Lecture 21

ECE 1145: Software Construction and Evolution

MiniDraw (CH 30)

Announcements

- Relevant Exercises: 30.4
- Code Swap 2 due Nov. 17 – same teams as Code Review 1
  - Share Iteration 7 before class
- Code Review 2 due Nov. 21
- Iteration 8 (last one!): Frameworks and MiniDraw due Dec. 12
  - Recommendations:
    - Week of Nov. 29: Frameworks (36.36), MiniDraw Integration (test out gradle tasks), Subject behavior (36.37), Observer updates (36.38)
    - Week of Dec. 6: Tool development (36.39-40, 42-44), SemiCiv GUI

Questions for Today

How do we use frameworks with compositional design?

Review: Frameworks

Frameworks provide **variability points** that let the developer customize the framework for a given application.

- Hot spots / frozen spots:
- Inversion of control
- Reuse of working code as well as reuse of design

What is MiniDraw?

MiniDraw is a **framework** that supports user interaction with 2D image-based graphics via mouse events

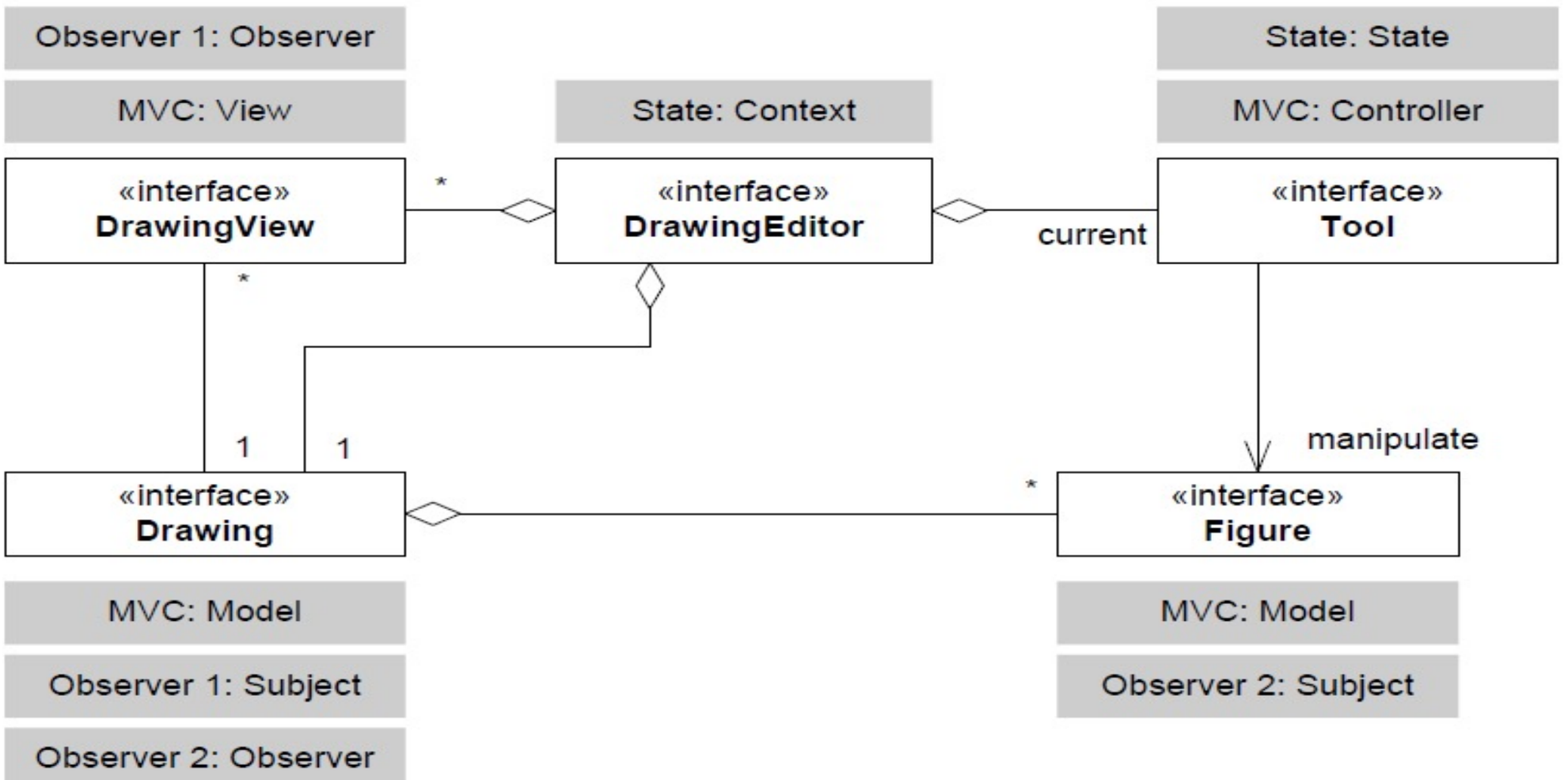
- Hot spots / frozen spots:
  - Customization is done by defining new tools, adding image files, or configuring the factory with proper implementations of MiniDraw's roles
- Inversion of control
  - When calling open(), it does all the processing of mouse events, calls your tool, draws your images at the correct times
- Reuse of working code as well as reuse of design
- Framework composition
  - MiniDraw + Java Swing
  - MiniDraw + HotCiv

What is MiniDraw?

MiniDraw + HotCiv

- Move units
- Perform unit actions
- Change production
- End turn

MiniDraw


Observer 1: Observer
State: State
MVC: View
State: Context
MVC: Controller
«interface»
DrawingView
*
«interface»
DrawingEditor
current
«interface»
Tool
*
1
1
manipulate
«interface»
Drawing
*
«interface»
Figure
MVC: Model
MVC: Model
Observer 1: Subject
Observer 2: Subject
Observer 2: Observer

MiniDraw

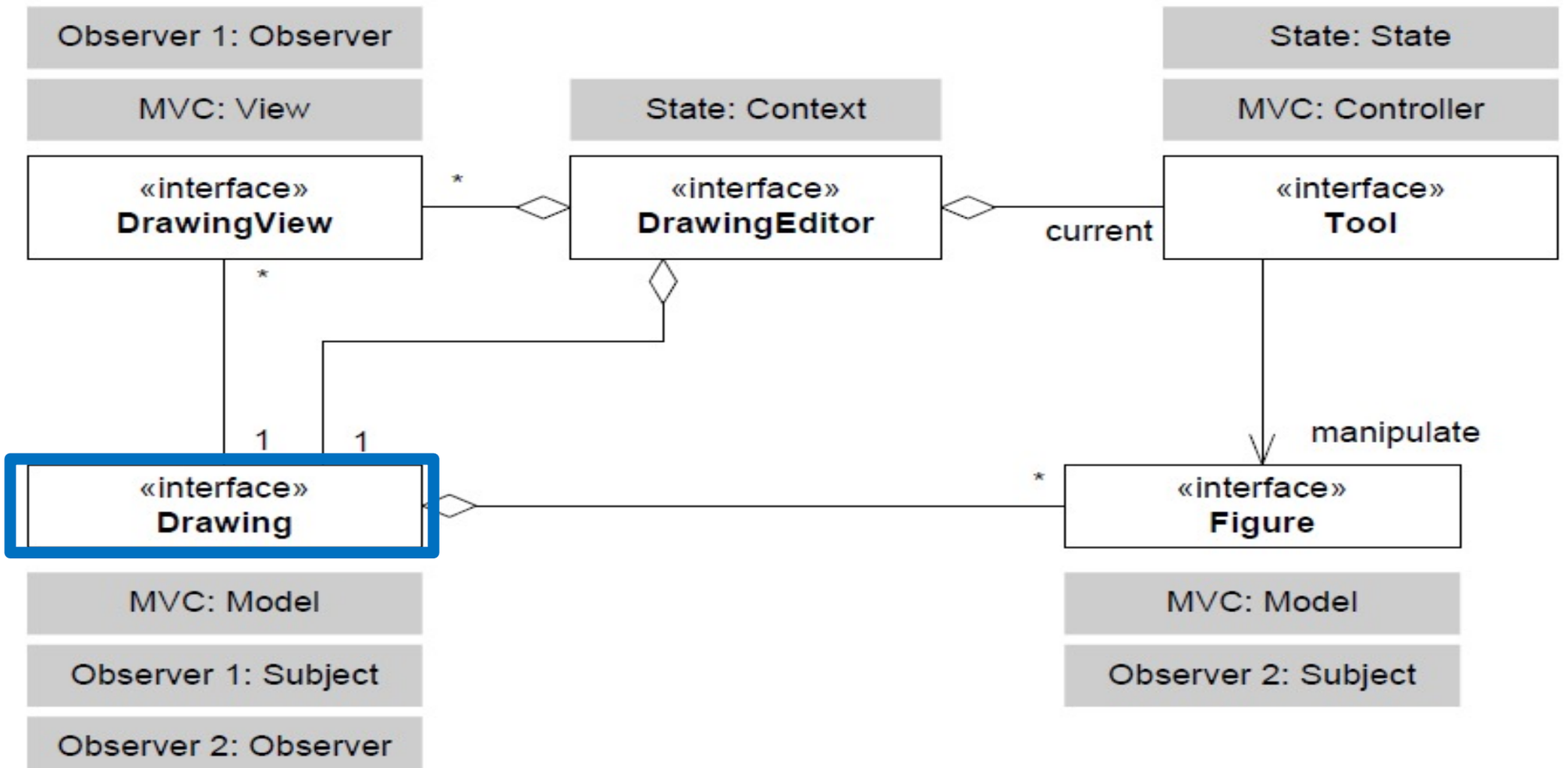
MiniDraw can be customized to develop applications

- Jigsaw puzzle
- Shape drawing
- HotCiv

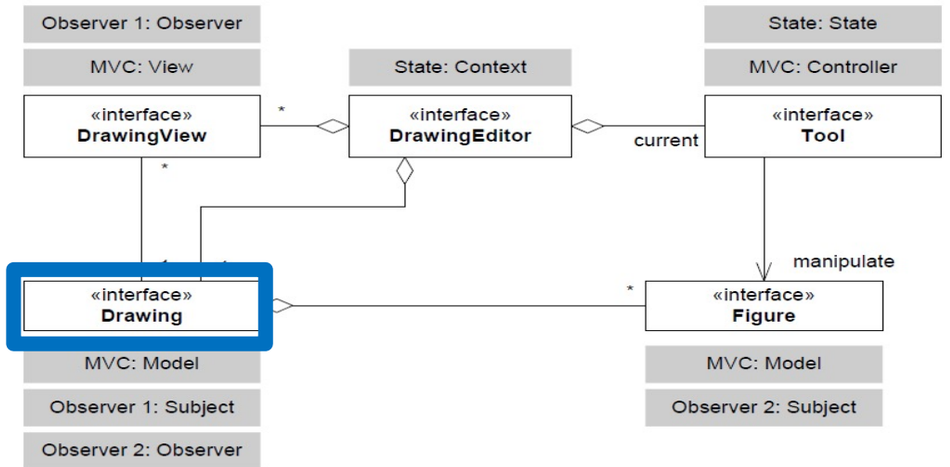
MiniDraw does not “know” anything about the application, but it knows about images and defines behavior to draw and manipulate them.

→ Customization

MiniDraw: Design

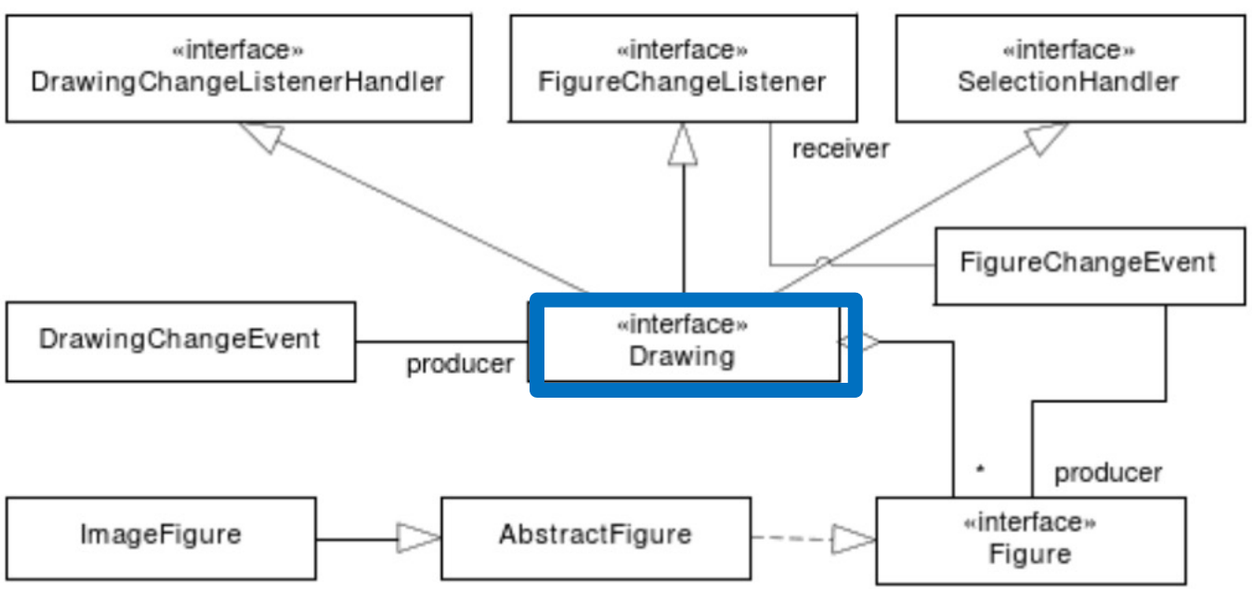


MiniDraw: Design

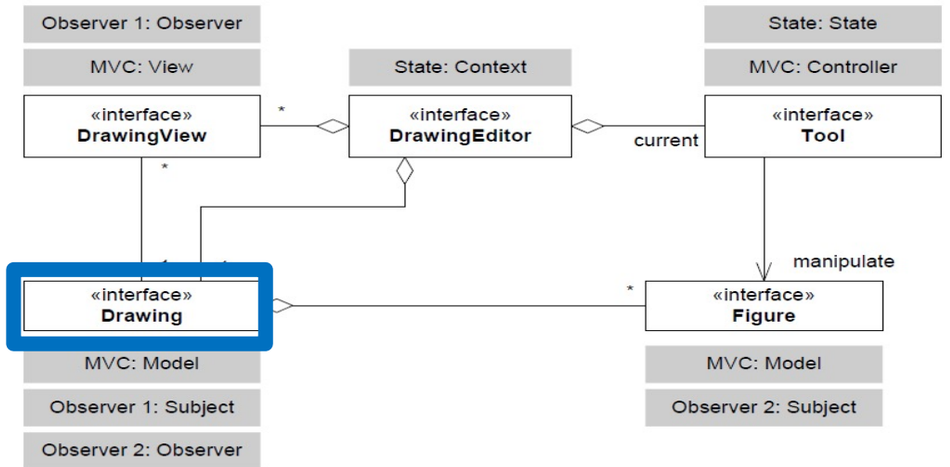


Drawing (MVC: Model)

- Be a collection of figures
- Allow figures to be added and removed
- Maintain a temporary subset of figures (selection)
- Broadcast DrawingChangeEvents to all registered DrawingChangeListeners



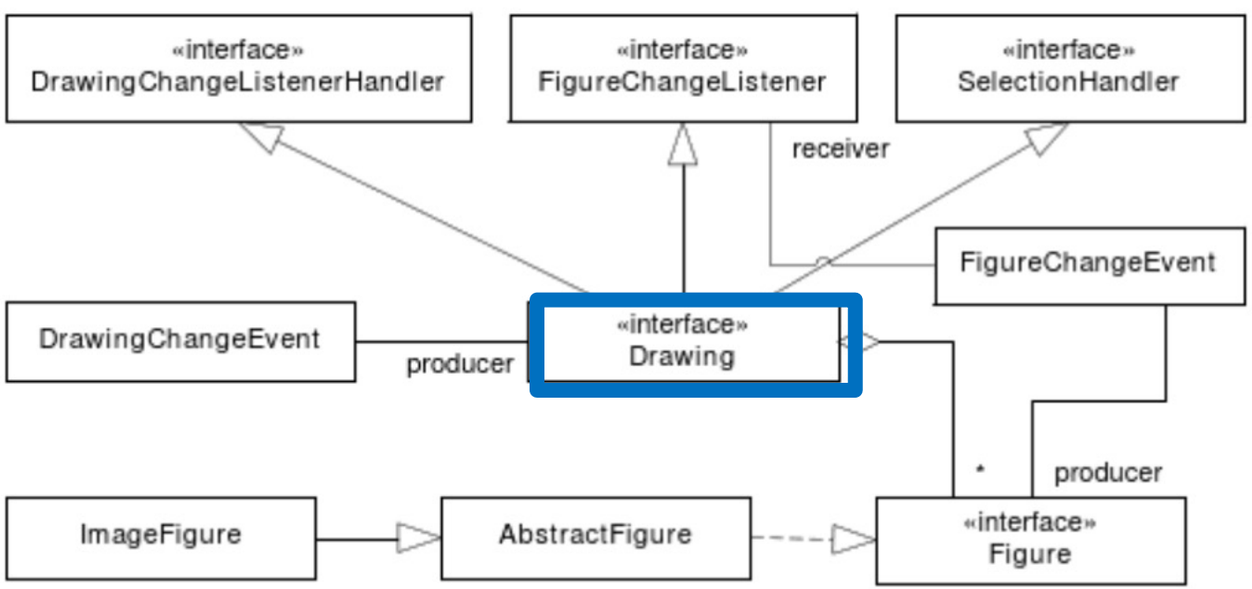
MiniDraw: Design



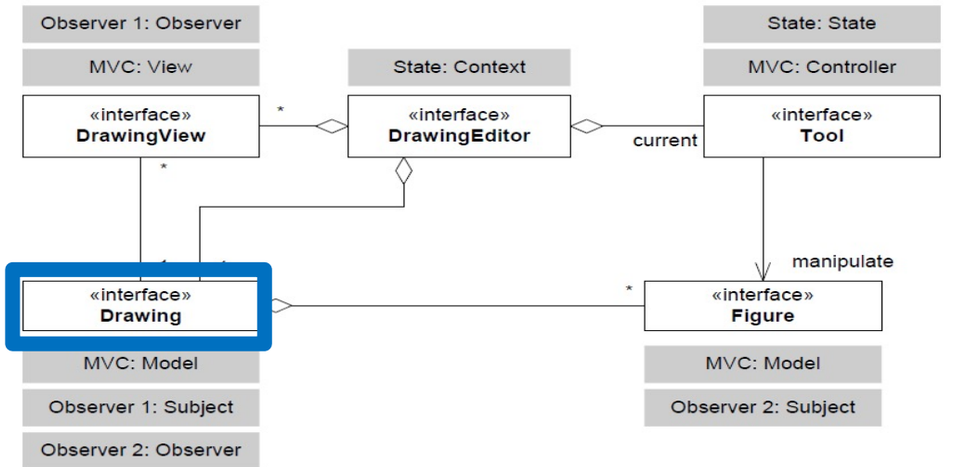
Drawing (MVC: Model)

