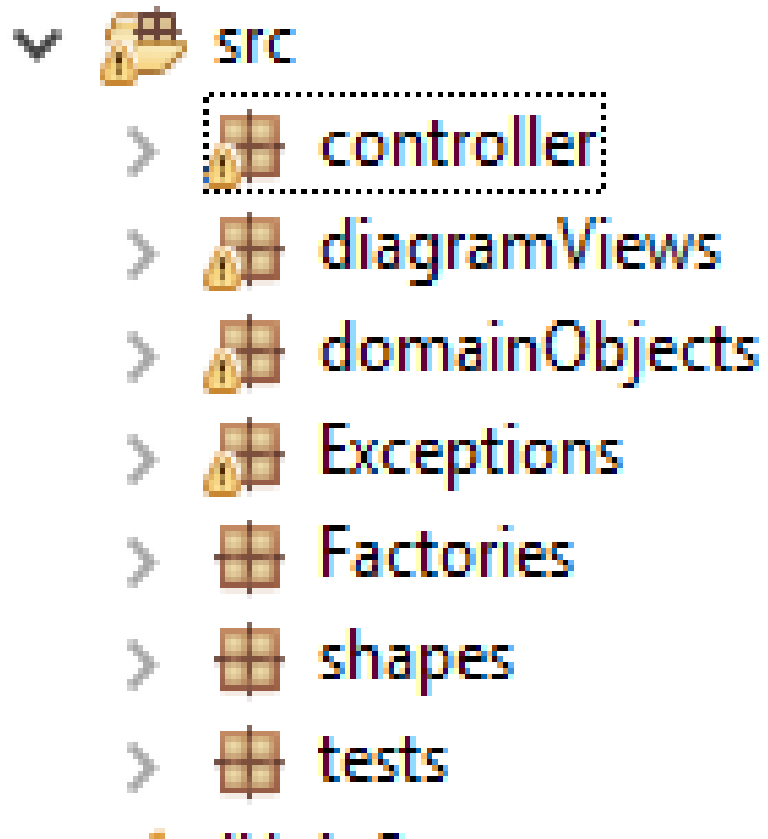
- Create New Interaction
- Create New Diagram
- Move SubWindow
- Resize SubWindow
- Activate SubWindow
- Close SubWindow

□ Testing

- Eclemma
- JUnit tests

➤ First Iteration Sequence Diagrams (due to redesign)

Package Structure:

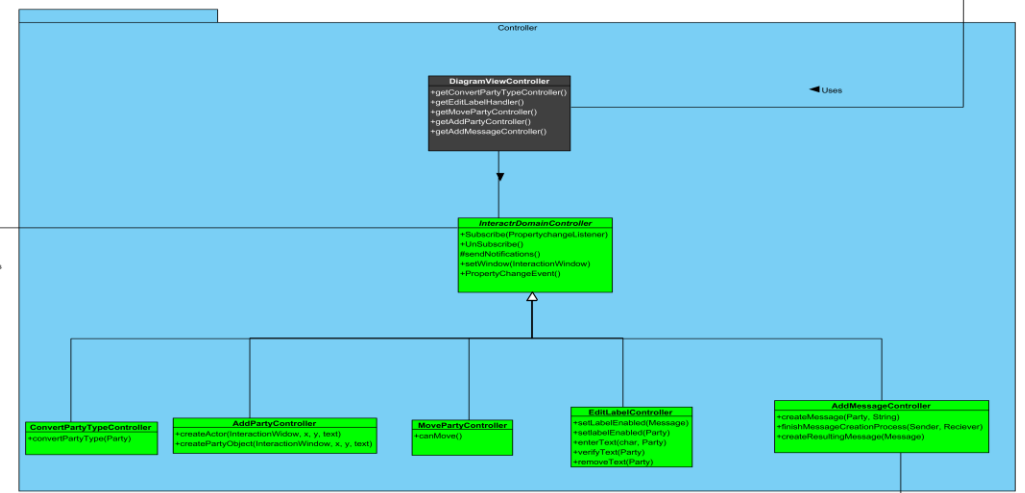
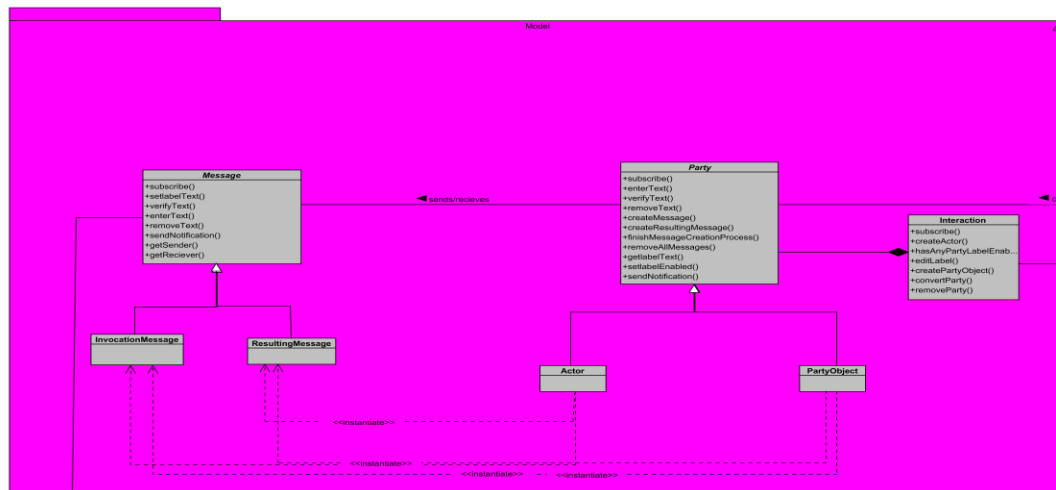
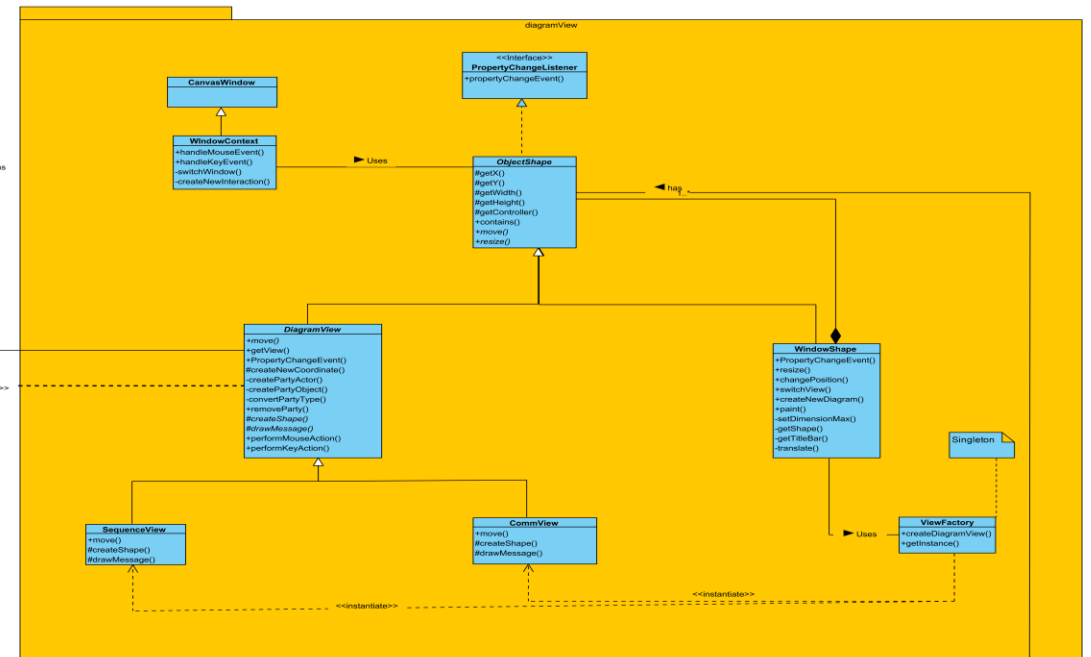
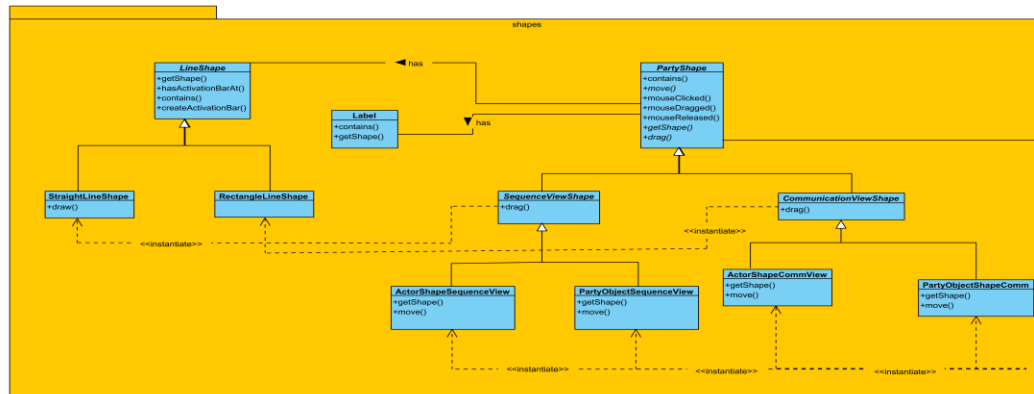


