Hands-On 3
Design Patterns

arnaud.nauwynck@gmail.com

February 2023

Outline

Objectives

Design in UML a Drawing Application

Recognize/Use many Design Patterns

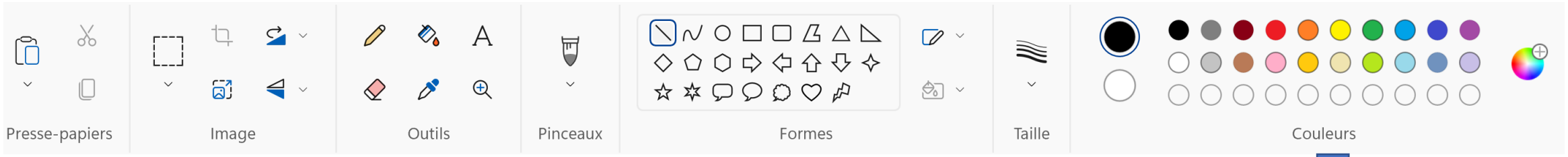
Initial Step : Design the Core Domain

Exercise 1: Design the Core Domain classes

Draw UML classes diagram for

Line, Text, Rectangle, Circle,
Curve = list of CurveElement (segment,arc,bezier curve)

Drawing Toolbar



Change color for next drawing
Cf next Exercise 2,3 ..

Change mouse behavior in canvas
Example:
« Line » : mouse button down … move … button up
Cf next Exercise 4,..

Changing Color Selection

The simplest that possibly works…

```
buttonBlue.setActionListener( (e) -> setCurrentColorSelection(blue) );
buttonWhite.setActionListener( (e) -> setCurrentColorSelection(white) );
buttonRed.setActionListener( (e) -> setCurrentColorSelection(red) );
```

Problem :
How to record the history of changes ?
How can you UNDO (Cntrl+Z) then REDO (Ctrl+Y) any previous action ?

NOTICE :

- in Microsoft Paint app, you can not undo « global color selection change », you can undo only drawings
- In Microsoft PowerPoint, there is 1 color per drawing element, and 1 default color per drawing element

Exercise 2 : « ColorChangeCommand » … minimalist Command Design Pattern

Complete following code (marked as « … »)
And draw as UML a minimalist Command design pattern

```
abstract class Command {
      ….
}

class ColorSelectionChangeCommand extends Command {
      ….
}
```

Exercise 3 : enrich the Command pattern for Undo

On Button Click 

Get previous color selection
+ create new « ColorChangeCommand »
with previous & new color
+ execute command
+ add execution in command history

```
abstract class Command {
    abstract void execute();
    abstract boolean canUndo();
    abstract void undo();
}
class ColorChangeCommand extends .. {
    Color previousColor, color;
     …
}
```

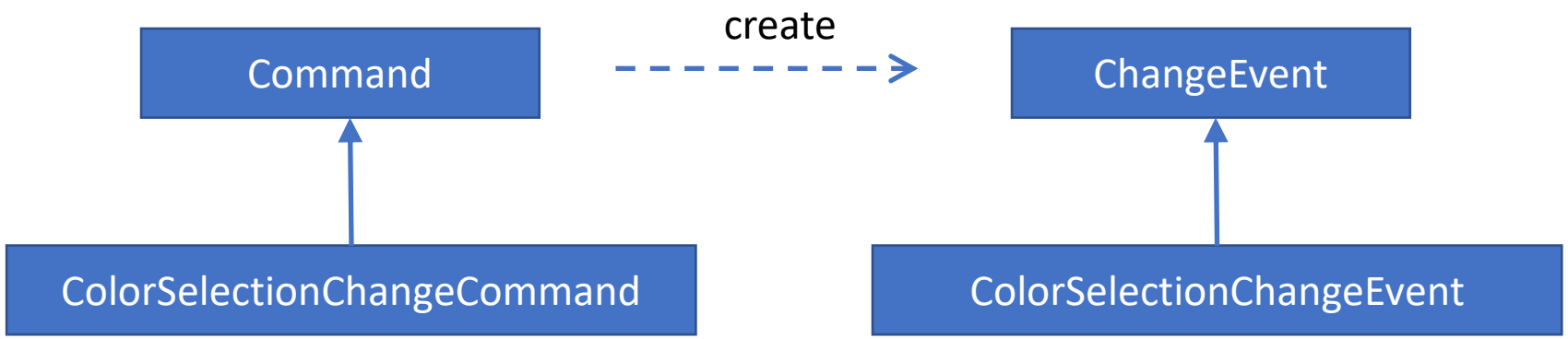
What happen with UNDO, then REDO ??
Draw UML Sequence Diagram
Draw UML class diagram with CommandHistory, Command (, Execution?)