- Be a collection of figures
- Allow figures to be added and removed
- Maintain a temporary subset of figures (selection)
- Broadcast DrawingChangeEvents to all registered DrawingChangeListeners

StandardDrawing



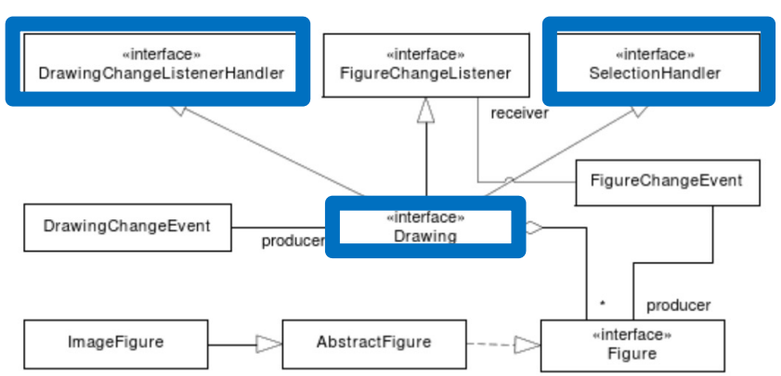
MiniDraw: Design



Drawing (MVC: Model)

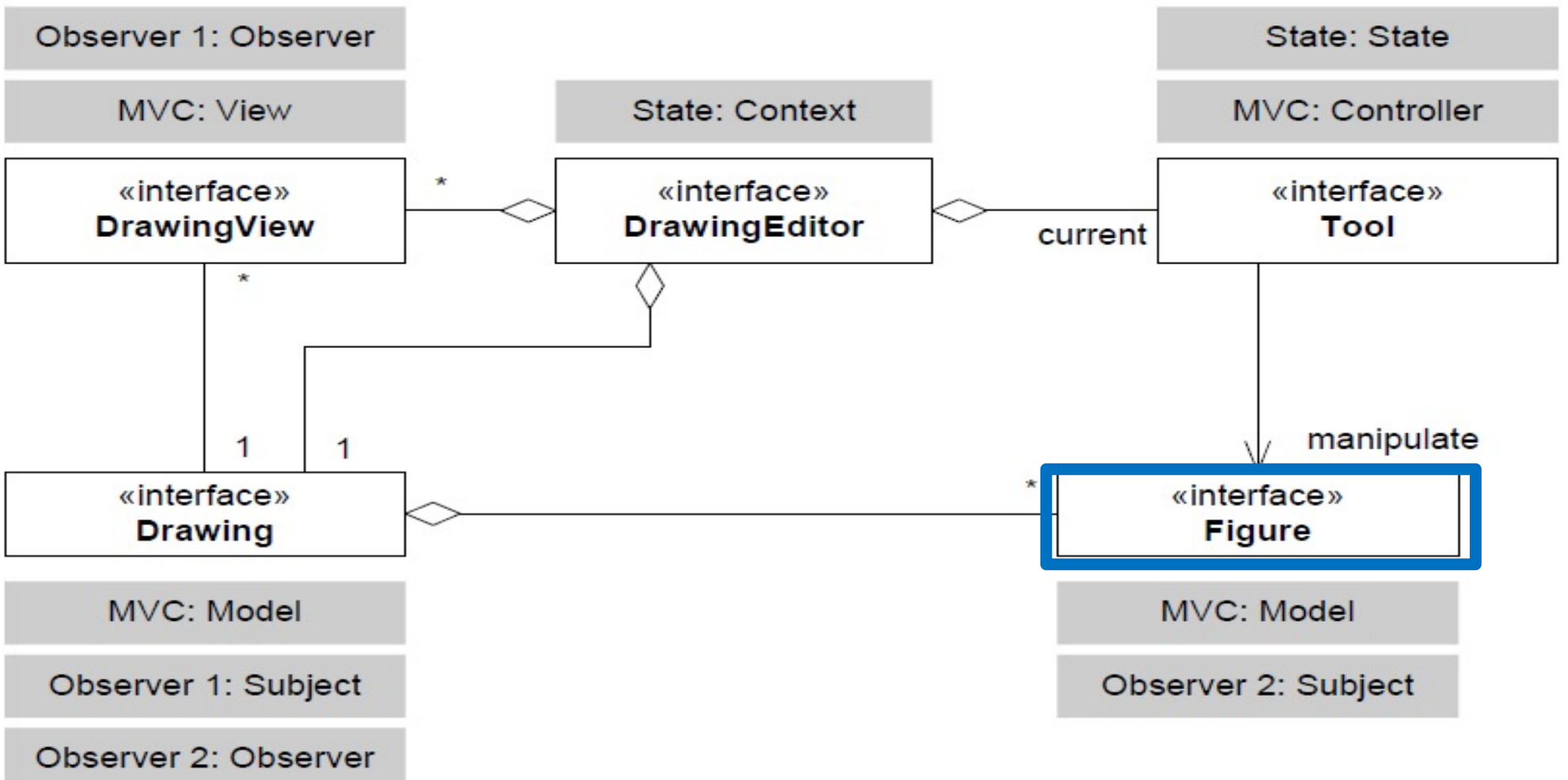
- Be a collection of figures
- Allow figures to be added and removed
- Maintain a temporary subset of figures (selection)
- Broadcast DrawingChangeEvents to all registered DrawingChangeListeners

StandardDrawing




```
/**
 * Adds a listener for this drawing.
 */
public void addDrawingChangeListener(DrawingChangeListener
                                      listener) {
  listenerHandler.addDrawingChangeListener(listener);
}
[...]
/**
 * Get a list of all selected figures
 */
public List<Figure> selection() {
  return selectionHandler.selection();
}
```

MiniDraw: Design



MiniDraw: Design

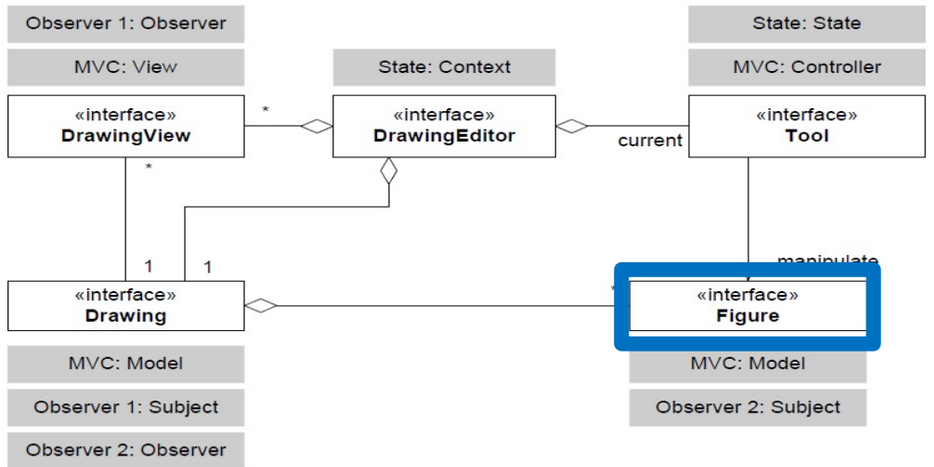
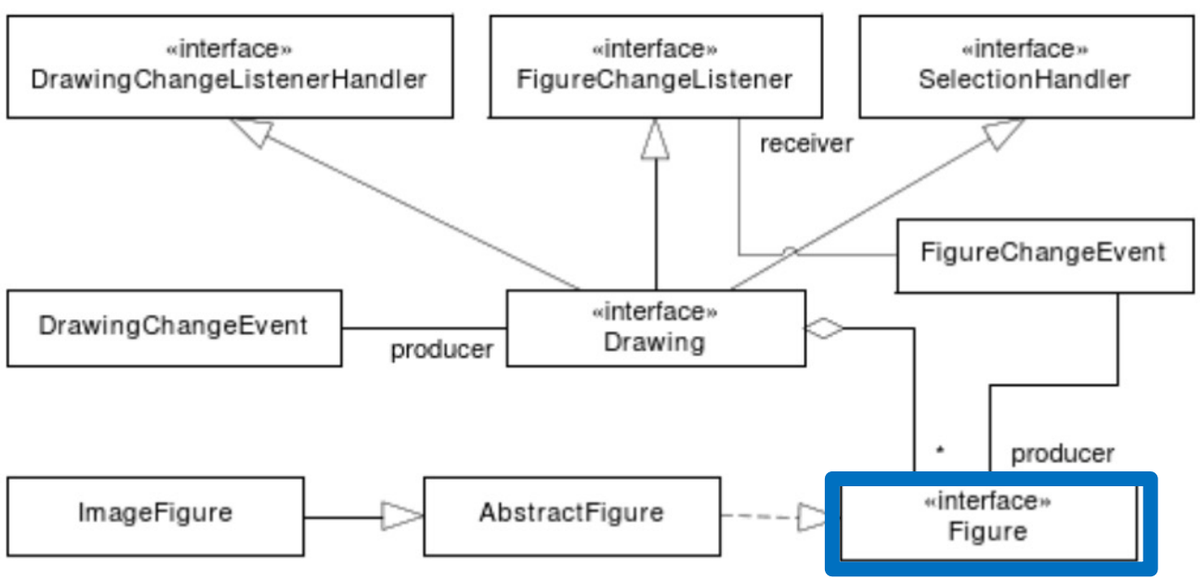


Figure (MVC: Model)

- Know how to draw itself
- Know its display box
- Can be moved
- Broadcast FigureChangeEvents to all registered FigureChangeListeners



MiniDraw: Design

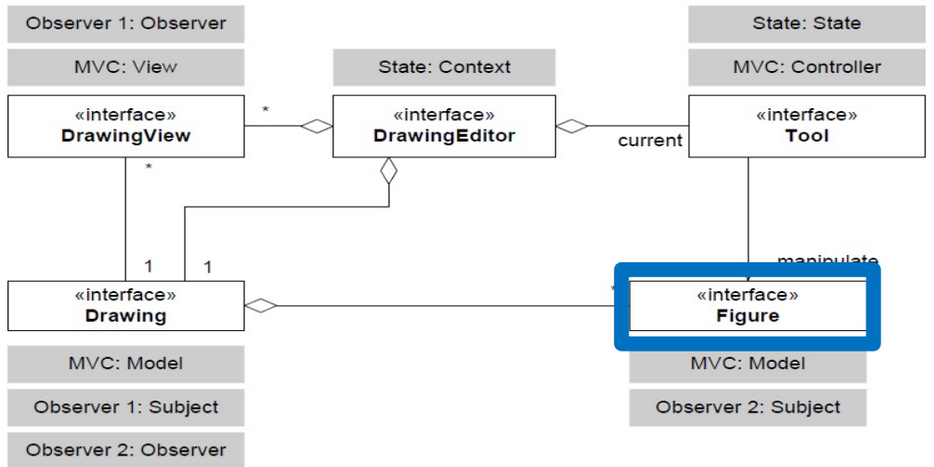
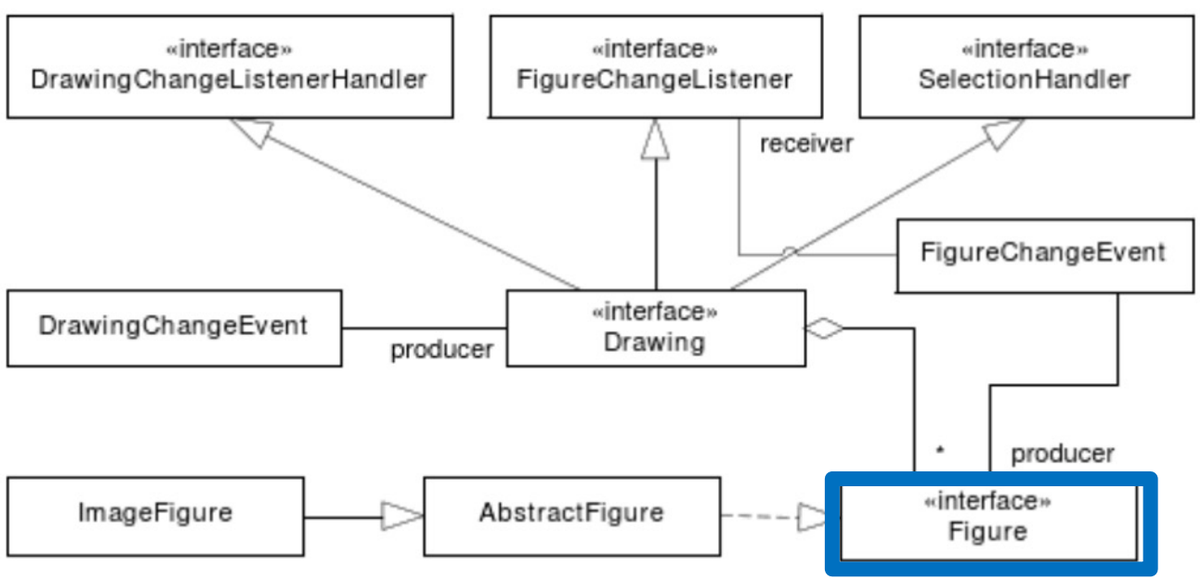


Figure (MVC: Model)

- Know how to draw itself
- Know its display box
- Can be moved
- Broadcast FigureChangeEvents to all registered FigureChangeListeners

ImageFigure



MiniDraw: Design

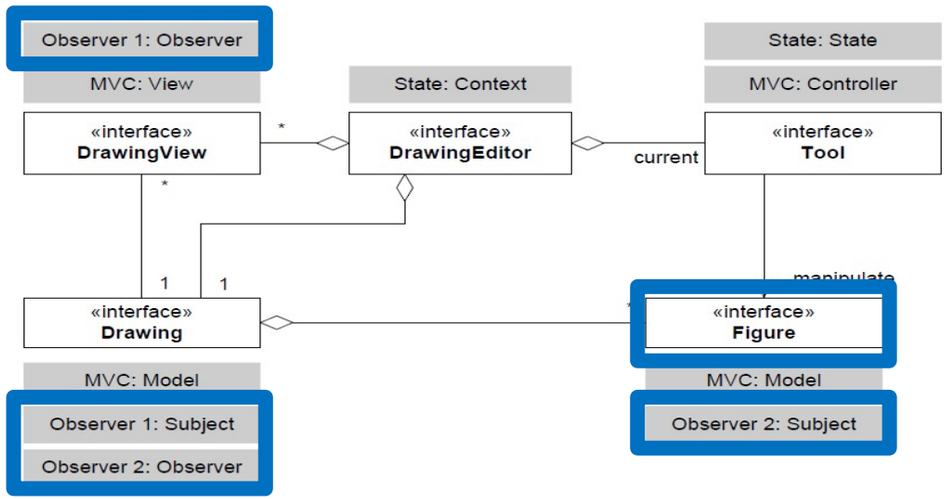
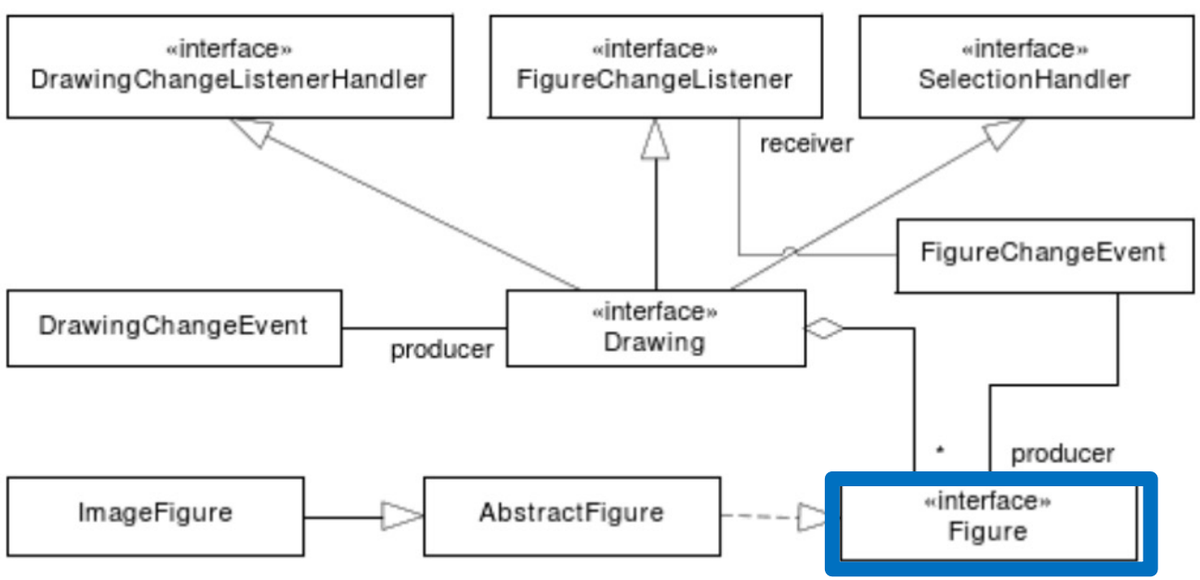