Model View and Controller's Overview:

View

Controller

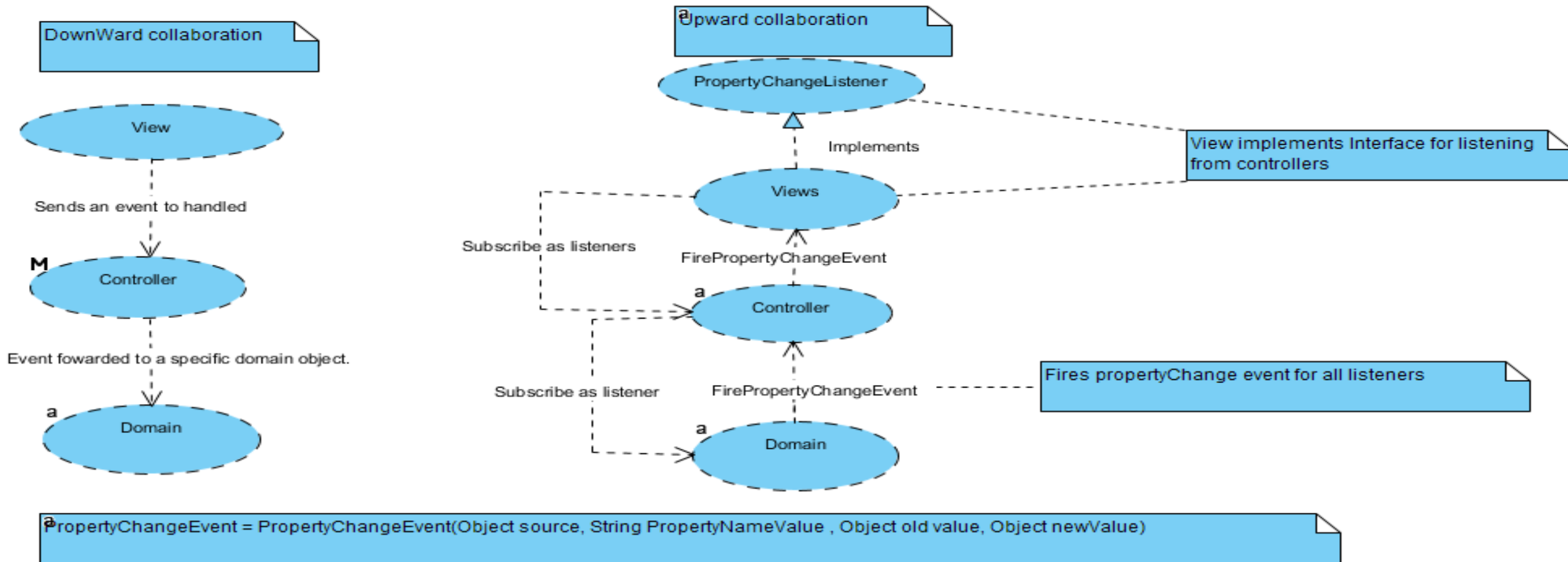
Model



Collaboration between View and Model:

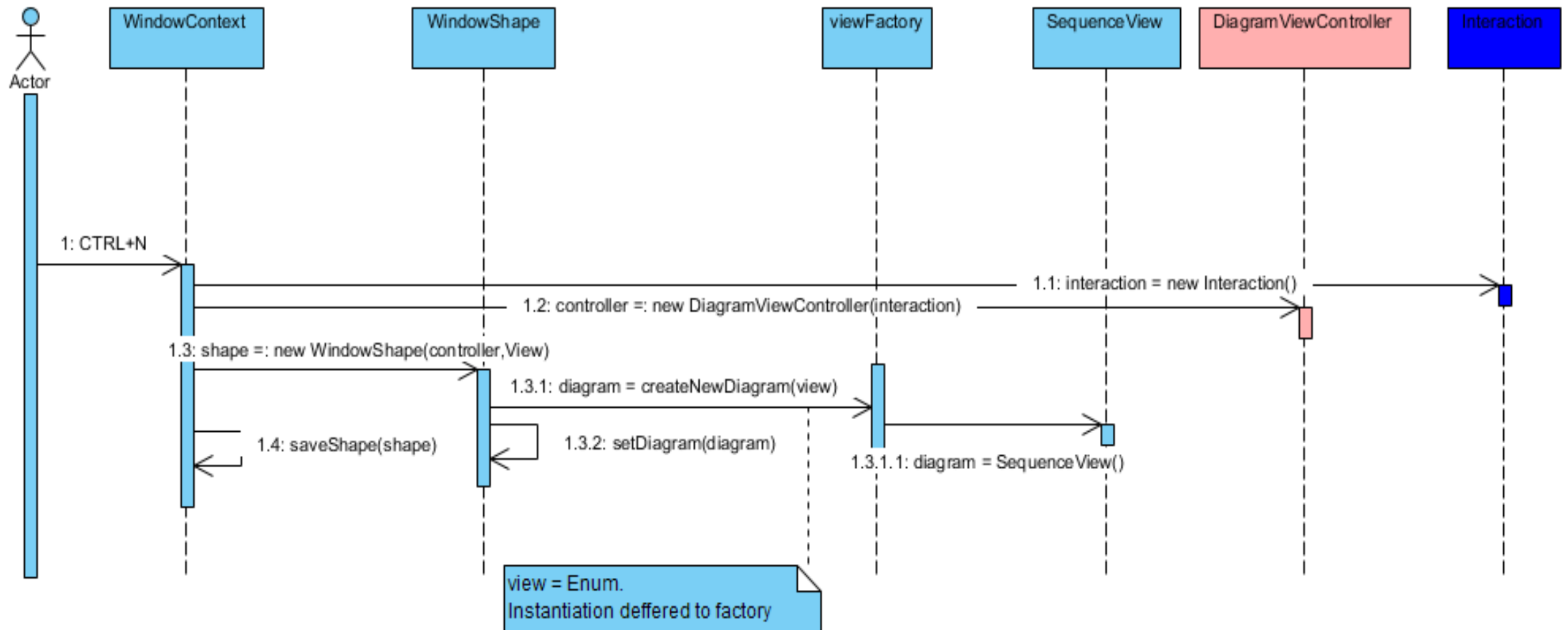
Observer design pattern:

(java.beans.PropertyChangeEvent)