… alternative on Undo Command, using Memento design pattern

```
abstract class Memento {}

abstract class Command {
    abstract Memento execute();
    abstract boolean canUndo(Memento m);
    abstract void undo(Memento m);
}
```

… Alternative (CQRS / EventSourcing Architecture)
Command = Factory for ChangeEvent



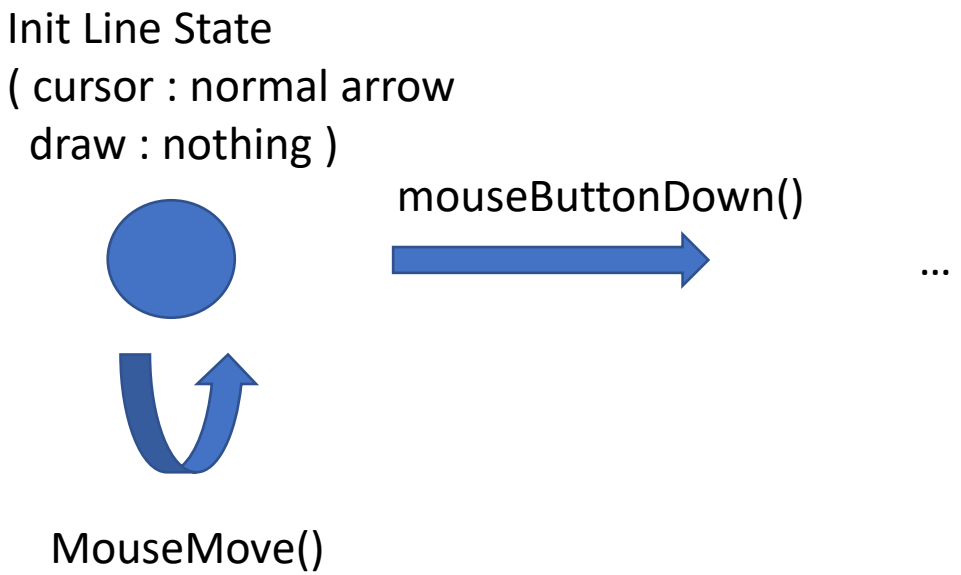
Contains only the new color
to be applied later
… not the « current »

Contains both the new color,
and the previous, when it changed

Exercise 3 : State design-pattern … handling mouse events, state transition

Change mouse behavior in canvas
Example:
« Line » : mouse button down … move … button up
Cf next Exercise 4,..

Exercise 3 : Draw UML State Automaton Diagram, for drawing « Line »

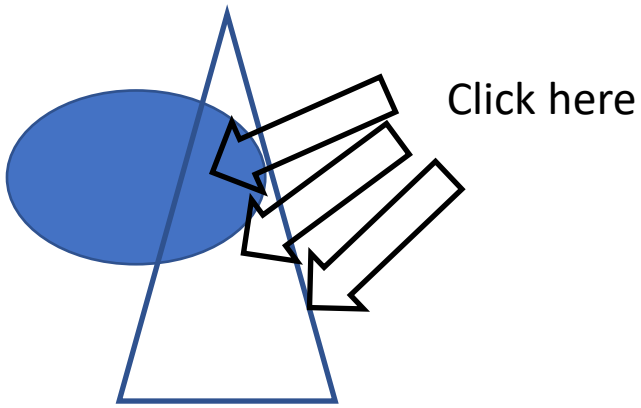


Exercise 4 : Handling Selection … Chain Of Responsability, Mediator,

In « Selection Mode », what happen when you click in the canvas…

There can be several elements on top of each others,
Element can be Filled / Empty
…

Model behavior as UML class diagram / UML sequence diagram



Exercise 2: Enrich the Core Domain classes

Enrich the domain classes with the
- « Composite » design pattern
- « Decorator » design pattern