Figure (MVC: Model)

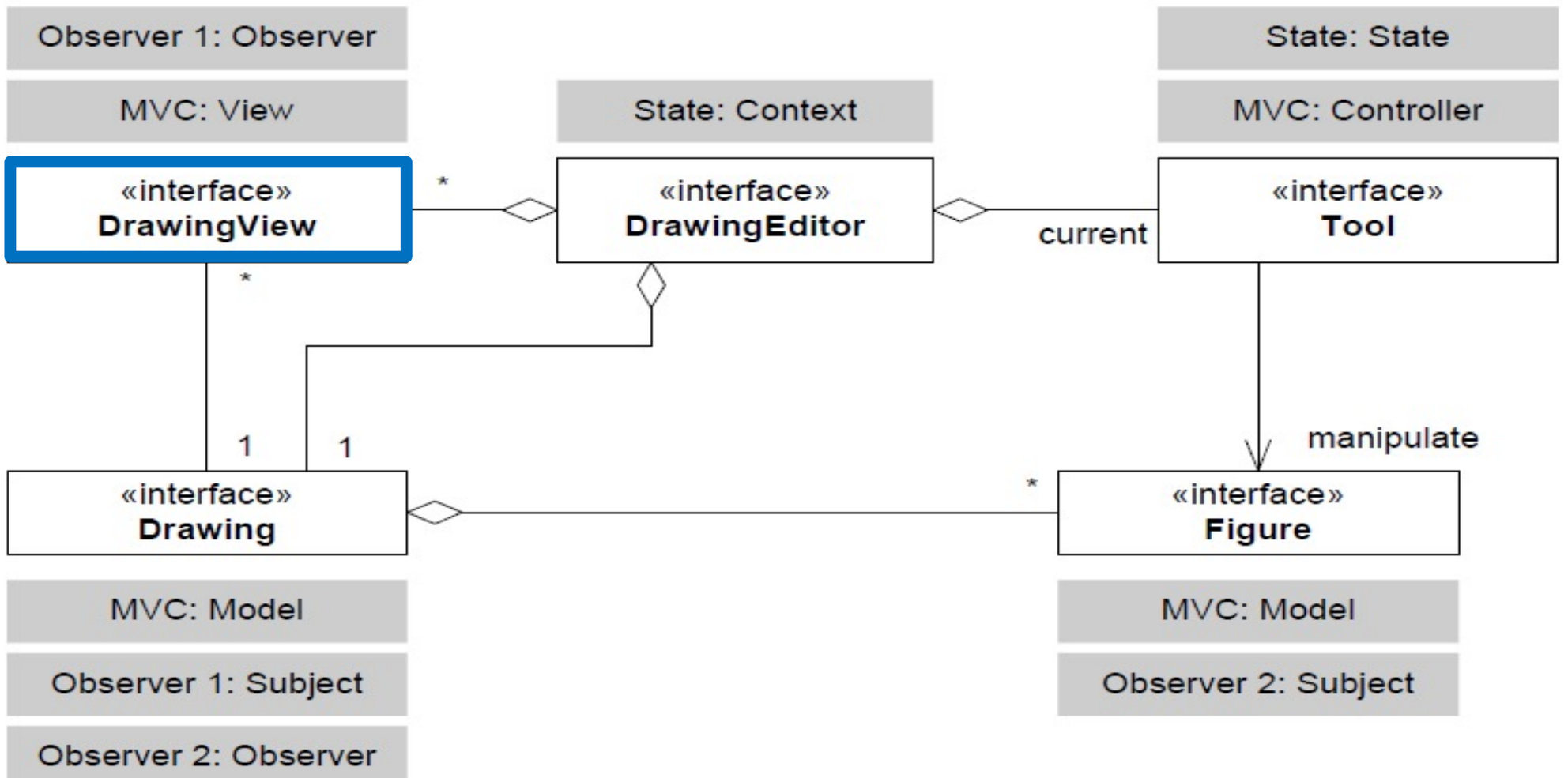
- Know how to draw itself
- Know its display box
- Can be moved
- Broadcast FigureChangeEvents to all registered FigureChangeListeners

ImageFigure

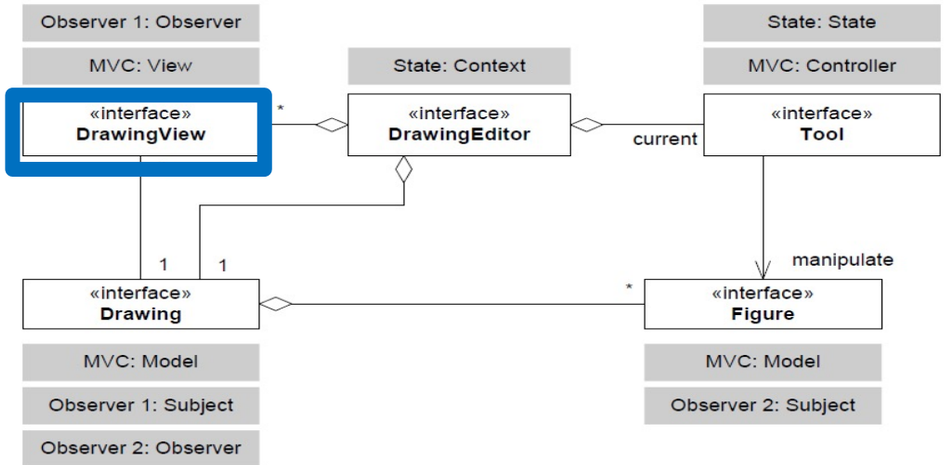
Drawing is both Subject
(observed by DrawingView)
and Observer
(of subject Figure)



MiniDraw: Design

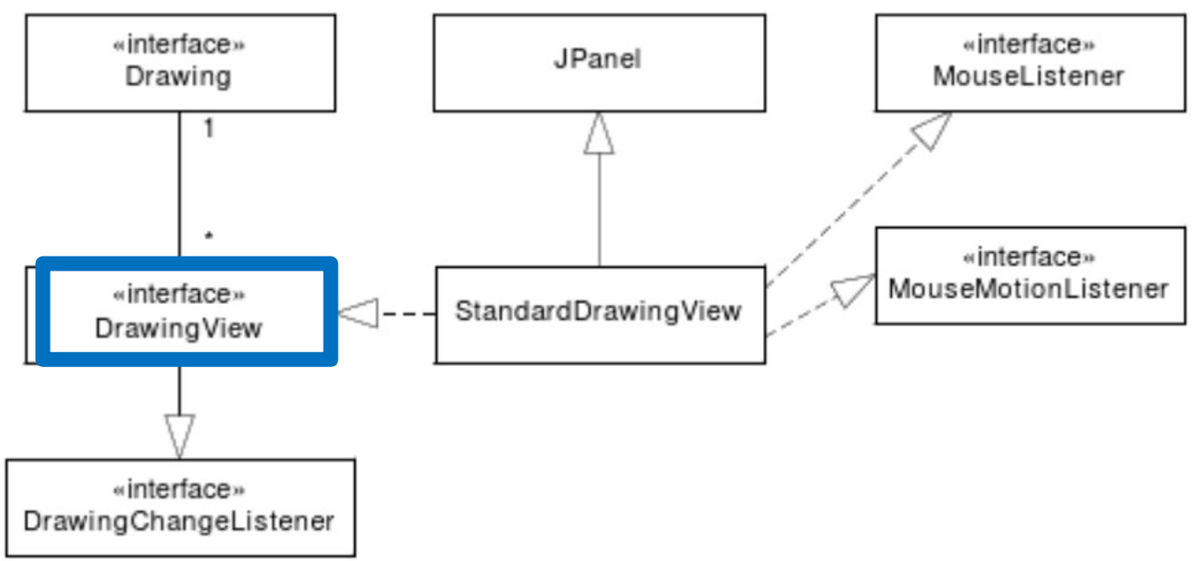


MiniDraw: Design

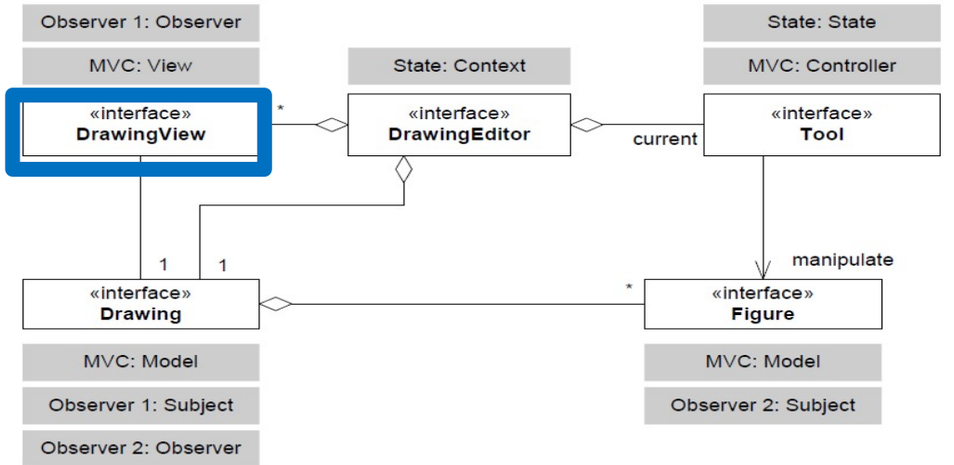


DrawingView (MVC: View)

- Define four layers of graphics drawn in order: background, drawing contents, selection highlights, overlay
- Respond to change events from associated Drawing and ensure redrawing
- Forward all mouse and key events to the editor's associated tool



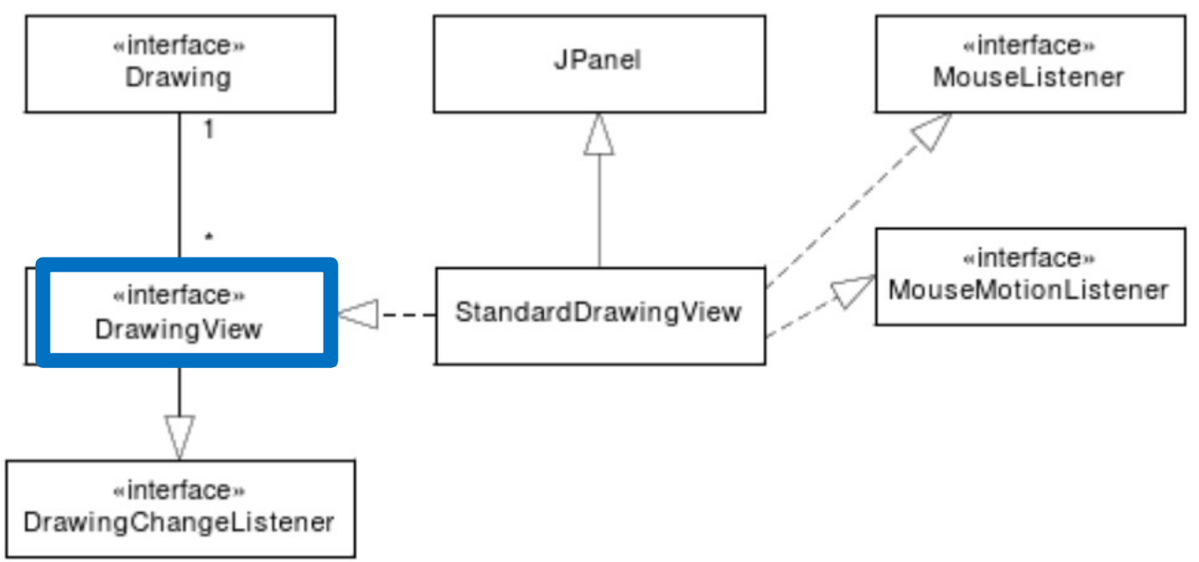
MiniDraw: Design



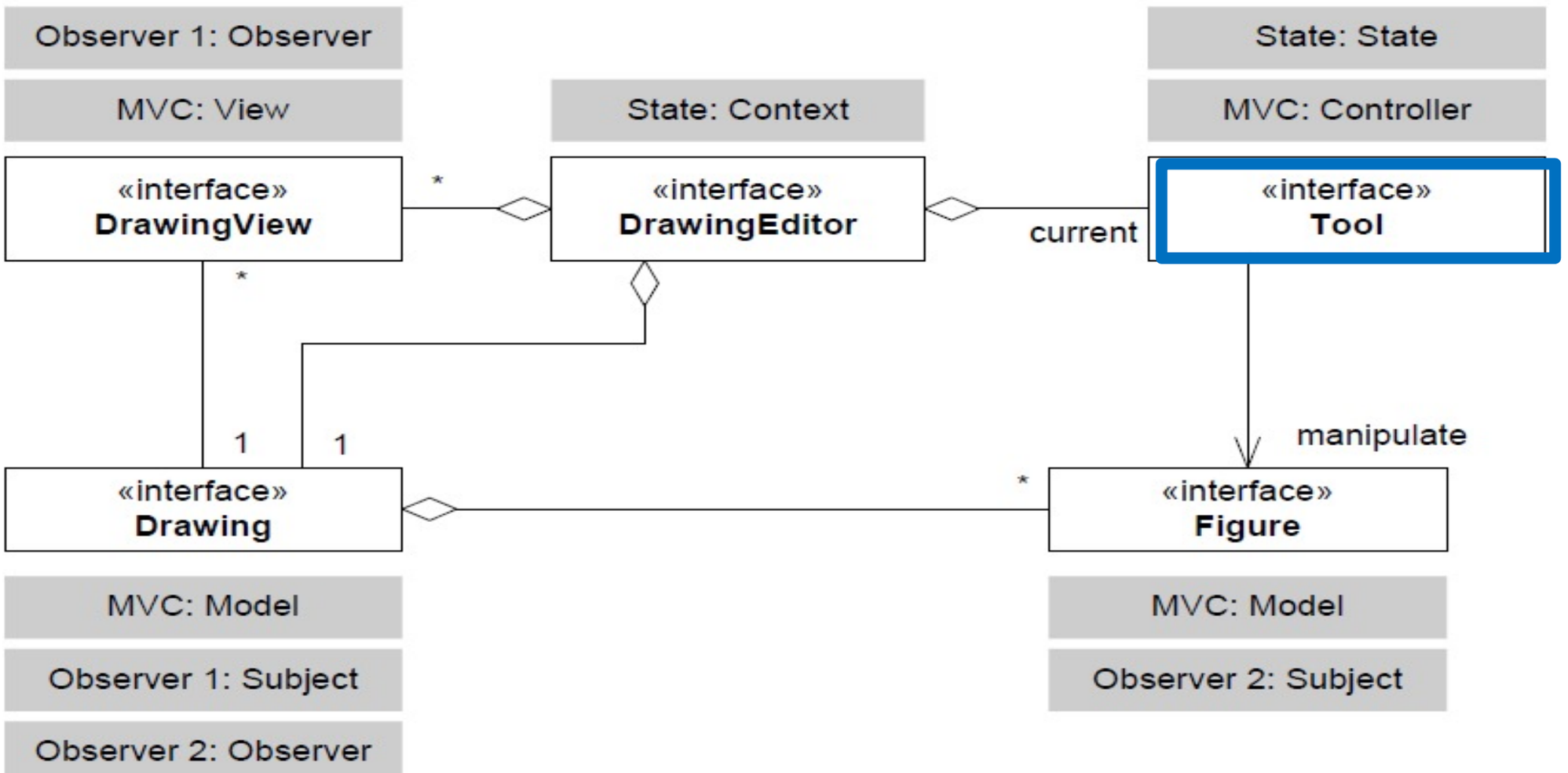
DrawingView (MVC: View)