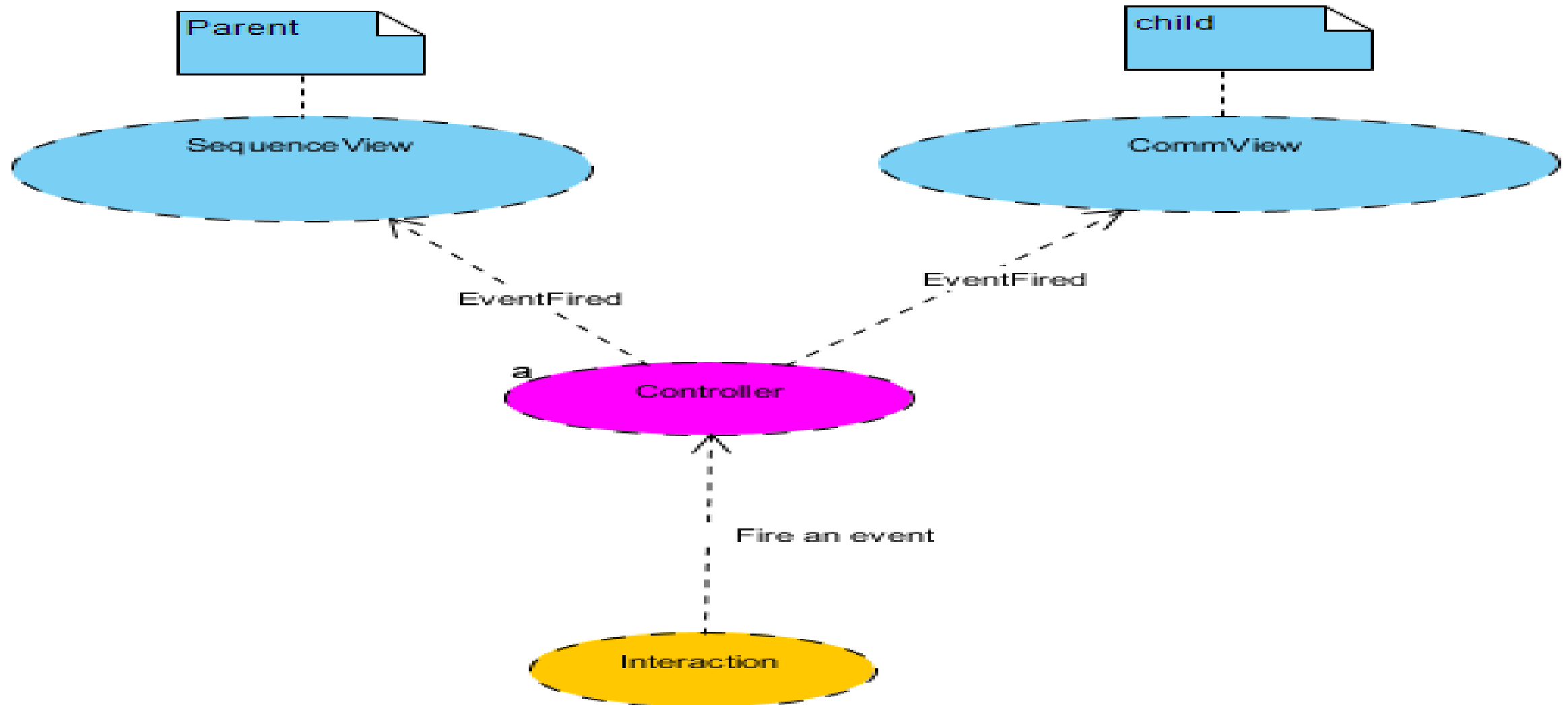


- Create New Interaction:
WindowContext responsible for initializing domain
interaction, controller and view.
Factory Method Design pattern.

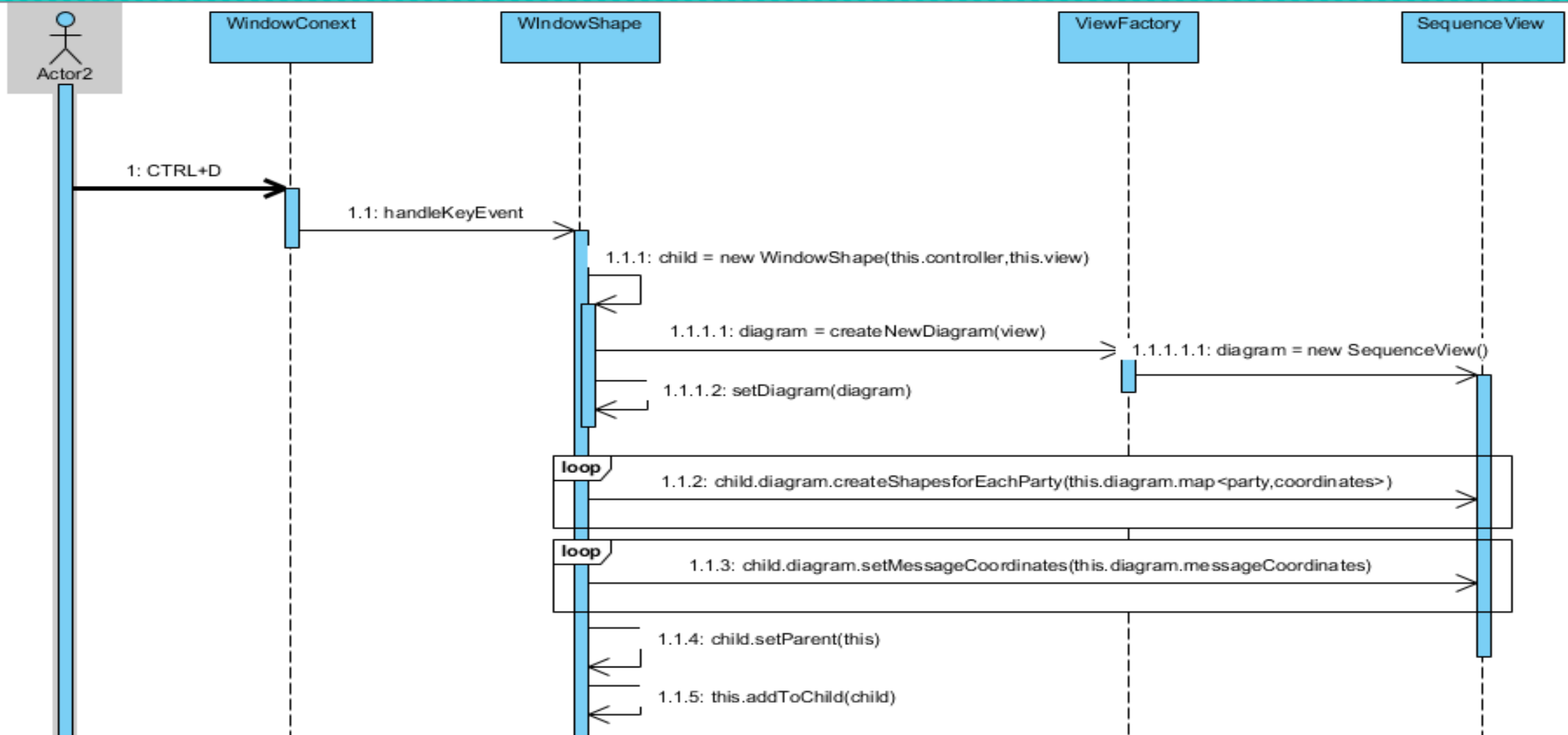
createNewInteraction



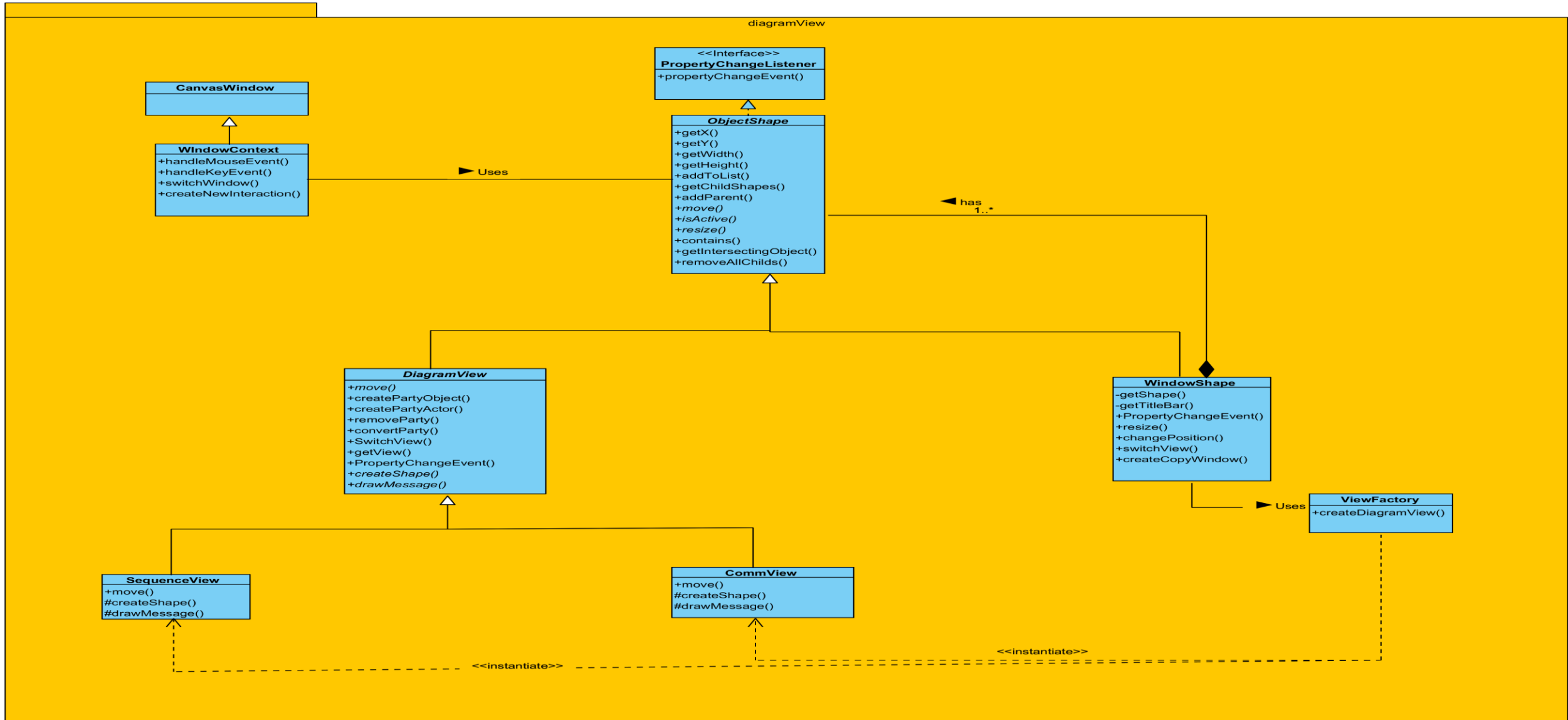
- **Create New Diagram:**
Concept: Same interaction with different Views by sharing same controller among all childs.