Explain what it allows in terms of graphics in the application

Exercise 3: Embedding Image / other document

Enrich the domain classes with the

- « Adapter » design pattern, for Image
- « Proxy » design pattern

Explain what it allows in terms of graphics in the application

(Optional) Exercise 5 : Bridge .. To Mathematical Expression

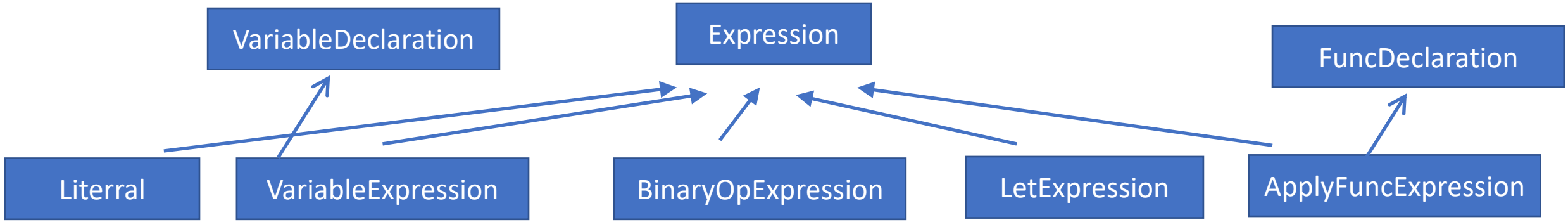
Design in UML for « Bridging » all int/double values, by an abstract Mathematical Expression

For example

Intead of having 2 « aligned » rectangles
Rect1 = Rectangle(left=10, width=100, top=20, height=50)
Rect2 = Rectangle(left=10, width=10000, top=200, height=30)

=>
declare Expression sharedVariable1 = new Variable(10);
Rect1=Rectangle (leftExpression=sharedVariable1, ….)
Rect2=Rectangle (leftExpression=sharedVariable1, ….)

(Optional Exercise 5) Math Expression



Possible expression:
« 123 »
« 3.1415926 »

Given « declare x= 123 »
..
Possible expression:
« x » => resolved as 123

For standard operators +,-,*,/,%

Example
« 1 + x »
«2 * (3 + x) »

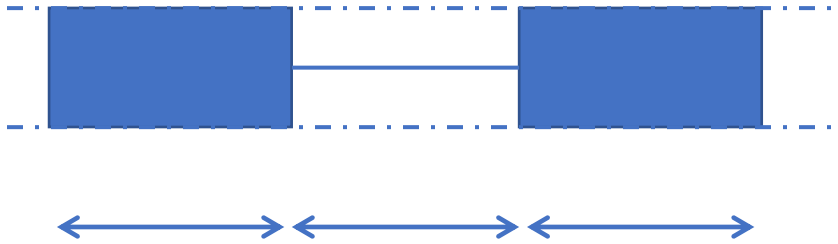
For local vars

Let x=123 in {
x + x + x
}

For calling function

sin(x)
sqrt(x)
max(0, x)

(Optional) Exercise 5: « Use » Domain Class. write sample grammar / file for drawing this



```
Document=
  declare var x = 0, var1Width = 30, var2Height = 10  {
   Composite {
       Rectangle( left: « x »,  width: « var1Width », ….),

       Line( x1= « x »,  y1:..,  x2=« x+var1Width », y2=  …)

       Rectangle(  left: « x+ 2 * var1Width »,  …)
    }
   }
  }
```

Exercise 6 : Model – View (– Controller) / Observer / Publish&Subscribe

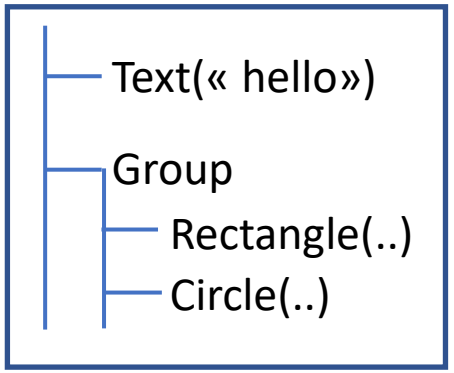
Design in UML Classes that a « Document » = Model
Can be viewed/edited in different UIs