- Define four layers of graphics drawn in order: background, drawing contents, selection highlights, overlay
- Respond to change events from associated Drawing and ensure redrawing
- Forward all mouse and key events to the editor's associated tool

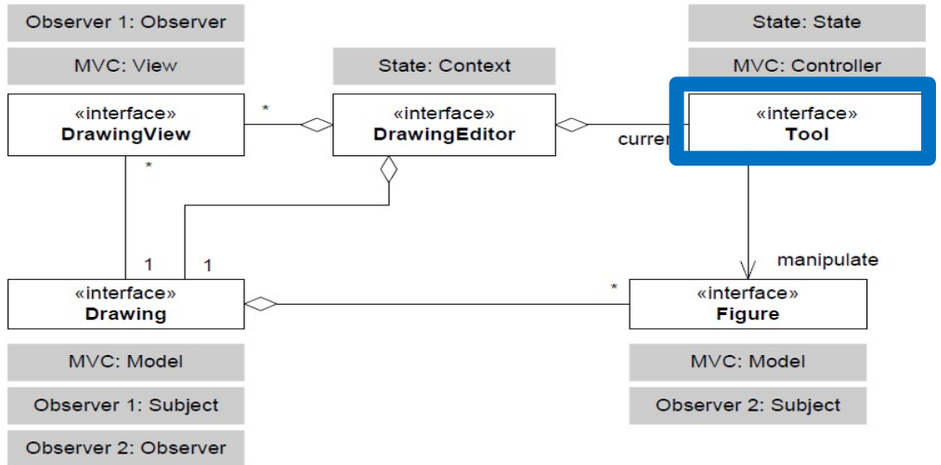
StandardDrawingView
StdViewWithBackground



MiniDraw: Design

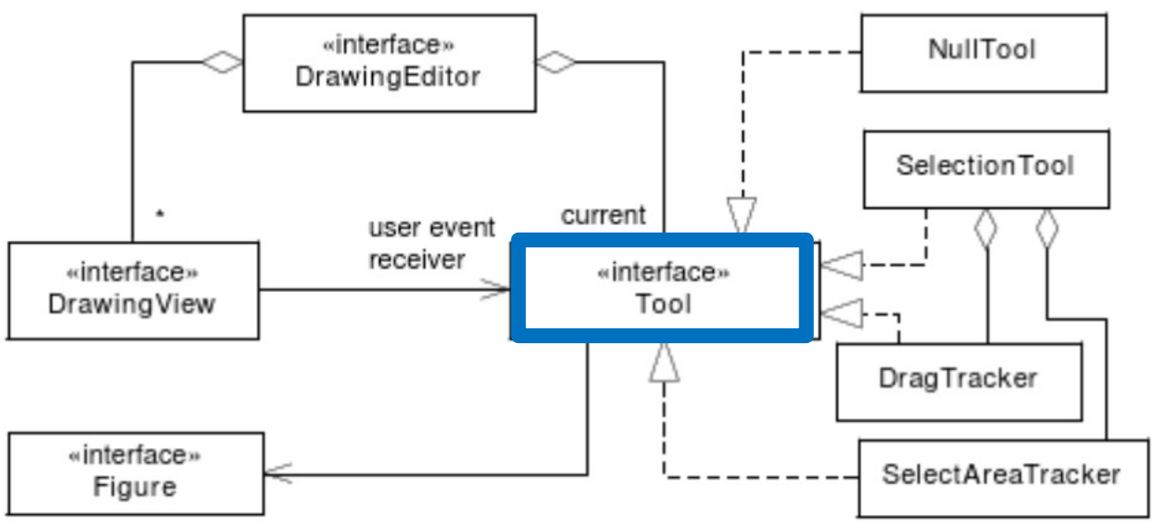


MiniDraw: Design

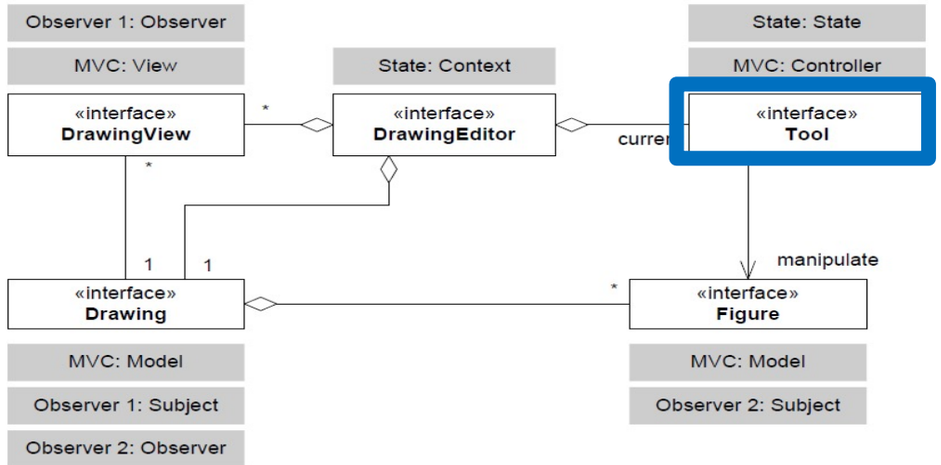


Tool (MVC: Controller)

- Receive mouse events and key events
- Define manipulation of the contents of Drawing or other changes relevant for the application



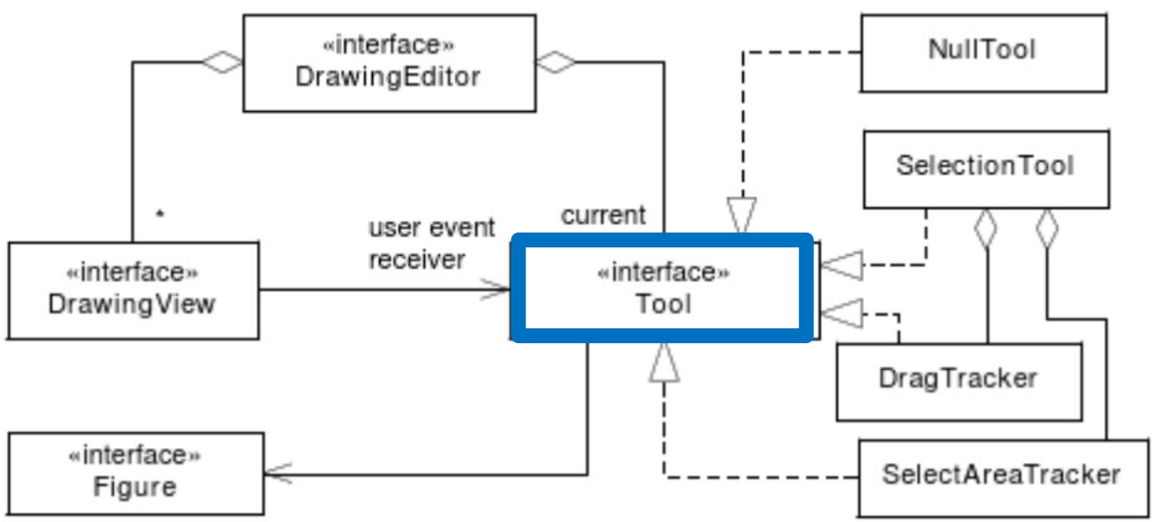
MiniDraw: Design



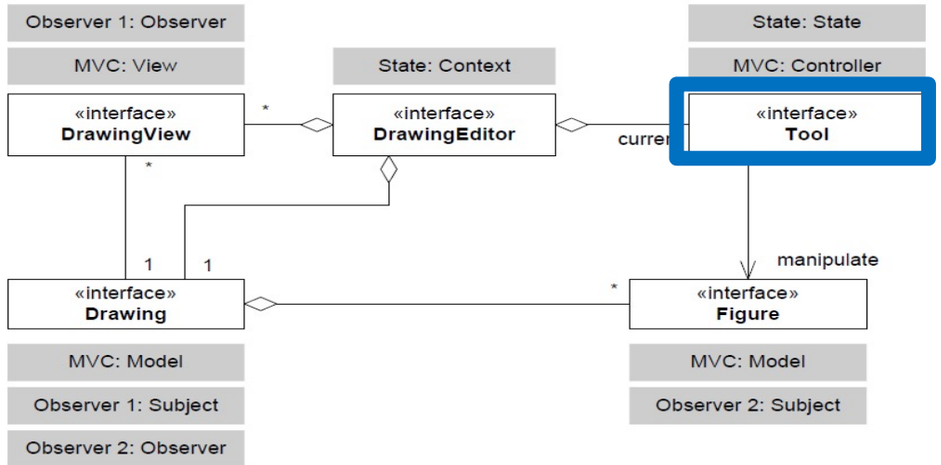
Tool (MVC: Controller)

- Receive mouse events and key events
- Define manipulation of the contents of Drawing or other changes relevant for the application

NullTool
SelectionTool
(DragTracker + SelectAreaTracker)



MiniDraw: Design

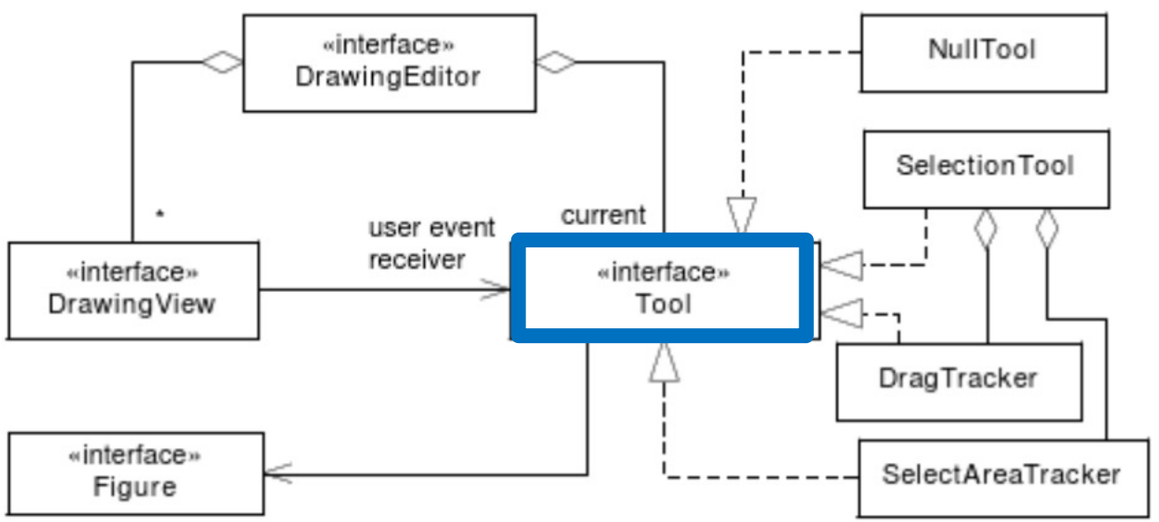


Tool (MVC: Controller)

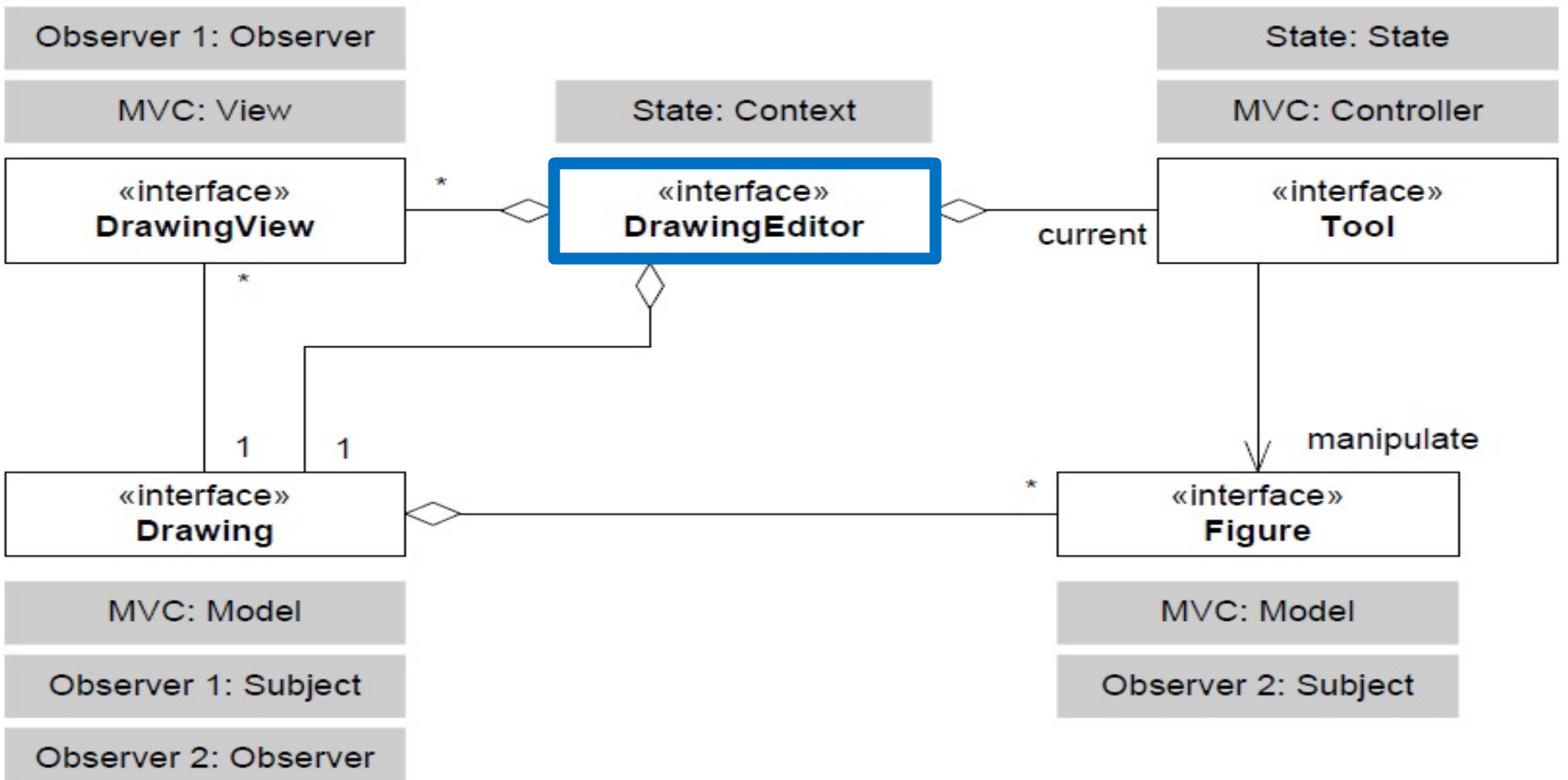
- Receive mouse events and key events
- Define manipulation of the contents of Drawing or other changes relevant for the application

NullTool
SelectionTool
(DragTracker + SelectAreaTracker)

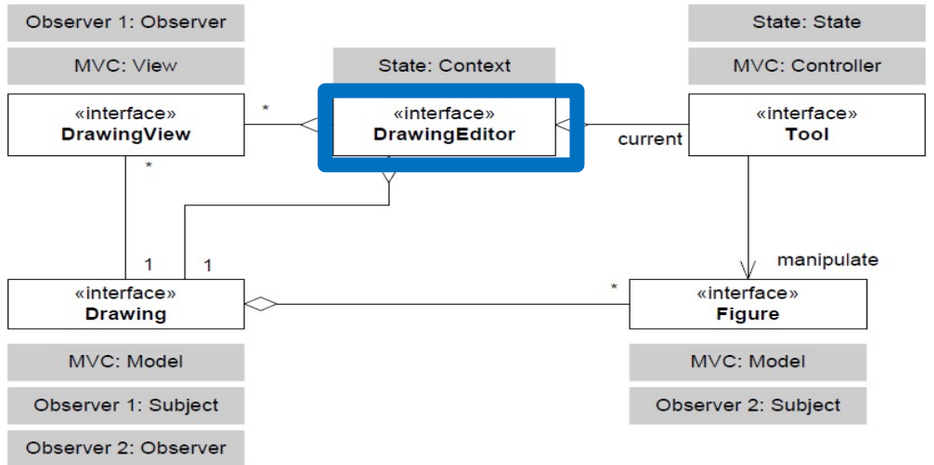
“State” in state pattern



MiniDraw: Design

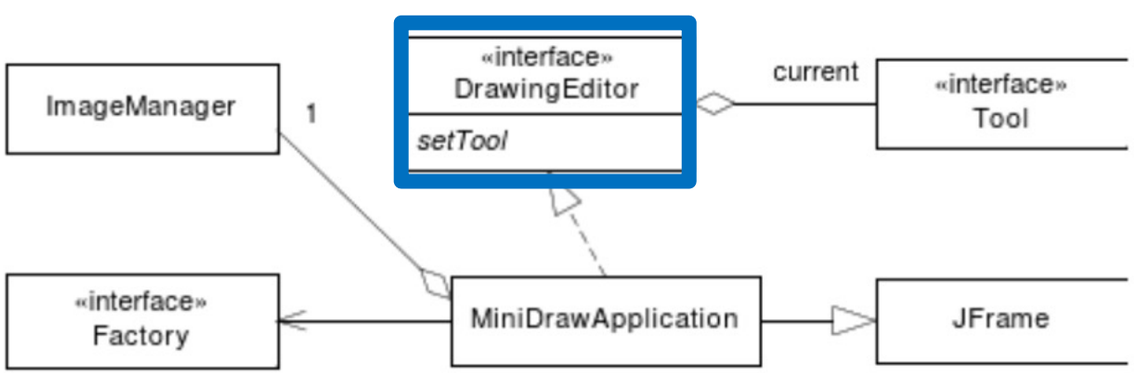


MiniDraw: Design

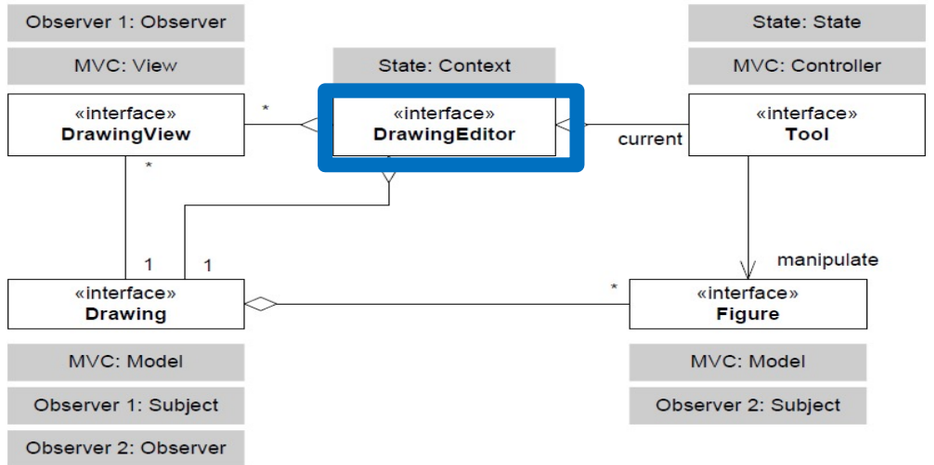


DrawingEditor

- Main class, instantiate all parts of the application
- Open a window to make a visible application
- Central access point for various parts of MiniDraw
- Allow changing the active tool
- Allow displaying messages in the status bar