➤ Creating New Diagram: Composite design pattern.

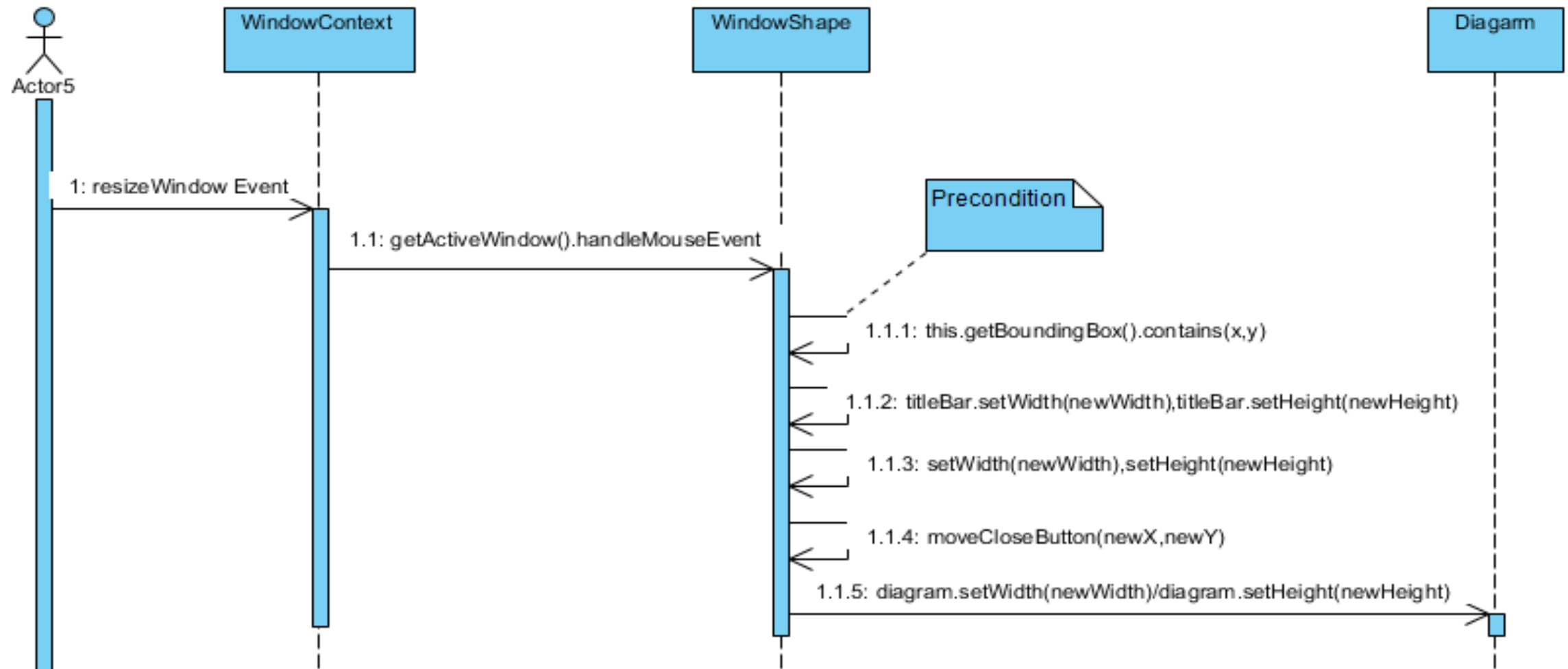


Overview of diagram and it's outer frame. Composite design pattern.

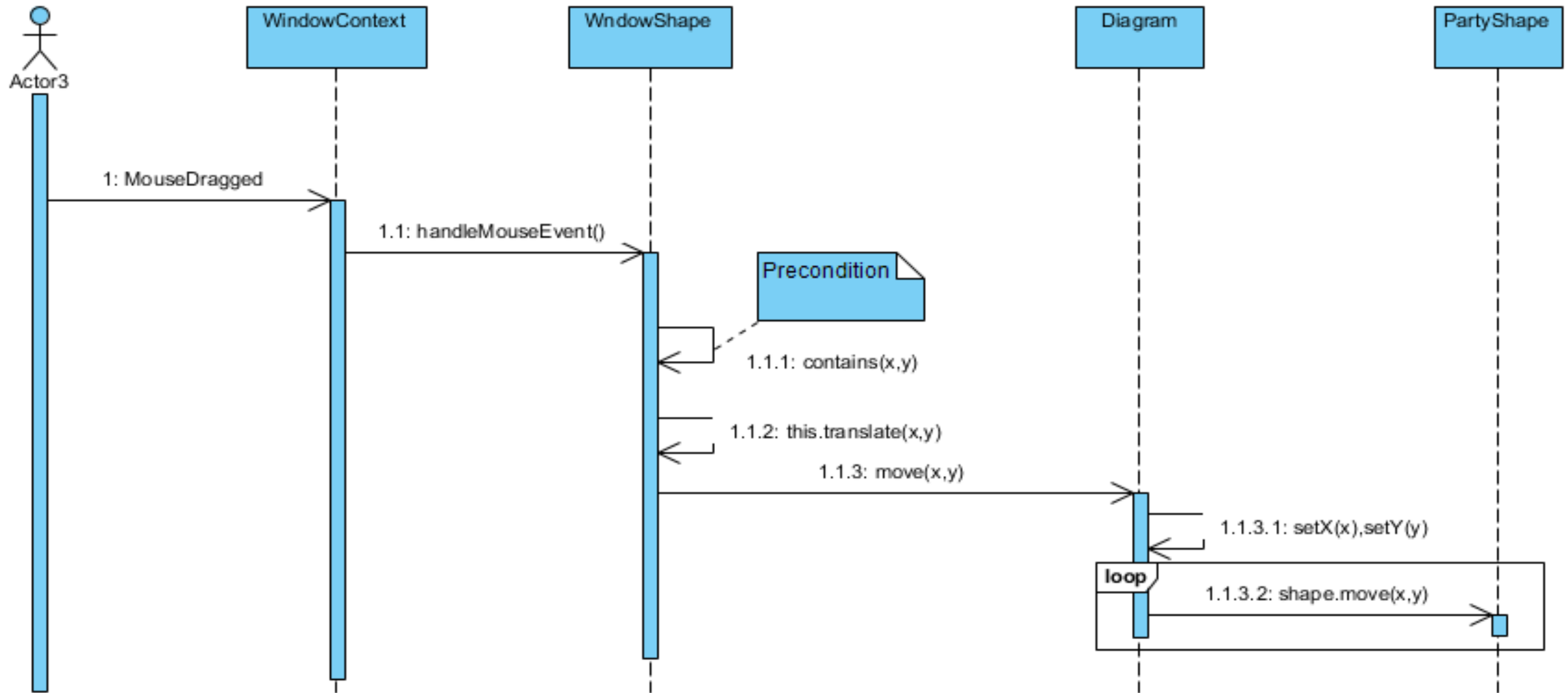


➤ Resize SubWindow:

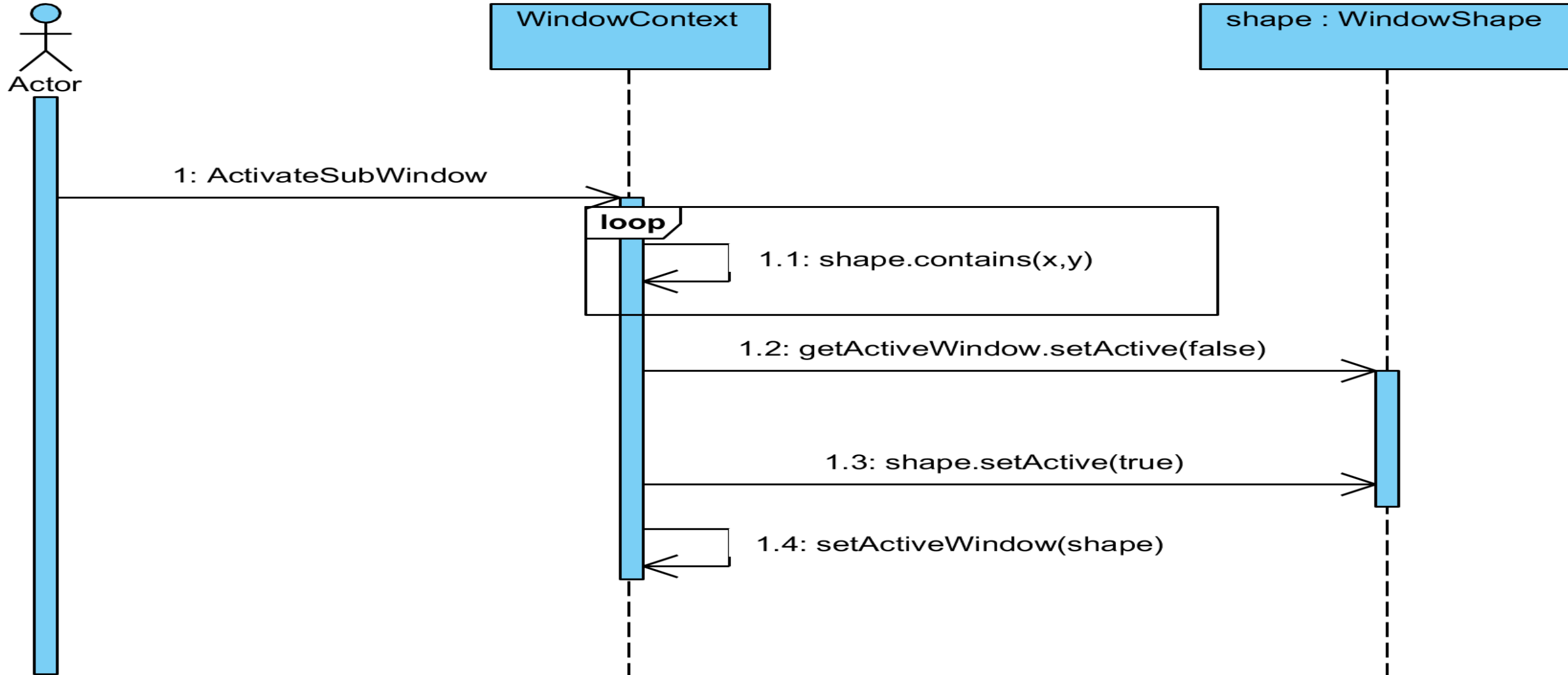
Resize Window



➤ Move window:

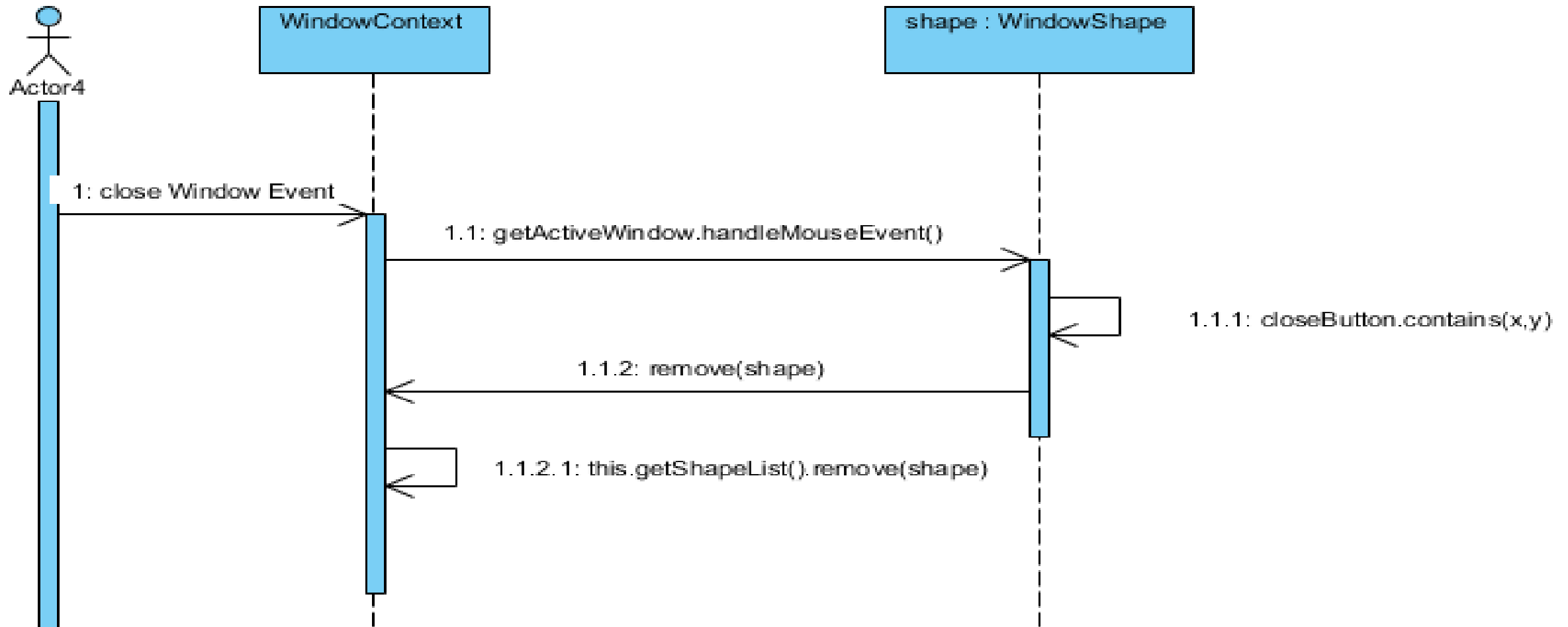


➤ Activate SubWindow:



➤ Close subwindow:





















































close SubWindow





Tests and Coverage.

Tests: Eclemma

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
▼  Projects	 63.9 %	8,695	4,906	13,601
▼  src	 63.9 %	8,695	4,906	13,601
▼  diagramViews	 42.7 %	2,609	3,498	6,107
>  SequenceView.java	 22.6 %	314	1,074	1,388
>  CanvasWindow.java	 1.7 %	16	935	951
>  CommView.java	 34.6 %	294	556	850
>  WindowShape.java	 51.9 %	588	546	1,134
>  DiagramView.java	 79.9 %	915	230	1,145
>  WindowContext.java	 67.0 %	240	118	358
>  View.java	 70.0 %	49	21	70
>  ObjectShape.java	 89.0 %	105	13	118
>  MouseAction.java	 92.9 %	65	5	70
>  Coordinate.java	 100.0 %	23	0	23
>  shapes	 49.0 %	1,048	1,089	2,137
▼  domainObjects	 87.7 %	965	135	1,100
>  Party.java	 87.7 %	426	60	486
>  Interaction.java	 81.0 %	226	53	279
>  Message.java	 93.0 %	293	22	315
>  Actor.java	 100.0 %	5	0	5
>  InvocationMessage.java	 100.0 %	5	0	5
>  PartyObject.java	 100.0 %	5	0	5
>  ResultingMessage.java	 100.0 %	5	0	5
>  controller	 74.3 %	379	131	510
>  Exceptions	 12.5 %	4	28	32
>  tests	 99.5 %	3,654	20	3,674
>  Factories	 87.8 %	36	5	41