1 UI Panel for Text edition (in a TreeView)
1 UI Panel for canvas drawing
1 UI Panel for zoom outline

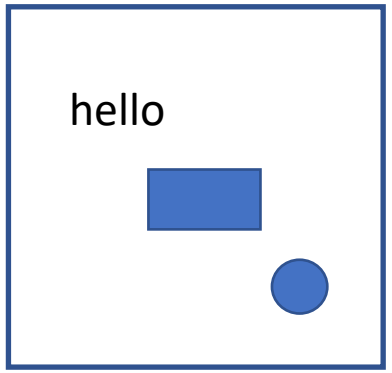
When the document changes, it emits a « changeEvent » … all subscribers are notified to « refresh »

Exercise 6 …

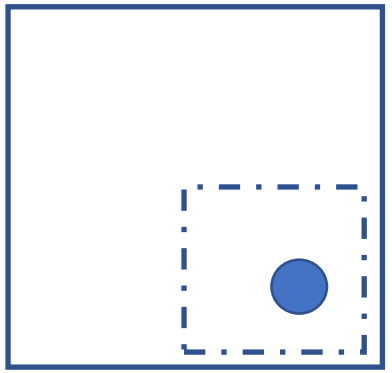
TreeView Editor



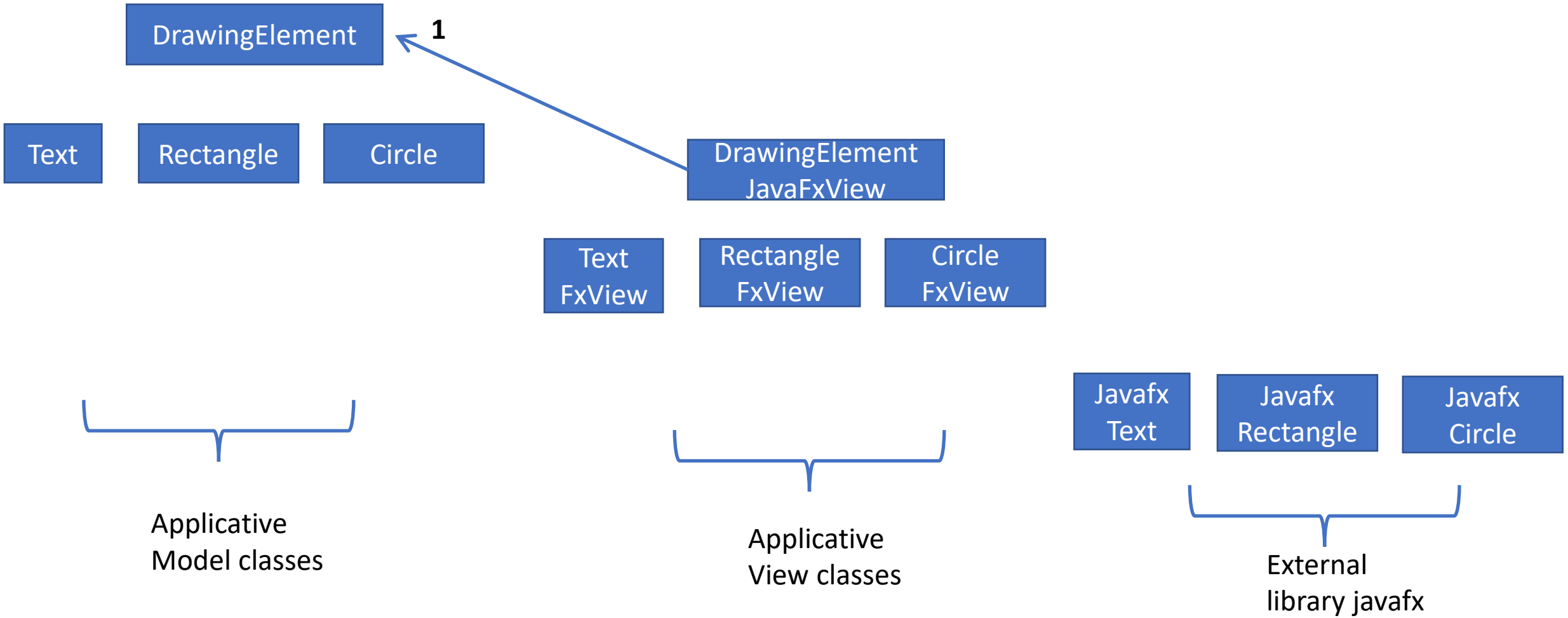
Canvas 1 Outline



Canvas 2 (Zooming)