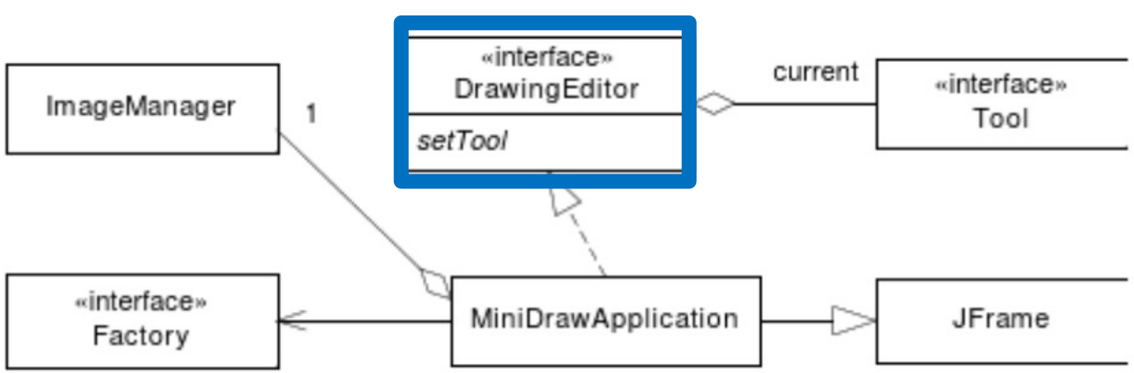
MiniDraw: Design



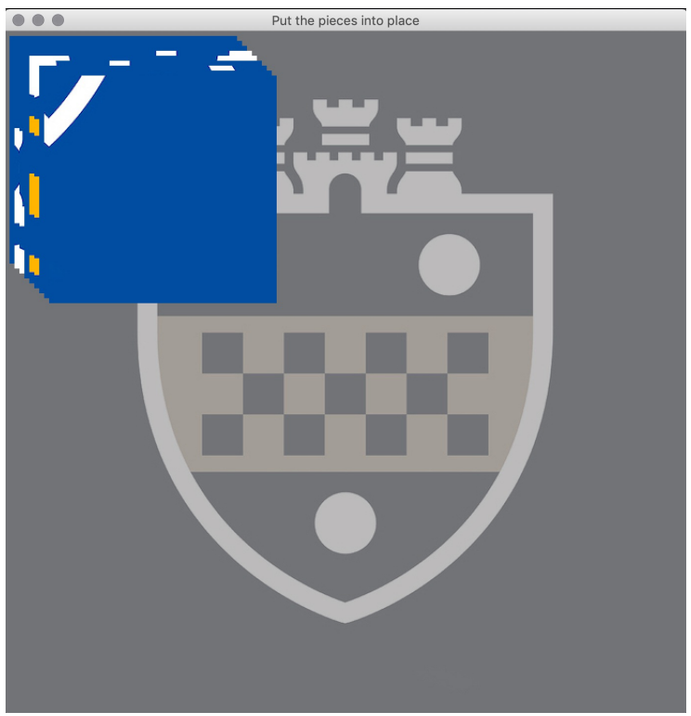
DrawingEditor

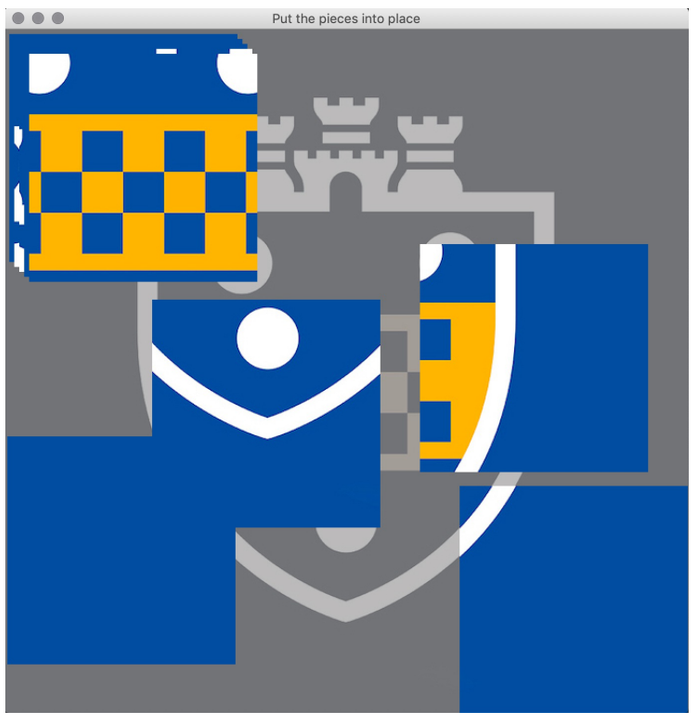
- Main class, instantiate all parts of the application
- Open a window to make a visible application
- Central access point for various parts of MiniDraw
- Allow changing the active tool
- Allow displaying messages in the status bar

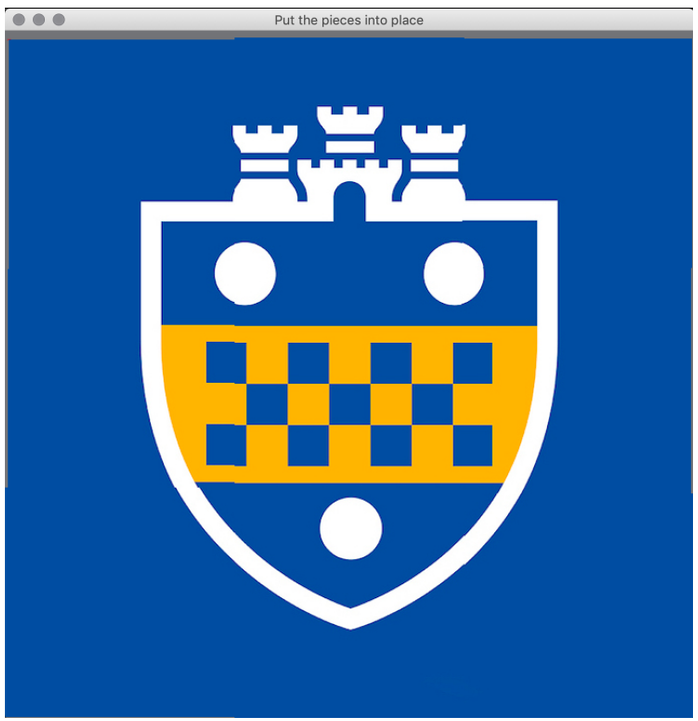
MiniDrawApplication



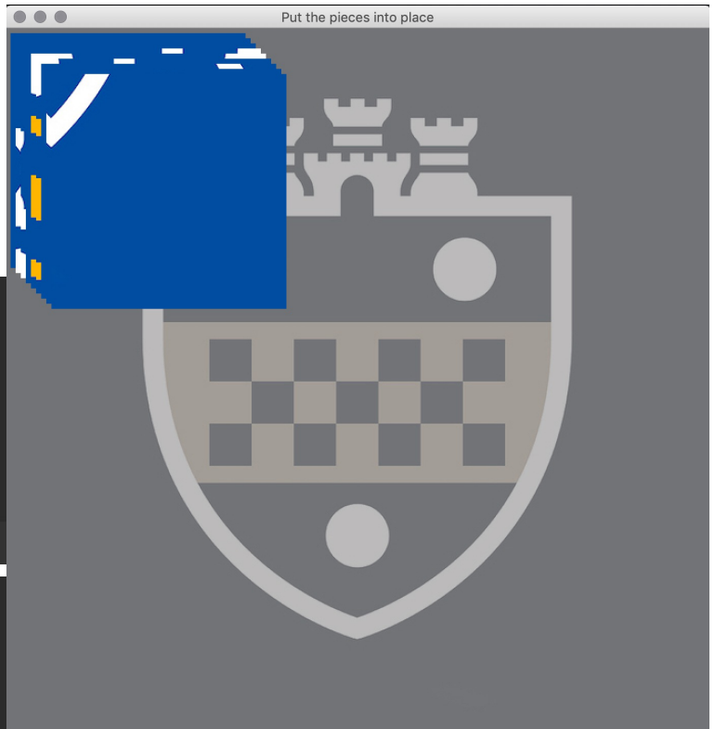
MiniDraw: Puzzle







MiniDraw: Puzzle



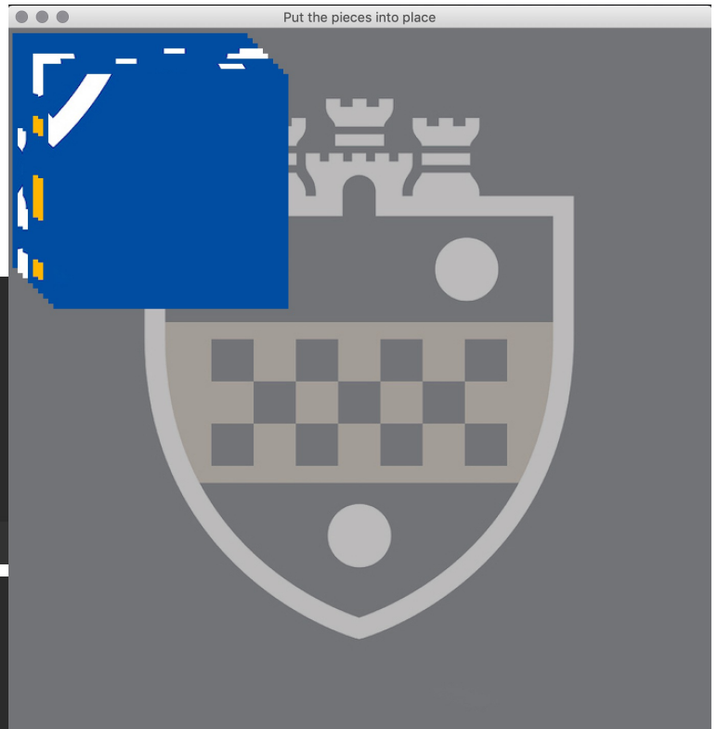

```
1   package puzzle;
2   import minidraw.standard.*;
3   import minidraw.framework.*;
4   import java.awt.*;
5   import javax.swing.*;
6
```

```
34  public class LogoPuzzle {
35
36    public static void main(String[] args) {
37      DrawingEditor editor =
38        new MiniDrawApplication( title: "Put the pieces into place",
39                                 new PuzzleFactory() );
40      editor.open();
41      editor.setTool( new SelectionTool(editor) );
42
43      Drawing drawing = editor.drawing();
44      drawing.add(  new ImageFigure( imagename: "11", new Point( x: 5,  y: 5)) );
45      drawing.add(  new ImageFigure( imagename: "12", new Point( x: 10,  y: 10)) );
46      drawing.add(  new ImageFigure( imagename: "13", new Point( x: 15,  y: 15)) );
47      drawing.add(  new ImageFigure( imagename: "21", new Point( x: 20,  y: 20)) );
48      drawing.add(  new ImageFigure( imagename: "22", new Point( x: 25,  y: 25)) );
49      drawing.add(  new ImageFigure( imagename: "23", new Point( x: 30,  y: 30)) );
50      drawing.add(  new ImageFigure( imagename: "31", new Point( x: 35,  y: 35)) );
51      drawing.add(  new ImageFigure( imagename: "32", new Point( x: 40,  y: 40)) );
52      drawing.add(  new ImageFigure( imagename: "33", new Point( x: 45,  y: 45)) );
53    }
54  }
```

MiniDraw: Puzzle

```
1   package puzzle;
2   import minidraw.standard.*;
3   import minidraw.framework.*;
4   import java.awt.*;
5   import javax.swing.*;
6
```

```
34  public class LogoPuzzle {
35
36    public static void main(String[] args) {
37      DrawingEditor editor =
38        new MiniDrawApplication( title: "Put the pieces into place",
39                                 new PuzzleFactory() );
40      editor.open();
41      editor.setTool( new SelectionTool(editor) );
42
43      Drawing drawing = editor.drawing();
44      drawing.add(  new ImageFigure( imagename: "11", new Point( x: 5,  y: 5)) );
45      drawing.add(  new ImageFigure( imagename: "12", new Point( x: 10,  y: 10)) );
46      drawing.add(  new ImageFigure( imagename: "13", new Point( x: 15,  y: 15)) );
47      drawing.add(  new ImageFigure( imagename: "21", new Point( x: 20,  y: 20)) );
48      drawing.add(  new ImageFigure( imagename: "22", new Point( x: 25,  y: 25)) );
49      drawing.add(  new ImageFigure( imagename: "23", new Point( x: 30,  y: 30)) );
50      drawing.add(  new ImageFigure( imagename: "31", new Point( x: 35,  y: 35)) );
51      drawing.add(  new ImageFigure( imagename: "32", new Point( x: 40,  y: 40)) );
52      drawing.add(  new ImageFigure( imagename: "33", new Point( x: 45,  y: 45)) );
53    }
54  }
```