Code with huge non-covered percentage: Shapes Package:

```
@Override
public Shape getShape() {
    GeneralPath path = new GeneralPath();
    path.moveTo(this.getX(), this.getY());
    path.lineTo(this.getX()+this.getWidth(),
    path.lineTo(this.getX()+this.getWidth(),
    path.lineTo(this.getX(), this.getY()+this.getHeight());
    path.lineTo(this.getX(), this.getY());
    path.moveTo(this.getX()+this.getWidthBound(), this.getY());
    path.lineTo(this.getX()+this.getWidth()-this.getWidthBound(), this.getY());
    path.lineTo(this.getX()+this.getWidthBound(), this.getY());
    path.moveTo((this.getX()+this.getWidthBound()+this.getWidthBound()), this.getY());
    path.lineTo(this.getX()+this.getWidth()-this.getWidthBound(), this.getY());
    path.moveTo((this.getX()+this.getWidthBound()+this.getWidthBound()), this.getY());
    path.lineTo(this.getX()+this.getWidth()-this.getWidthBound(), this.getY());
    path.moveTo((this.getX()+this.getWidthBound()+this.getWidthBound()), this.getY());
    path.lineTo(this.getX()+this.getWidthBound(), this.getY());
    path.moveTo((this.getX()+this.getWidthBound()+this.getWidthBound()), this.getY());
    path.lineTo(this.getX()+this.getWidth()-5, this.getY());
    path.moveTo((this.getX()+this.getWidthBound()+this.getWidthBound()), this.getY());
    path.lineTo(this.getX()+5, this.getY()+this.getHeight());
    path.closePath();
    return path;
}
```

```
@Override
public Shape getShape() {
    GeneralPath path = new GeneralPath();
    path.moveTo(this.getX(), this.getY());
    path.lineTo(this.getX()+this.getWidth(), this.getY());
    path.lineTo(this.getX()+this.getWidth(), this.getY()+this.getHeight());
    path.lineTo(this.getX(), this.getY());
    path.lineTo(this.getX(), this.getY());
    path.moveTo(this.getX()+this.getWidthBound(), this.getY());
    path.lineTo(this.getX()+this.getWidth()-this.getWidthBound(), this.getY());
    path.lineTo(this.getX()+this.getWidthBound(), this.getY());
    path.moveTo((this.getX()+this.getWidthBound()+this.getWidthBound()), this.getY());
    path.lineTo(this.getX()+this.getWidth()-this.getWidthBound(), this.getY());
    path.lineTo(this.getX()+this.getWidthBound(), this.getY());
    path.moveTo((this.getX()+this.getWidthBound()+this.getWidthBound()), this.getY());
    path.lineTo(this.getX()+this.getWidth()-5, this.getY());
    path.moveTo((this.getX()+this.getWidthBound()+this.getWidthBound()), this.getY());
    path.lineTo(this.getX()+5, this.getY()+this.getHeight());
    return path;
}
```

```
/**
 * A getter to get the shape of the actor in scene
 */
@Override
public Shape getShape() {
    GeneralPath path = new GeneralPath();
    path.moveTo(this.getX(), this.getY());
    path.lineTo(this.getX()+this.getWidth(), this.getY());
    path.lineTo(this.getX()+this.getWidth(), this.getY()+this.getHeight());
    path.lineTo(this.getX(), this.getY());
    return path;
}
```

```
/**
 * A getter to get the shape of actor in scene
 */
@Override
public Shape getShape() {
    GeneralPath path = new GeneralPath();
    path.moveTo(this.getX(), this.getY());
    path.lineTo(this.getX()+this.getWidth(), this.getY());
    path.lineTo(this.getX()+this.getWidth(), this.getY()+this.getHeight());
    path.lineTo(this.getX(), this.getY()+this.getHeight());
    path.lineTo(this.getX(), this.getY());
    return path;
}
```


Diagram view un-covered code:

```
@Override
protected void paint(Graphics g) {
    super.paint(g);
    Graphics2D g2 = (Graphics2D)g;
    for(PartyShape shape : this.getShapeList()) {
        g2.setColor(Color.GREEN);
        g2.draw(shape.getShape());
        g2.setColor(Color.LIGHT_GRAY);
        g2.draw(shape.getLine().getShape());
        g2.setColor(Color.LIGHT_GRAY);
        g2.fill(shape.getLine().getShape());
        g2.setColor(Color.orange);
        g2.setFont(new Font("TimesRoman", Font.PLAIN, 12));
        if(shape.getHasActiveLabel()) {
            g2.setColor(Color.RED);
            g2.drawString(shape.getLabel().getText(),
                shape.getLine().getX() + shape.getLine().getWidth() / 2,
                shape.getLine().getY() + shape.getLine().getHeight() / 2);
        }
        else
            g2.drawString(shape.getLabel().getText(),
                shape.getLine().getX() + shape.getLine().getWidth() / 2,
                shape.getLine().getY() + shape.getLine().getHeight() / 2);
        g2.setColor(Color.WHITE);
        shape.getLine().draw(g2);
        g2.setColor(Color.WHITE);
        this.drawMessage(g2);
        this.move(this.getX(), this.getY()); // Bug
    }
}
```

```
private void drawMessage(Graphics g, Message message) {
    Graphics2D g2 = (Graphics2D)g;
    PartyShape sender = this.getAssociationList().get(message.getSender());
    if(message.equals(getMessageUnderConstruction()) || message.getSender() == null)
        return;
    PartyShape reciever = this.getAssociationList().get(message.getReceiver());
    if(this.getMessageMap().get(message) == null)
        this.createNewCoordinates(message);
    Coordinate temp = this.getMessageMap().get(message);
    this.adjustCoordinates(sender, reciever, temp);
    int x = 0;
    int xSecond = 0;
    if(sender.getX() > reciever.getX()) {
        x = sender.getLine().getX();
        xSecond = reciever.getLine().getX() + reciever.getLine().getWidth();
    }
    else
    {
        x = sender.getLine().getX() + sender.getLine().getWidth();
        xSecond = reciever.getX();
    }
    int xCo = this.getXCo(sender, reciever);
    g.drawLine(x, temp.getY(), xSecond, temp.getY());
    this.saveMessageLabelCoords(xCo, temp.getY() + (temp.getY() - temp.getY()) / 2);
    if(message.isActive())
        g.setColor(Color.RED);
    this.drawText(g2, message, xCo, temp.getY() + (temp.getY() - temp.getY()) / 2);
    g.setColor(Color.WHITE);
}
```

```
private Shape getResultingShape(PartyShape sender, PartyShape reciever) {
    int distance = this.getDistance(sender.getLine().getX(), reciever.getLine().getX());
    int distanceSecond = this.getDistance(reciever.getLine().getX(), sender.getLine().getX());
    GeneralPath path = new GeneralPath();
    int arrow = this.getArrow(sender.getLine().getX() + distance * sender.getLine().getX(),
        reciever.getLine().getX() + distance * reciever.getLine().getX());
    path.moveTo(sender.getLine().getX() + distance * sender.getLine().getX(),
        sender.getLine().getY() + distance * sender.getLine().getY());
    path.lineTo(reciever.getLine().getX() + distance * reciever.getLine().getX(),
        reciever.getLine().getY() + distance * reciever.getLine().getY());
    path.moveTo(reciever.getLine().getX() + distance * reciever.getLine().getX(),
        reciever.getLine().getY() + distance * reciever.getLine().getY());
    path.lineTo(sender.getLine().getX() + distance * sender.getLine().getX(),
        sender.getLine().getY() + distance * sender.getLine().getY());
    return path;
}
```

```
private void drawInvocationMessage(Graphics g, Message message) {
    Graphics2D g2 = (Graphics2D)g;
    float[] dashes = {1,1,1};
    BasicStroke stroke = new BasicStroke(1, BasicStroke.CAP_SQUARE, BasicStroke.DASH_DASH);
    Message resulting = getMessageAssociationMap().get(message);
    Coordinate temp = getMessageMap().get(resulting);
    PartyShape sender = this.getAssociationList().get(message.getSender());
    PartyShape reciever = this.getAssociationList().get(message.getReceiver());
    if(sender == null || reciever == null)
        return;
    int y = temp.getY() - (sender.getLine().getActivationBarHeight() / 2);
    int ySecond = temp.getY() - (reciever.getLine().getActivationBarHeight() / 2);
    int index = this.getIndex(sender);
    Shape shape = this.getResultingShape(sender, reciever, ySecond);
    Shape stroked = stroke.createStrokedShape(shape);
    g2.draw(stroked);
    int labelX = this.getLabelX(sender, reciever);
    if(message.isActive())
        g.setColor(Color.RED);
    this.saveLabelCoordinate(message, labelX, temp.getY() - sender.getLine().getActivationBarHeight() / 2);
    this.drawText(g, message, labelX, temp.getY() - sender.getLine().getActivationBarHeight() / 2);
    if(message.isActive())
        g.setColor(Color.RED);
    g.setColor(Color.WHITE);
}
```