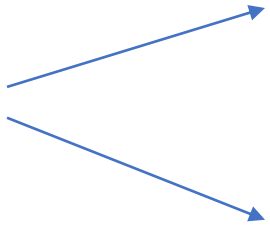
Exercise 7 : (MVC) View = Bridge to JavaFx Complete Links



Exercise 8 : Abstract Kit, Factory Classes

Draw UML classes diagram
showing « Kit » design-pattern + Factory Classes :

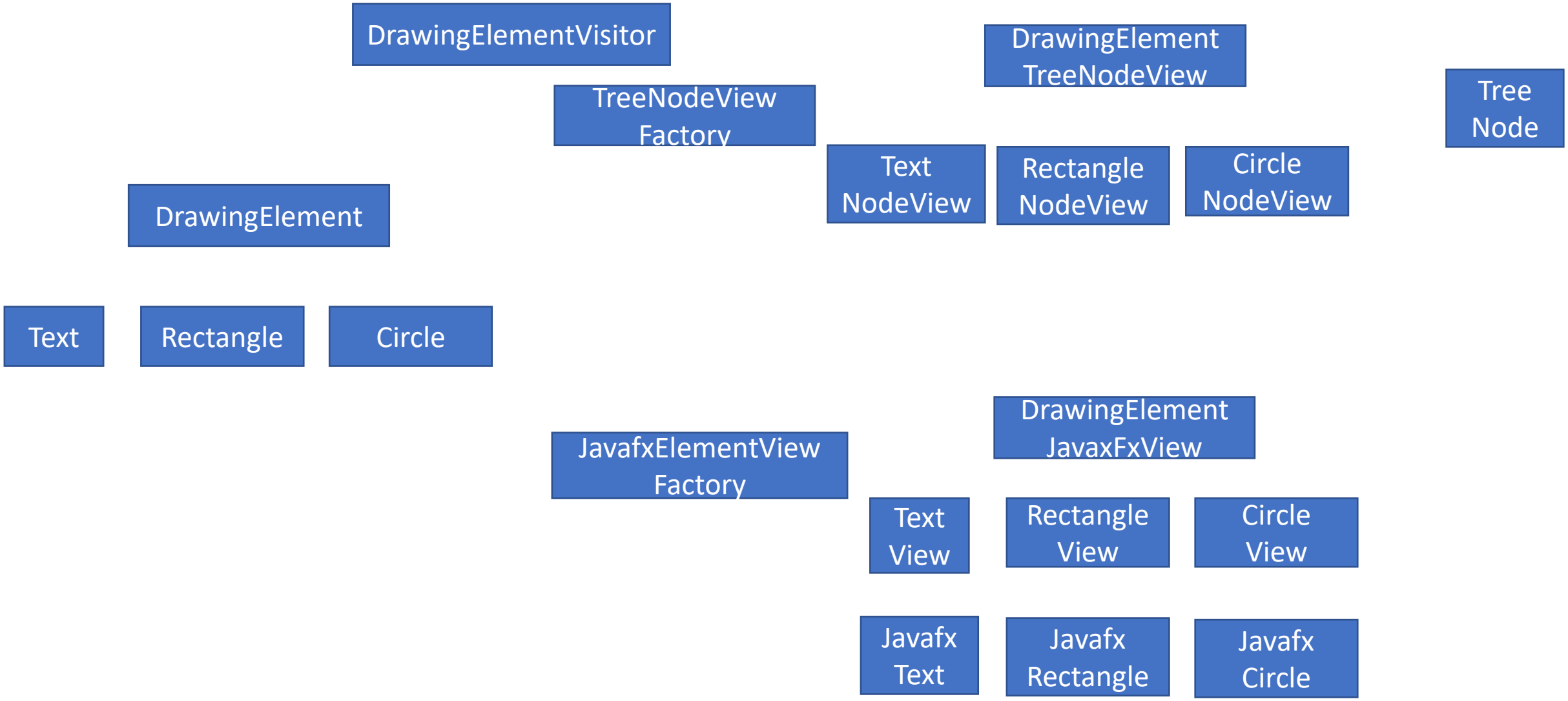
A graphical element (example « Rectangle »)



In a javafx view :
Create a « javafx.Rectangle » view adapter

In a TreeView :
create a « Rectangle » Node view adapter,
with child nodes
- node « x »,
- node « y »,
- node « width »,
- node « height »

Exercise 8 … complete links



Exercise 8 : Reminder.. Visitor Design-Pattern