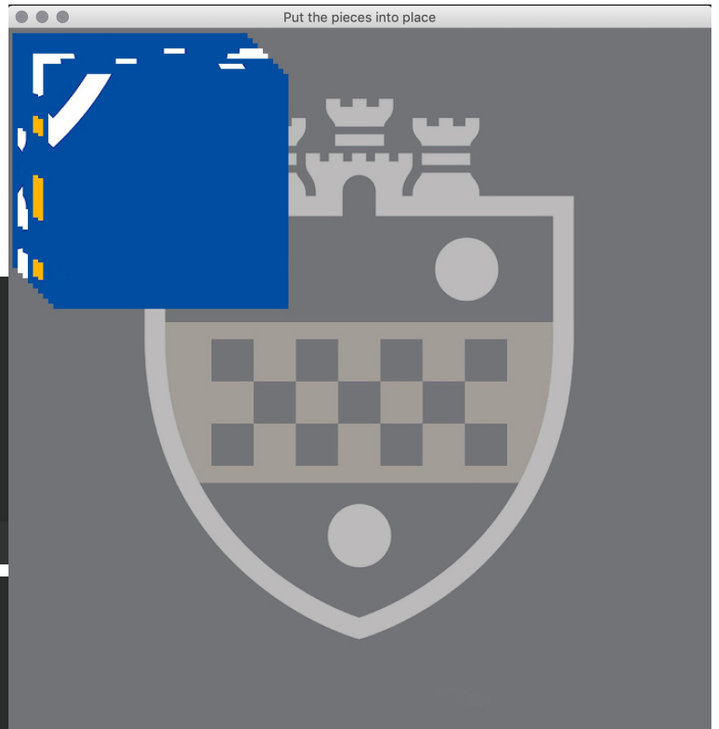
DrawingEditor

MiniDrawApplication: Window name, Abstract Factory

MiniDraw: Puzzle

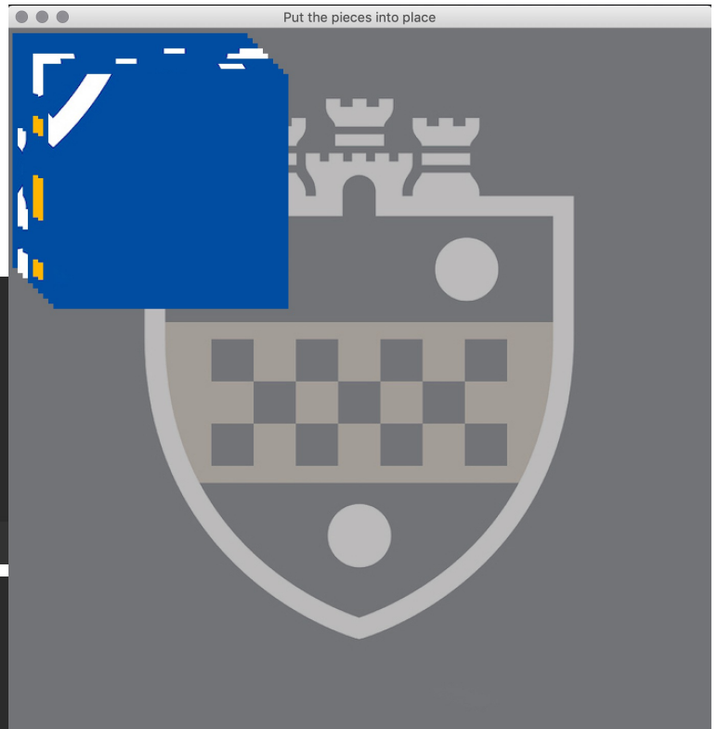
```
1   package puzzle;
2   import minidraw.standard.*;
3   import minidraw.framework.*;
4   import java.awt.*;
5   import javax.swing.*;
6
```

```
34  public class LogoPuzzle {
35
36    public static void main(String[] args) {
37      DrawingEditor editor =
38        new MiniDrawApplication( title: "Put the pieces into place",
39                                 new PuzzleFactory() );
40      editor.open();
41      editor.setTool( new SelectionTool(editor) );
42
43      Drawing drawing = editor.drawing();
44      drawing.add(  new ImageFigure( imagename: "11", new Point( x: 5,  y: 5)) );
45      drawing.add(  new ImageFigure( imagename: "12", new Point( x: 10,  y: 10)) );
46      drawing.add(  new ImageFigure( imagename: "13", new Point( x: 15,  y: 15)) );
47      drawing.add(  new ImageFigure( imagename: "21", new Point( x: 20,  y: 20)) );
48      drawing.add(  new ImageFigure( imagename: "22", new Point( x: 25,  y: 25)) );
49      drawing.add(  new ImageFigure( imagename: "23", new Point( x: 30,  y: 30)) );
50      drawing.add(  new ImageFigure( imagename: "31", new Point( x: 35,  y: 35)) );
51      drawing.add(  new ImageFigure( imagename: "32", new Point( x: 40,  y: 40)) );
52      drawing.add(  new ImageFigure( imagename: "33", new Point( x: 45,  y: 45)) );
53    }
54  }
```



Figures:
MiniDraw graphical elements
Add to Drawing (name, location)

MiniDraw: Puzzle



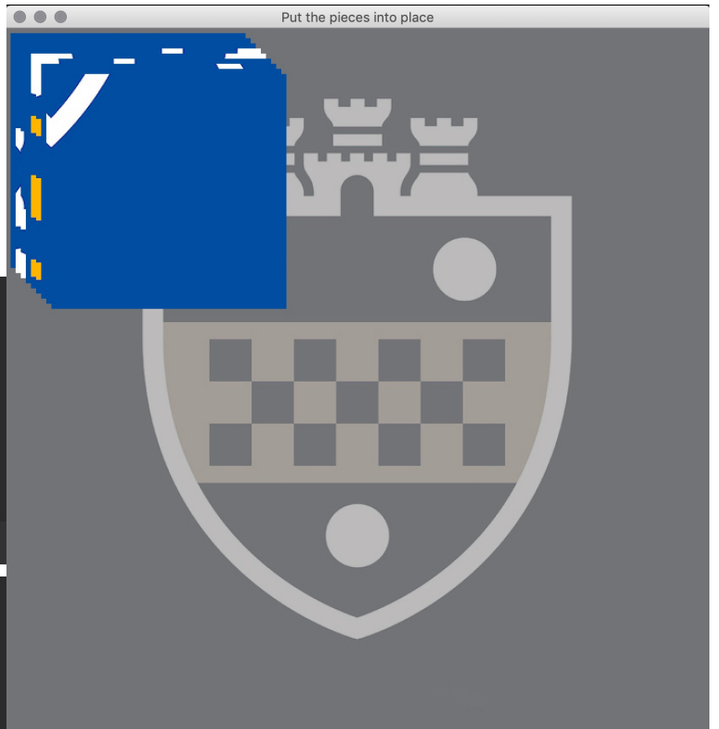

```
1   package puzzle;
2   import minidraw.standard.*;
3   import minidraw.framework.*;
4   import java.awt.*;
5   import javax.swing.*;
6
```

```
34  public class LogoPuzzle {
35
36    public static void main(String[] args) {
37      DrawingEditor editor =
38        new MiniDrawApplication( title: "Put the pieces into place",
39                                 new PuzzleFactory() );
40      editor.open();
41      editor.setTool( new SelectionTool(editor) );
42
43      Drawing drawing = editor.drawing();
44      drawing.add(  new ImageFigure( imagename: "11", new Point( x: 5,  y: 5)) );
45      drawing.add(  new ImageFigure( imagename: "12", new Point( x: 10,  y: 10)) );
46      drawing.add(  new ImageFigure( imagename: "13", new Point( x: 15,  y: 15)) );
47      drawing.add(  new ImageFigure( imagename: "21", new Point( x: 20,  y: 20)) );
48      drawing.add(  new ImageFigure( imagename: "22", new Point( x: 25,  y: 25)) );
49      drawing.add(  new ImageFigure( imagename: "23", new Point( x: 30,  y: 30)) );
50      drawing.add(  new ImageFigure( imagename: "31", new Point( x: 35,  y: 35)) );
51      drawing.add(  new ImageFigure( imagename: "32", new Point( x: 40,  y: 40)) );
52      drawing.add(  new ImageFigure( imagename: "33", new Point( x: 45,  y: 45)) );
53    }
54  }
```

Drawing:
Collection of figures

Once added, they show up in the window

MiniDraw: Puzzle




```
1   package puzzle;
2   import minidraw.standard.*;
3   import minidraw.framework.*;
4   import java.awt.*;
5   import javax.swing.*;
6
```

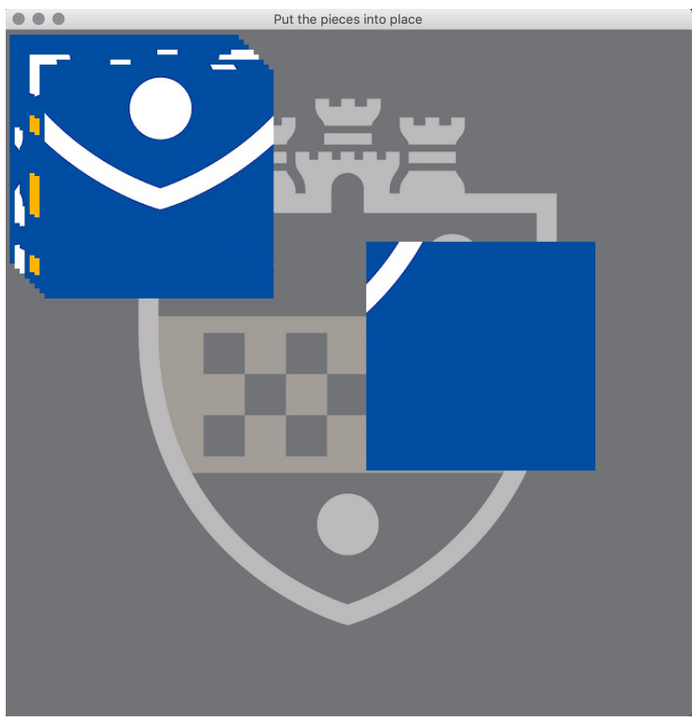
```
34  public class LogoPuzzle {
35
36    public static void main(String[] args) {
37      DrawingEditor editor =
38        new MiniDrawApplication( title: "Put the pieces into place",
39                                 new PuzzleFactory() );
40      editor.open();
41      editor.setTool( new SelectionTool(editor) );
42
43      Drawing drawing = editor.drawing();
44      drawing.add(  new ImageFigure( imagename: "11", new Point( x: 5,  y: 5)) );
45      drawing.add(  new ImageFigure( imagename: "12", new Point( x: 10,  y: 10)) );
46      drawing.add(  new ImageFigure( imagename: "13", new Point( x: 15,  y: 15)) );
47      drawing.add(  new ImageFigure( imagename: "21", new Point( x: 20,  y: 20)) );
48      drawing.add(  new ImageFigure( imagename: "22", new Point( x: 25,  y: 25)) );
49      drawing.add(  new ImageFigure( imagename: "23", new Point( x: 30,  y: 30)) );
50      drawing.add(  new ImageFigure( imagename: "31", new Point( x: 35,  y: 35)) );
51      drawing.add(  new ImageFigure( imagename: "32", new Point( x: 40,  y: 40)) );
52      drawing.add(  new ImageFigure( imagename: "33", new Point( x: 45,  y: 45)) );
53    }
54  }
```

Tool:
Manipulate the drawing area

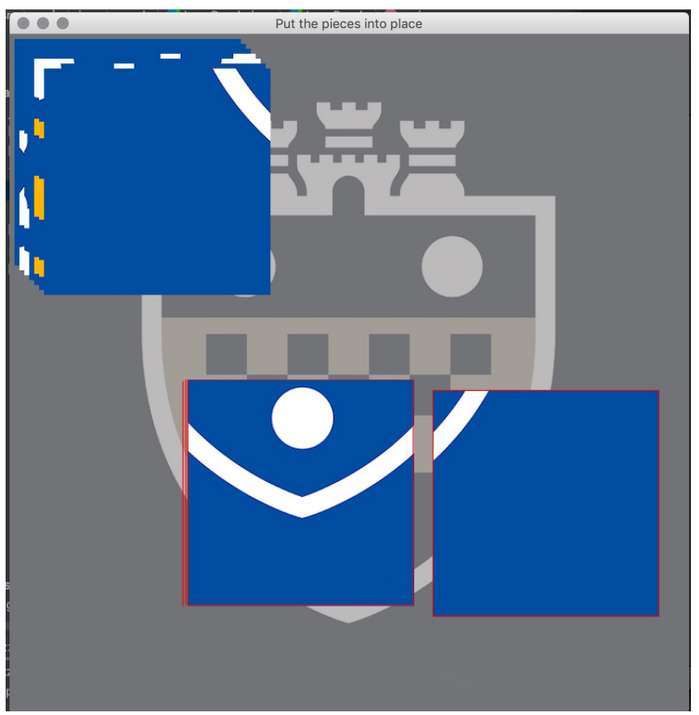
SelectionTool:
Move and selection behavior

MiniDraw: Puzzle

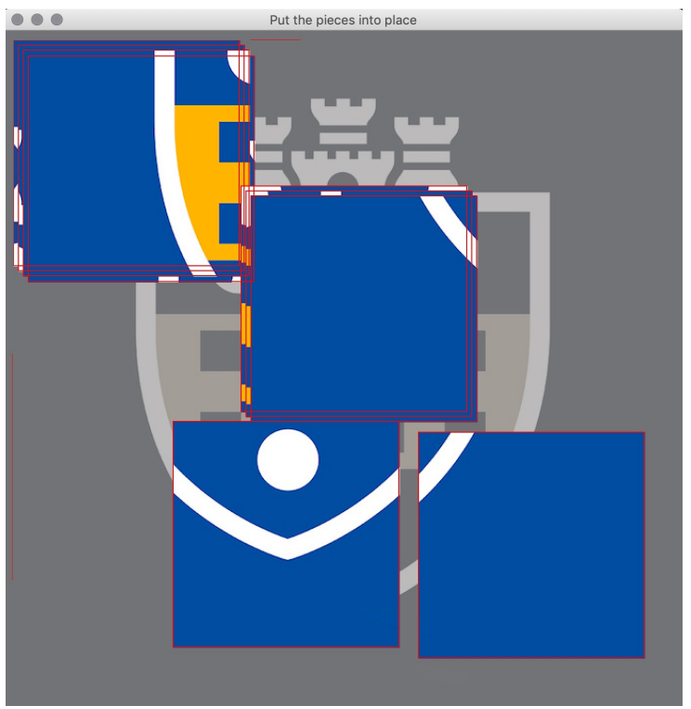
SelectionTool: Move and selection behavior



Click, hold, and drag



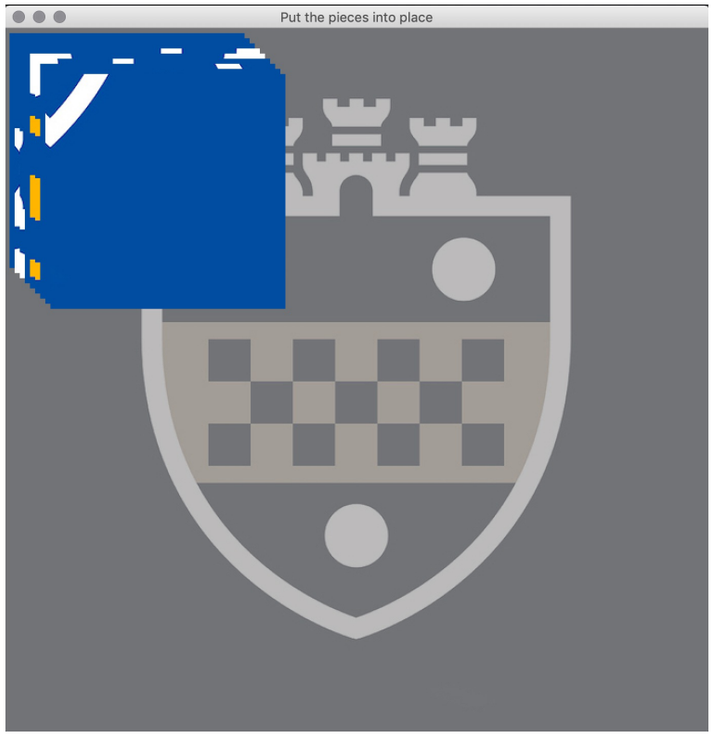
Shift-Click to select multiple



Click outside a figure and drag to “rubber-band” select

MiniDraw: Puzzle

```
56  class PuzzleFactory implements Factory {
57
58      public DrawingView createDrawingView( DrawingEditor editor ) {
59          DrawingView view =
60              new StdViewWithBackground(editor,  backgroundName: "pitt-shield-large");
61          return view;
62      }
63
64      public Drawing createDrawing( DrawingEditor editor ) { return new StandardDrawing(); }
67
68      public JTextField createStatusField( DrawingEditor editor ) { return null; }
71  }
72
```



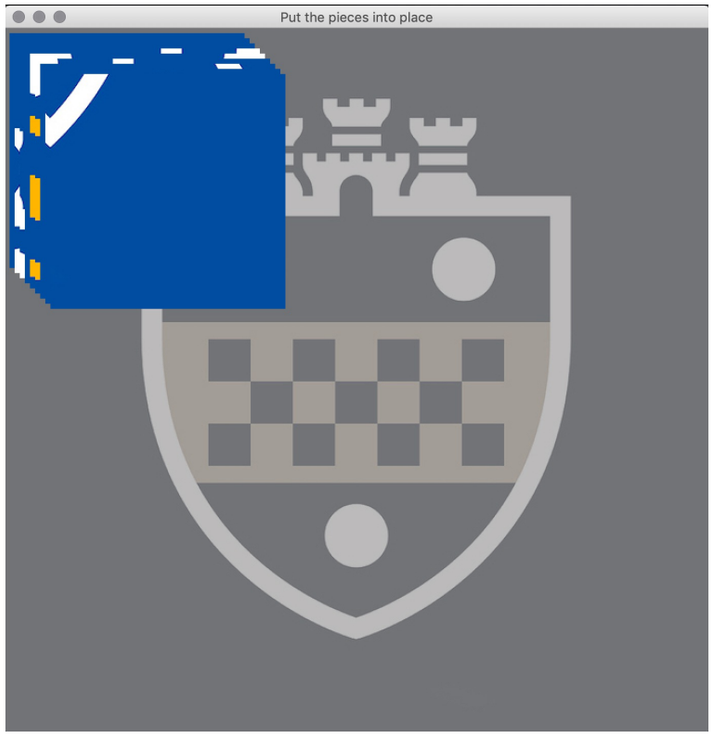
Factory:
Create concrete implementations of DrawingView and Drawing (and optional status text field)