➤ Time spent

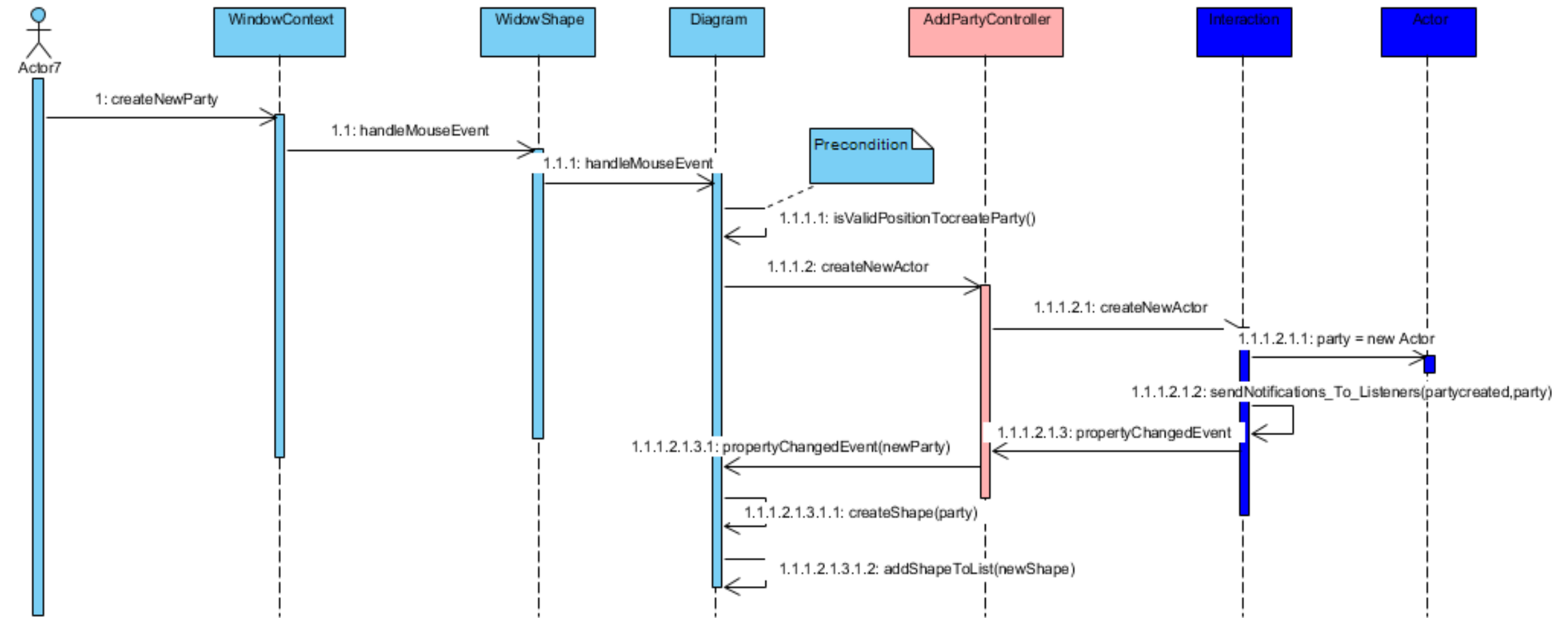
Afraz Salim	Week 1	Week 2	Week 3	Week 4	
Study	3-4 hours	2-3 hours	≈1 hours	0 hour	
Project-Work	10 hours	≈25	≈ 20	≈15	

❖ First Iteration Sequence Diagrams: Contents

- Add New Party
- Convert Party Type
- Delete Party Element
- Add Message
- Switch View

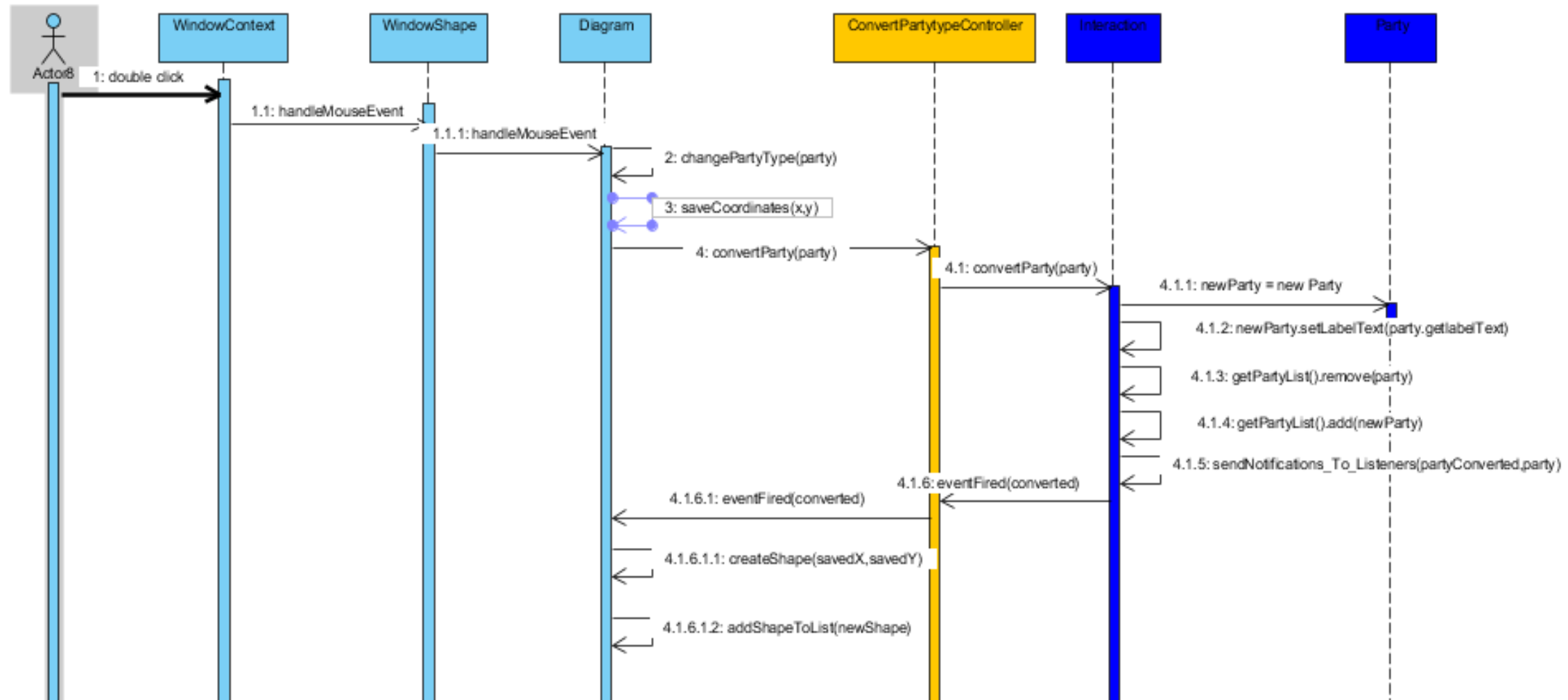
➤ Add New Party:

createNewParty



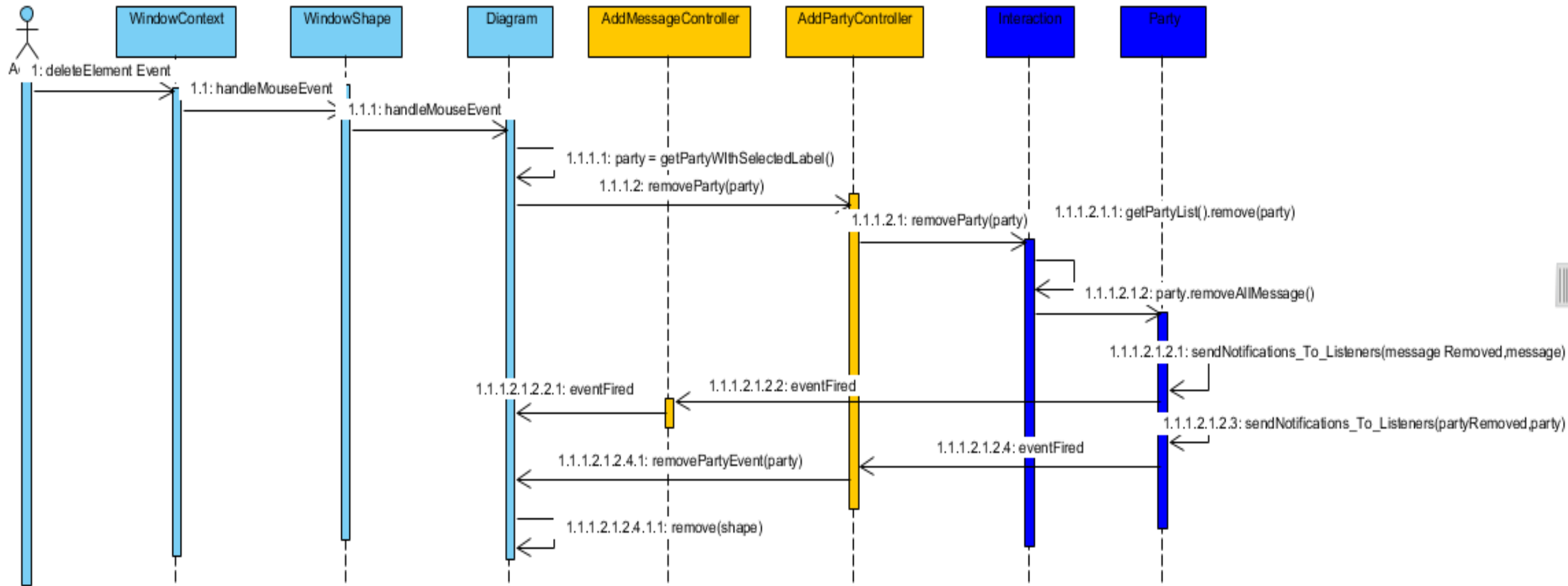
➤ Convert Party Type:

setPartyType



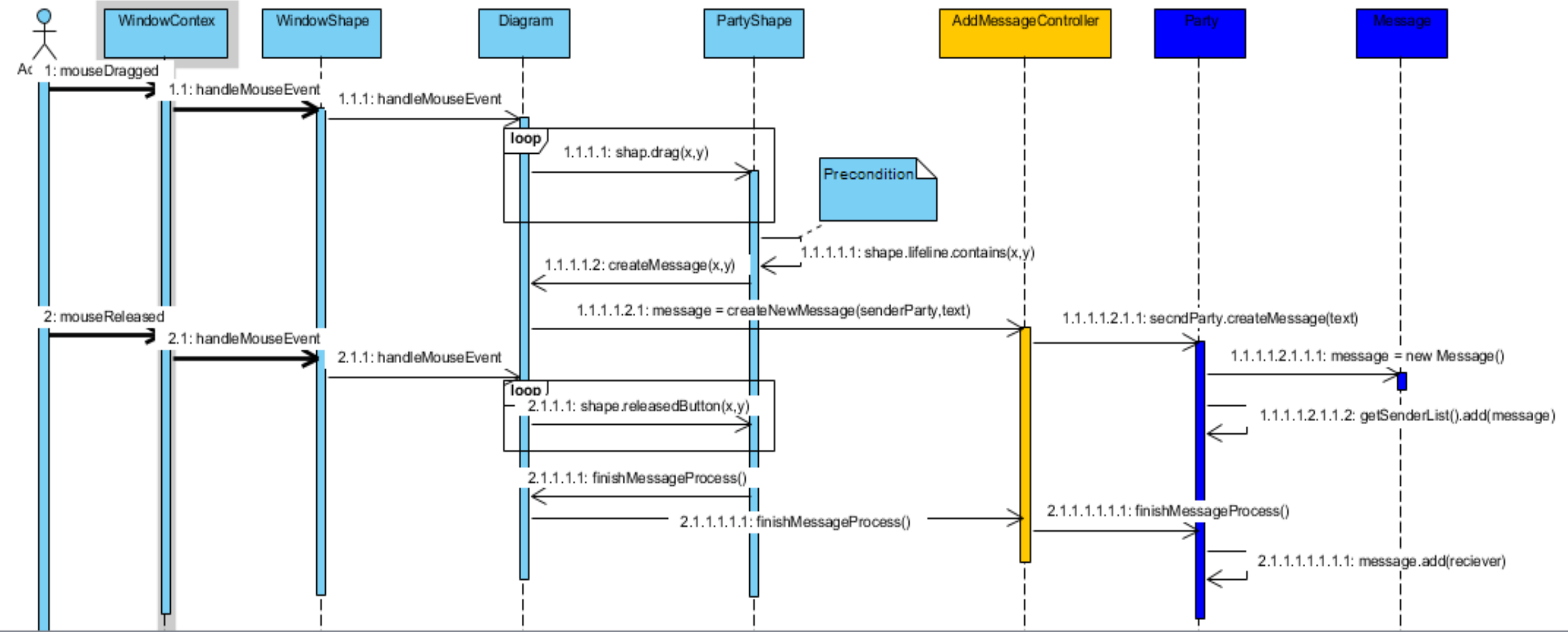
➤ Delete Party Element:

Sequence Diagram 1



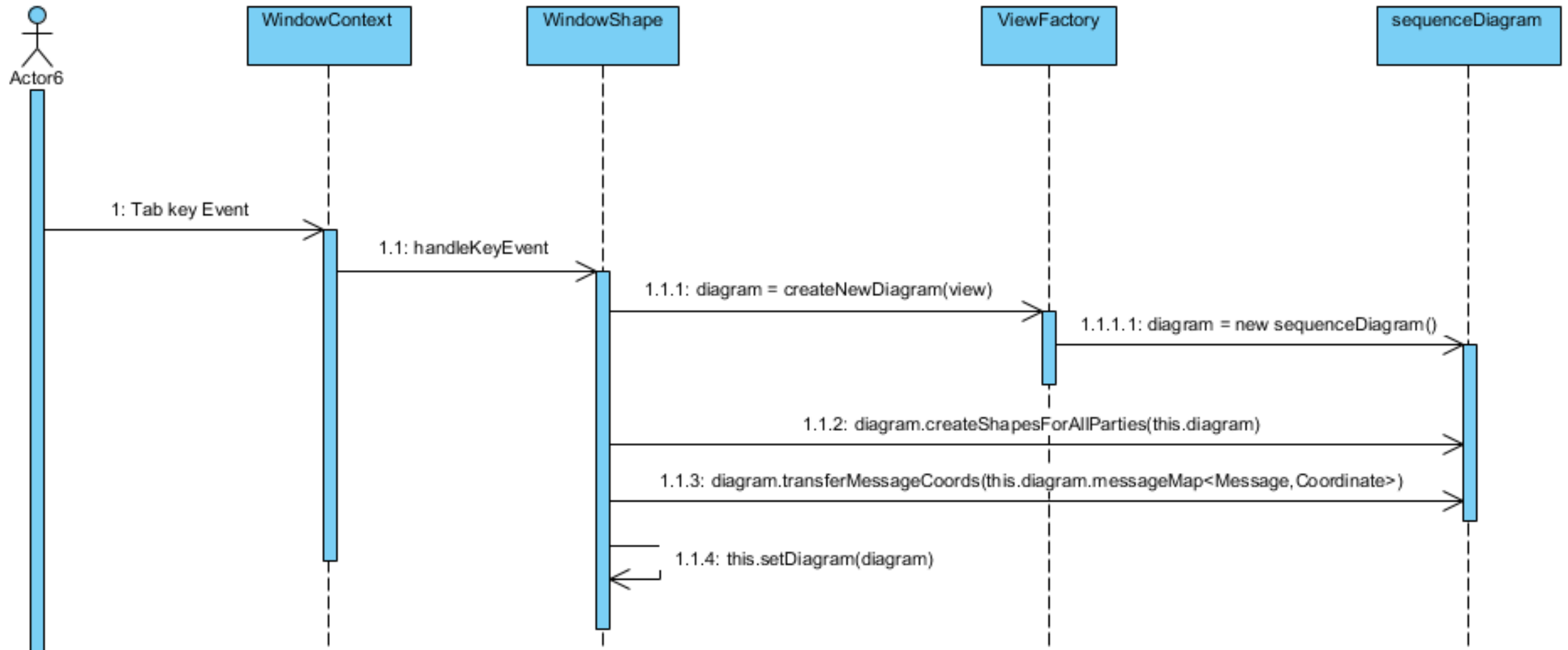
➤ Add Message:

sd addMessage



➤ Switch-View:

SwitchViewType



Thanks for your attention.