DrawingView: Display figures
Drawing: Maintain figures

MiniDraw: Puzzle

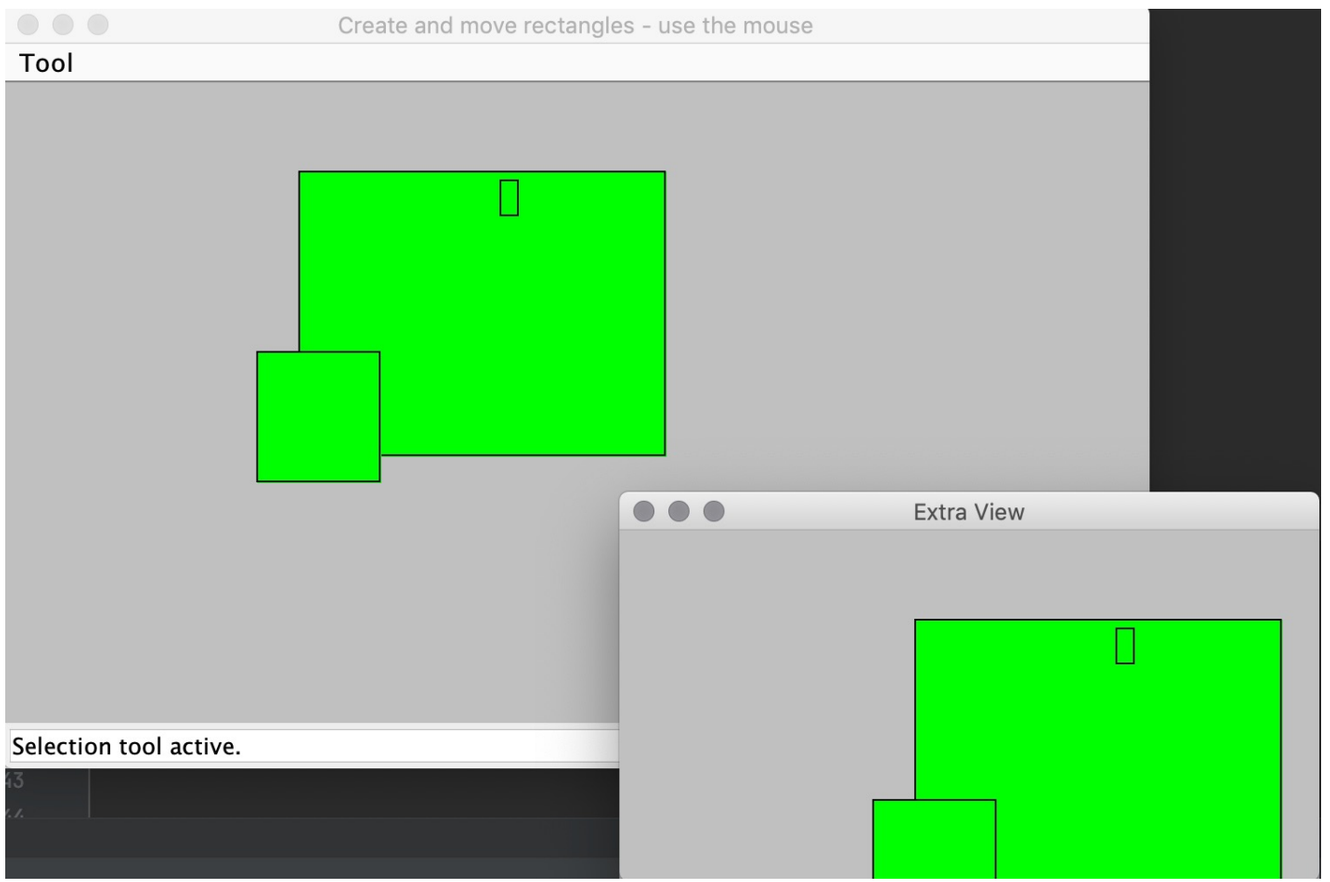
Use of factory in MiniDrawApplication:

```
public void open() {
  Container pane = getContentPane();

  // create the underlying model in the MVC triad
  fDrawing = factory.createDrawing( editor: this);

  // create a view for the MVC
  fView = factory.createDrawingView( editor: this);
  statusField = factory.createStatusField( editor: this);

  JPanel panel = createContents(fView, statusField);

  pane.add(panel);

  pack();
  setVisible(true);
}
```



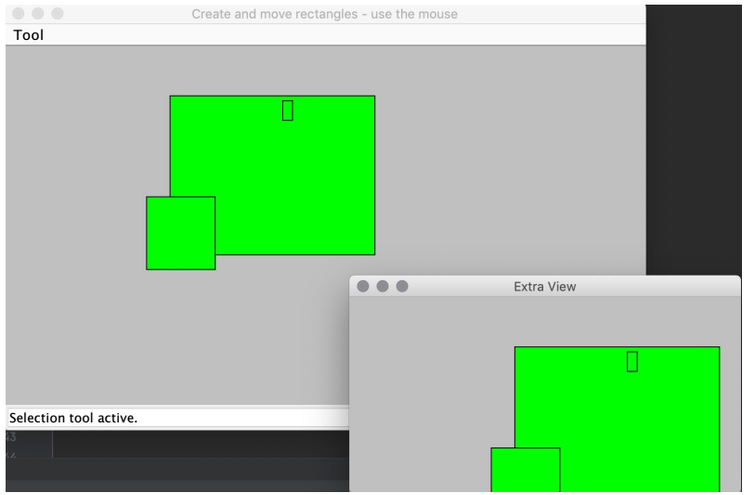
Factory:
Create concrete implementations of DrawingView and Drawing (and optional status text field)

MiniDraw: Rectangles



MiniDraw: Rectangles

```
42  public class ShowRectangle {
43
44    public static void main(String[] args) {
45      Factory f = new EmptyCanvasFactory();
46      DrawingEditor editor =
47        new MiniDrawApplication( title: "Create and move rectangles "+
48                                 "- use the mouse", f );
49      Tool
50        rectangleDrawTool = new RectangleTool(editor),
51        selectionTool = new SelectionTool(editor);
52      addToolSelectMenusToWindow( editor,
53                                  rectangleDrawTool,
54                                  selectionTool );
55      editor.open();
56
57      editor.setTool( rectangleDrawTool );
58      editor.showStatus( "MiniDraw version: "+DrawingEdit
59
60      // create second view
61      JFrame newWindow = new JFrame( title: "Extra View");
62      newWindow.setLocation( x: 620, y: 20 );
63      newWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
64
65      DrawingView extraView = f.createDrawingView(editor);
66      JPanel panel = (JPanel) extraView;
67      newWindow.getContentPane().add(panel);
68      newWindow.pack();
69      newWindow.setVisible(true);
70    }
```



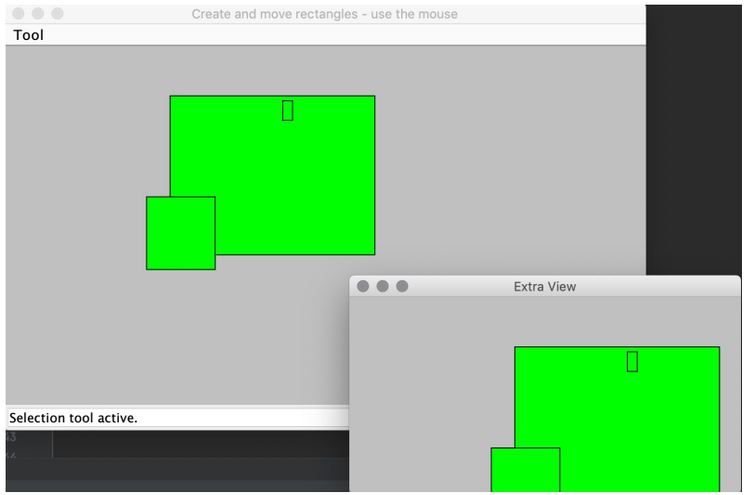
Instantiate DrawingEditor

Open DrawingEditor

Set Tool

MiniDraw: Rectangles

```
42  public class ShowRectangle {
43
44    public static void main(String[] args) {
45      Factory f = new EmptyCanvasFactory();
46      DrawingEditor editor =
47        new MiniDrawApplication( title: "Create and move rectangles "+
48                                 "- use the mouse", f );
49      Tool
50        rectangleDrawTool = new RectangleTool(editor),
51        selectionTool = new SelectionTool(editor);
52      addToolSelectMenusToWindow( editor,
53                                  rectangleDrawTool,
54                                  selectionTool );
55      editor.open();
56
57      editor.setTool( rectangleDrawTool );
58      editor.showStatus( "MiniDraw version: "+DrawingEdit
59
60      // create second view
61      JFrame newWindow = new JFrame( title: "Extra View");
62      newWindow.setLocation( x: 620, y: 20 );
63      newWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
64
65      DrawingView extraView = f.createDrawingView(editor);
66      JPanel panel = (JPanel) extraView;
67      newWindow.getContentPane().add(panel);
68      newWindow.pack();
69      newWindow.setVisible(true);
70    }
```



Instantiate DrawingEditor

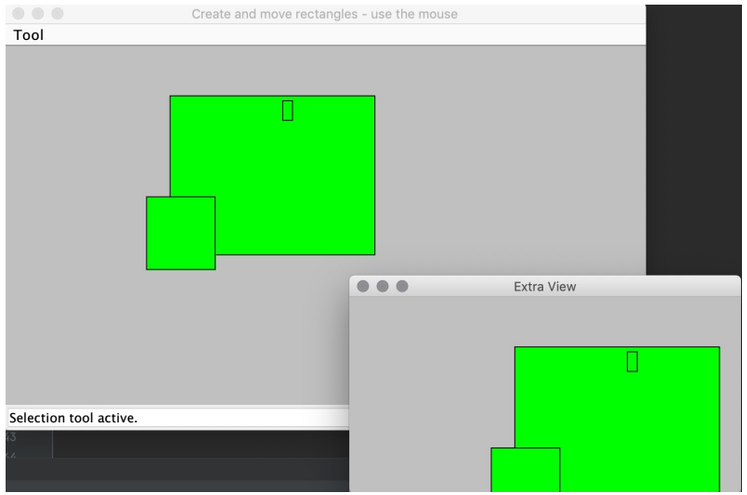
Open DrawingEditor

Set Tool

→ Template

MiniDraw: Rectangles

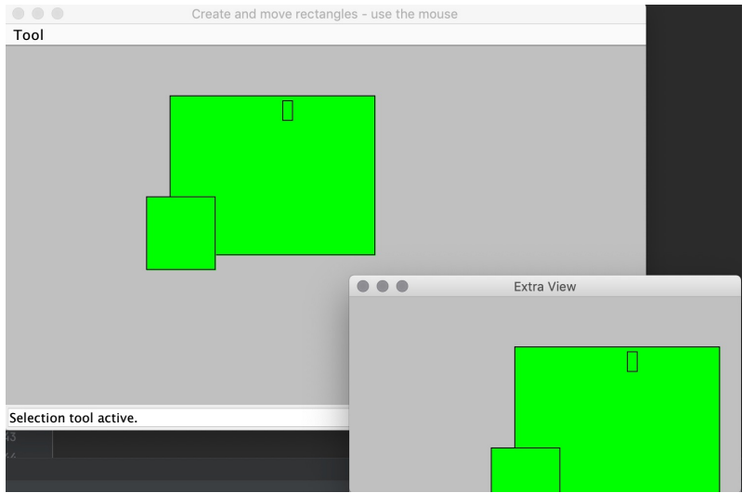
```
42  public class ShowRectangle {
43
44    public static void main(String[] args) {
45      Factory f = new EmptyCanvasFactory();
46      DrawingEditor editor =
47        new MiniDrawApplication( title: "Create and move rectangles "+
48                                 "- use the mouse", f );
49      Tool
50        rectangleDrawTool = new RectangleTool(editor),
51        selectionTool = new SelectionTool(editor);
52      addToolSelectMenusToWindow( editor,
53                                  rectangleDrawTool,
54                                  selectionTool );
55      editor.open();
56
57      editor.setTool( rectangleDrawTool );
58      editor.showStatus( "MiniDraw version: "+DrawingEditor.VERSION );
59
60      // create second view
61      JFrame newWindow = new JFrame( title: "Extra View");
62      newWindow.setLocation( x: 620, y: 20 );
63      newWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
64
65      DrawingView extraView = f.createDrawingView(editor);
66      JPanel panel = (JPanel) extraView;
67      newWindow.getContentPane().add(panel);
68      newWindow.pack();
69      newWindow.setVisible(true);
70    }
```



Instantiate DrawingEditor

EmptyCanvasFactory

MiniDraw: Rectangles

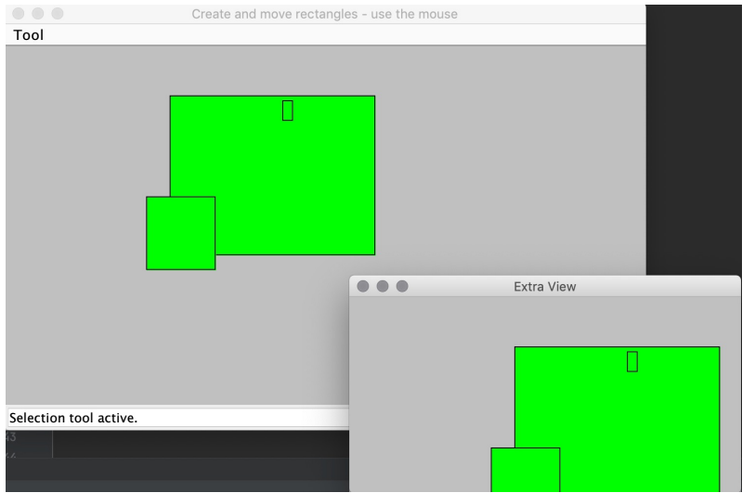


EmptyCanvasFactory

```
class EmptyCanvasFactory implements Factory {

  public DrawingView createDrawingView( DrawingEditor editor ) {
    return new StandardDrawingView(editor, new Dimension( width: 400, height: 200));
  }

  public Drawing createDrawing( DrawingEditor editor ) { return new StandardDrawing(); }

  public JTextField createStatusField( DrawingEditor editor ) { return new JTextField(); }
}
```

MiniDraw: Rectangles

```
42  public class ShowRectangle {
43
44    public static void main(String[] args) {
45      Factory f = new EmptyCanvasFactory();
46      DrawingEditor editor =
47        new MiniDrawApplication( title: "Create and move rectangles "+
48                                 "- use the mouse", f );
49      Tool
50        rectangleDrawTool = new RectangleTool(editor),
51        selectionTool = new SelectionTool(editor);
52      addToolSelectMenusToWindow( editor,
53                                  rectangleDrawTool,
54                                  selectionTool );
55      editor.open();
56
57      editor.setTool( rectangleDrawTool );
58      editor.showStatus( "MiniDraw version: "+DrawingEditor.VERSION );
59
60      // create second view
61      JFrame newWindow = new JFrame( title: "Extra View");
62      newWindow.setLocation( x: 620, y: 20 );
63      newWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
64
65      DrawingView extraView = f.createDrawingView(editor);
66      JPanel panel = (JPanel) extraView;
67      newWindow.getContentPane().add(panel);
68      newWindow.pack();
69      newWindow.setVisible(true);
70    }
```

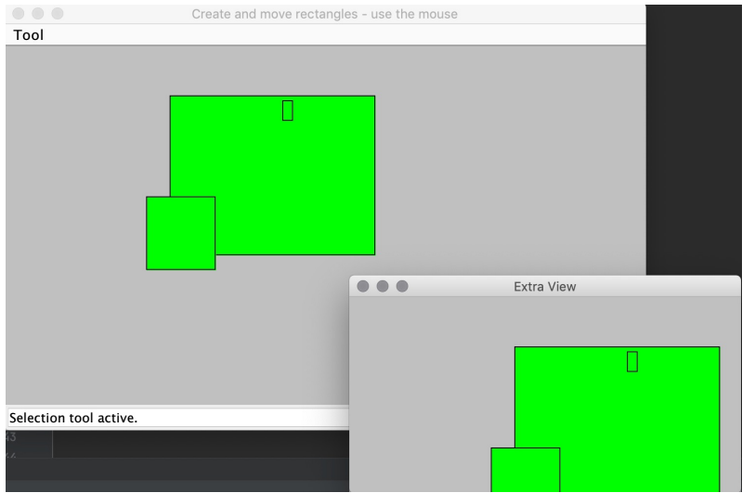


RectangleTool:
Draws rectangles

SelectionTool:
Select/move rectangles

MiniDraw: Rectangles

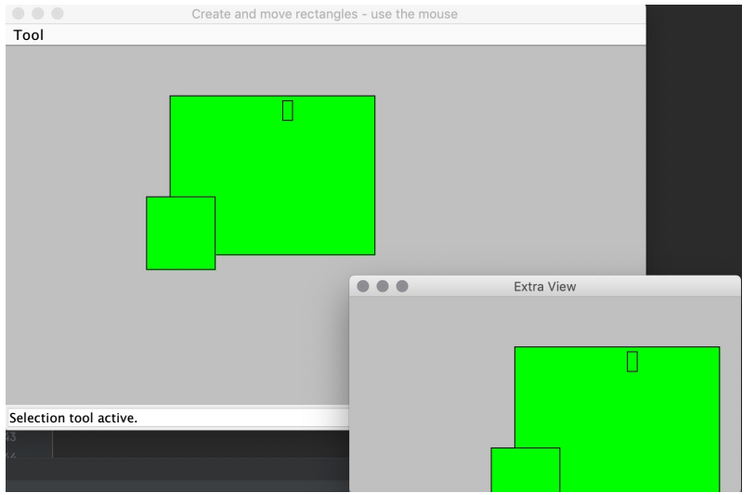
```
123   /** A tool to create rectangles */
124   class RectangleTool extends AbstractTool {
125     private Point corner;
126     private RectangleFigure f;
127
128     public RectangleTool(DrawingEditor editor) {
129       super(editor);
130       f = null;
131     }
132
133     public void mouseDown(MouseEvent e, int x, int y) {
134       super.mouseDown(e,x,y);
135       f = new RectangleFigure( new Point(x,y) );
136       editor.drawing().add(f);
137     }
138
139     public void mouseDrag(MouseEvent e, int x, int y) {
140       f.resize(new Point(fAnchorX, fAnchorY),
141                new Point(x,y) );
142     }
143
144     public void mouseUp(MouseEvent e, int x, int y) {
145       if (f.displayBox().isEmpty()) {
146         editor.drawing().remove(f);
147       }
148       f = null;
149     }
150   }
```



RectangleTool:
Draws rectangles

RectangleFigure:
Green rectangles
(extends AbstractFigure)

MiniDraw: Rectangles

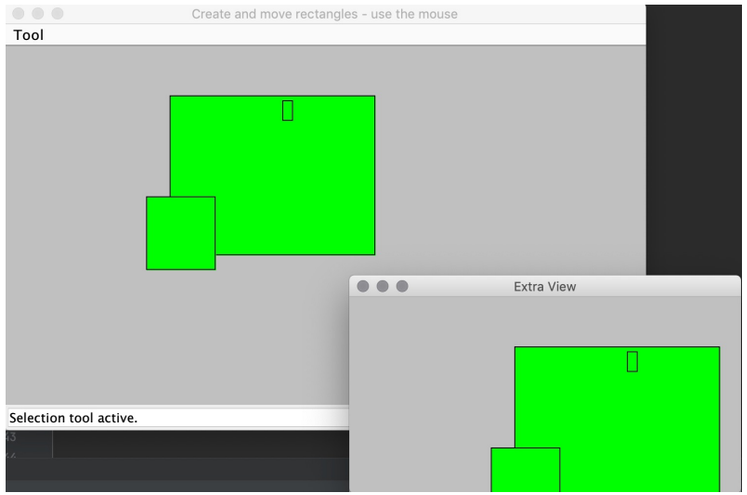


SelectionTool:
Part of MiniDraw

```
package minidraw.standard;

import ...

Selection tool: Uses a internal state pattern to define what type of tool to use in the current situation.

public class SelectionTool extends AbstractTool implements Tool {

    Sub tool to delegate to. The selection tool is in itself a state tool that may be in one of several states
    given by the sub tool. Class Invariant: fChild tool is never null
    protected Tool fChild;

    helper null tool to avoid creating and destroying objects all the time
    protected Tool cachedNullTool;
```

MiniDraw: Rectangles

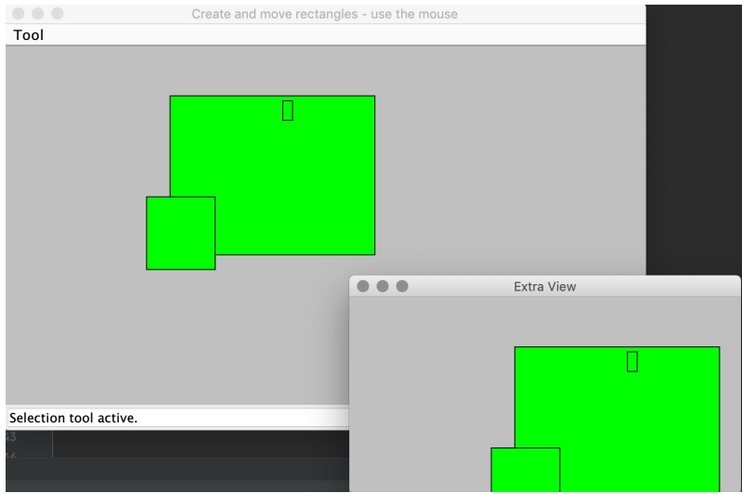
```
42  public class ShowRectangle {
43
44    public static void main(String[] args) {
45      Factory f = new EmptyCanvasFactory();
46      DrawingEditor editor =
47        new MiniDrawApplication( title: "Create and move rectangles "+
48                                 "- use the mouse", f );
49      Tool
50        rectangleDrawTool = new RectangleTool(editor),
51        selectionTool = new SelectionTool(editor);
52      addToolSelectMenusToWindow( editor,
53                                  rectangleDrawTool,
54                                  selectionTool );
55      editor.open();
56
57      editor.setTool( rectangleDrawTool );
58      editor.showStatus( "MiniDraw version: "+DrawingEditor.VERSION );
59
60      // create second view
61      JFrame newWindow = new JFrame( title: "Extra View");
62      newWindow.setLocation( x: 620, y: 20 );
63      newWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
64
65      DrawingView extraView = f.createDrawingView(editor);
66      JPanel panel = (JPanel) extraView;
67      newWindow.getContentPane().add(panel);
68      newWindow.pack();
69      newWindow.setVisible(true);
70    }
```



Second window acts as second view
Views are synchronized

MiniDraw: Rectangles

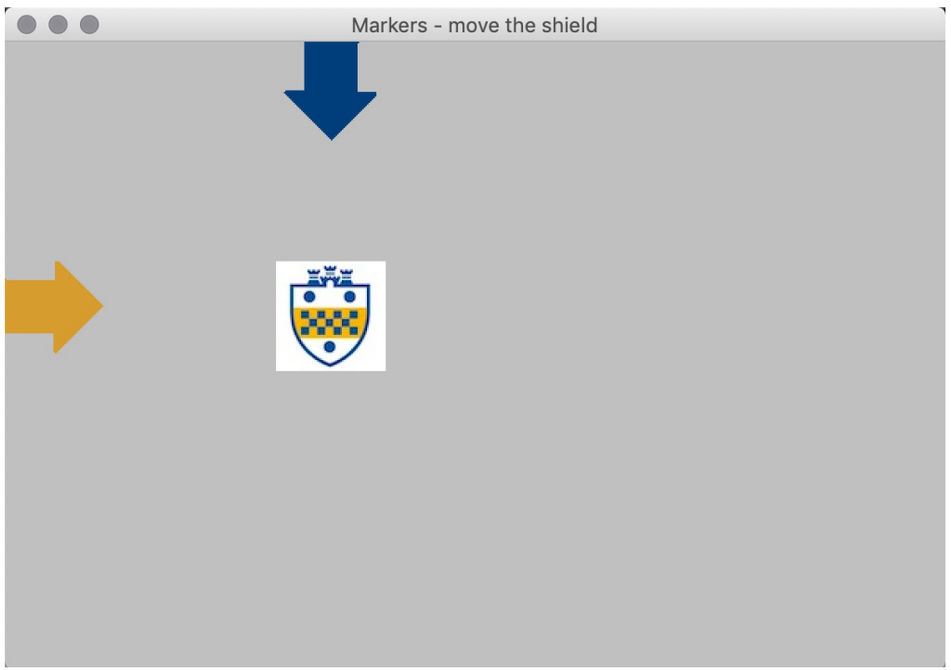
```
42  public class ShowRectangle {
43
44    public static void main(String[] args) {
45      Factory f = new EmptyCanvasFactory();
46      DrawingEditor editor =
47        new MiniDrawApplication( title: "Create and move rectangles "+
48                                 "- use the mouse", f );
49      Tool
50        rectangleDrawTool = new RectangleTool(editor),
51        selectionTool = new SelectionTool(editor);
52      addToolSelectMenusToWindow( editor,
53                                  rectangleDrawTool,
54                                  selectionTool );
55      editor.open();
56
57      editor.setTool( rectangleDrawTool );
58      editor.showStatus( "MiniDraw version: "+DrawingEditor.VERSION );
59
60      // create second view
61      JFrame newWindow = new JFrame( title: "Extra View");
62      newWindow.setLocation( x: 620, y: 20 );
63      newWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
64
65      DrawingView extraView = f.createDrawingView(editor);
66      JPanel panel = (JPanel) extraView;
67      newWindow.getContentPane().add(panel);
68      newWindow.pack();
69      newWindow.setVisible(true);
70    }
```

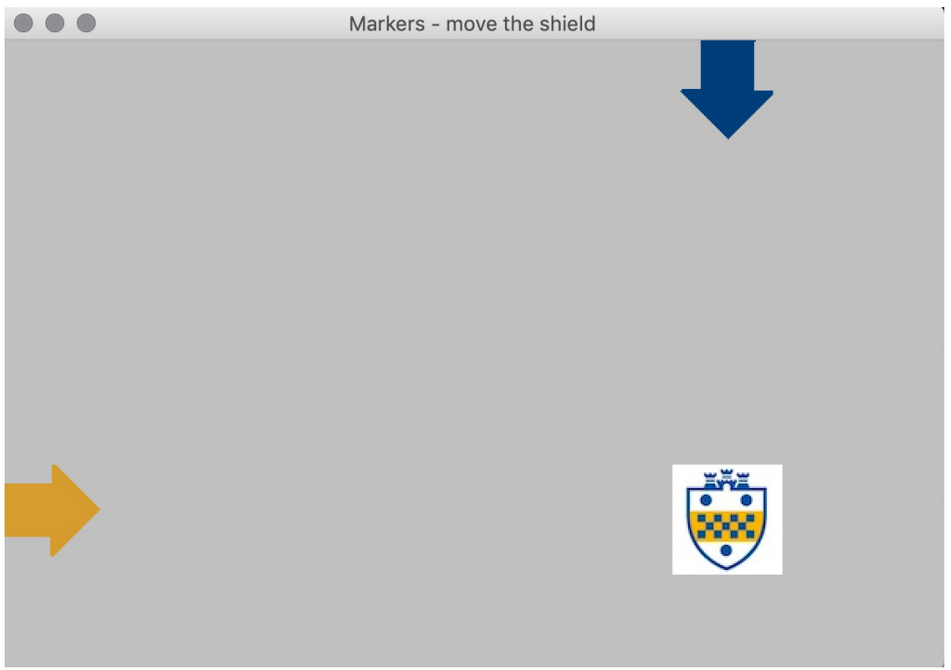


Second window acts as second view
Views are synchronized

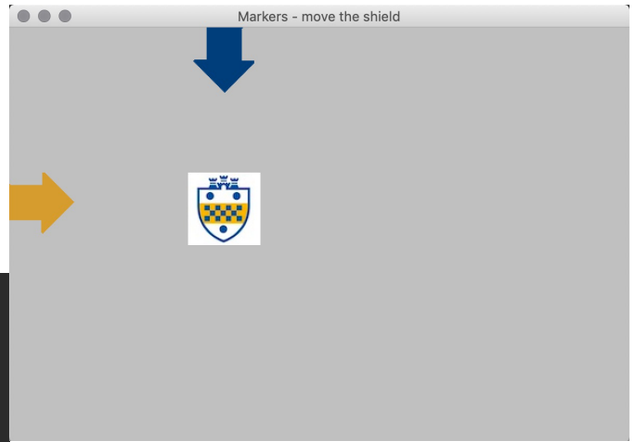
→ Observer

MiniDraw: Markers





MiniDraw: Markers



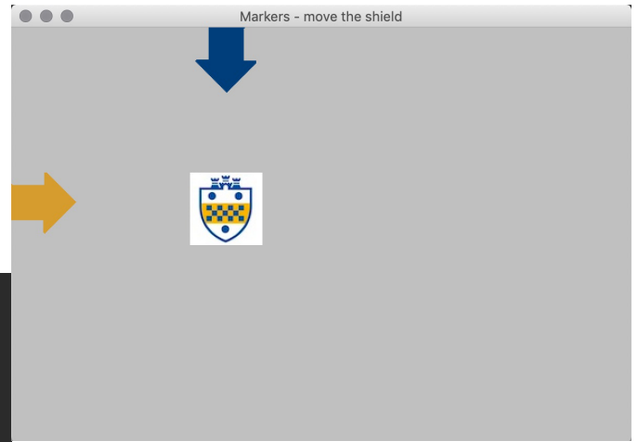

```
46  public class Markers {
47    public static void main(String[] args) {
48      Factory f = new EmptyCanvasFactory();
49      DrawingEditor editor =
50        new MiniDrawApplication( title: "Markers - move the shield", f );
51
52      editor.open();
53      Figure logo = new ImageFigure( imagename: "pitt-shield-small", new Point( x: 200,  y: 200));
54
55      Figure rightArrow =
56        new MarkerFigureDecorator( new ImageFigure( imagename: "arrow-right",
57                                                    new Point( x: 0,  y: 200)),
58                                   logo,
59                                   horizontal: false );
60      Figure downArrow =
61        new MarkerFigureDecorator( new ImageFigure( imagename: "arrow-down",
62                                                    new Point( x: 200,  y: 0)),
63                                   logo,
64                                   horizontal: true );
65
66      editor.setTool( new SelectionTool(editor) );
67
68      editor.drawing().add(rightArrow);
69      editor.drawing().add(downArrow);
70      editor.drawing().add(logo);
71    }
72  }
```

Instantiate DrawingEditor

Open DrawingEditor

Set Tool

MiniDraw: Markers




```
46  public class Markers {
47    public static void main(String[] args) {
48      Factory f = new EmptyCanvasFactory();
49      DrawingEditor editor =
50        new MiniDrawApplication( title: "Markers - move the shield", f );
51
52      editor.open();
53      Figure logo = new ImageFigure( imagename: "pitt-shield-small", new Point( x: 200,  y: 200));
54
55      Figure rightArrow =
56        new MarkerFigureDecorator( new ImageFigure( imagename: "arrow-right",
57                                                    new Point( x: 0,  y: 200)),
58                                   logo,
59                                   horizontal: false );
60      Figure downArrow =
61        new MarkerFigureDecorator( new ImageFigure( imagename: "arrow-down",
62                                                    new Point( x: 200,  y: 0)),
63                                   logo,
64                                   horizontal: true );
65
66      editor.setTool( new SelectionTool(editor) );
67
68      editor.drawing().add(rightArrow);
69      editor.drawing().add(downArrow);
70      editor.drawing().add(logo);
71    }
72  }
```

MarkerFigureDecorator

```
public MarkerFigureDecorator(Figure decoratee,
                             Figure target,
                             boolean horizontal )
```

MiniDraw: Markers


```
34      public class MarkerFigureDecorator implements Figure {
35
36        private Figure decoratee;
37        private boolean horizontal;
38
39  @     public MarkerFigureDecorator(Figure decoratee,
40                                     Figure target,
41                                     boolean horizontal ) {
42          this.decoratee = decoratee;
43          this.horizontal = horizontal;
44          Observer o = new Observer();
45          target.addFigureChangeListener(o);
46        }
47
48        public void draw(Graphics g) { decoratee.draw(g); }
49        public Rectangle displayBox() { return decoratee.displayBox(); }
50        public void moveBy(int dx, int dy) { decoratee.moveBy(dx,dy); }
51        public void invalidate() { decoratee.invalidate(); }
52        public void changed() { decoratee.changed(); }
53        public void addFigureChangeListener(FigureChangeListener l) {
54          decoratee.addFigureChangeListener(l);
55        }
56        public void removeFigureChangeListener(FigureChangeListener l) {
57          decoratee.removeFigureChangeListener(l);
58        }
59
```

addFigureChangeListener: Register observers of figure change events (state changes)

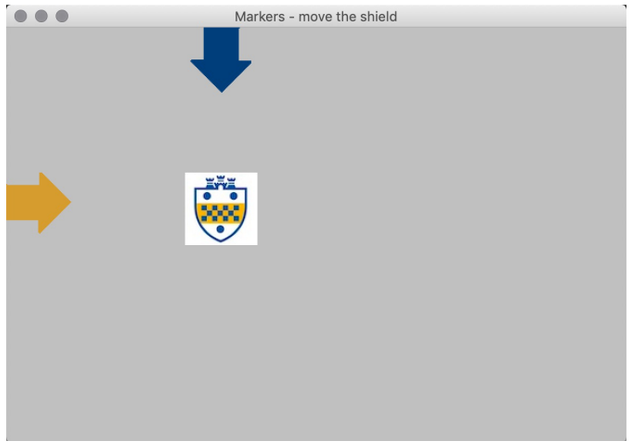
MiniDraw: Markers


```
34      public class MarkerFigureDecorator implements Figure {
35
36        private Figure decoratee;
37        private boolean horizontal;
38
39    @   public MarkerFigureDecorator(Figure decoratee,
40                                     Figure target,
41                                     boolean horizontal ) {
42          this.decoratee = decoratee;
43          this.horizontal = horizontal;
44          Observer o = new Observer();
45          target.addFigureChangeListener(o);
46        }
47
48        public void draw(Graphics g) { decoratee.draw(g); }
49        public Rectangle displayBox() { return decoratee.displayBox(); }
50        public void moveBy(int dx, int dy) { decoratee.moveBy(dx,dy); }
51        public void invalidate() { decoratee.invalidate(); }
52        public void changed() { decoratee.changed(); }
53        public void addFigureChangeListener(FigureChangeListener l) {
54          decoratee.addFigureChangeListener(l);
55        }
56        public void removeFigureChangeListener(FigureChangeListener l) {
57          decoratee.removeFigureChangeListener(l);
58        }
59
```

addFigureChangeListener: Register observers of figure change events (state changes)

Listener = Observer

MiniDraw: Markers



In MarkerFigureDecorator:

```
target.addFigureChangeListener(o);
```

From main class, target is an ImageFigure:

```
Figure logo = new ImageFigure( imagename: "pitt-shield-small", new Point( x: 200, y: 200)
```

```
public class ImageFigure extends AbstractFigure {
```

From AbstractFigure:

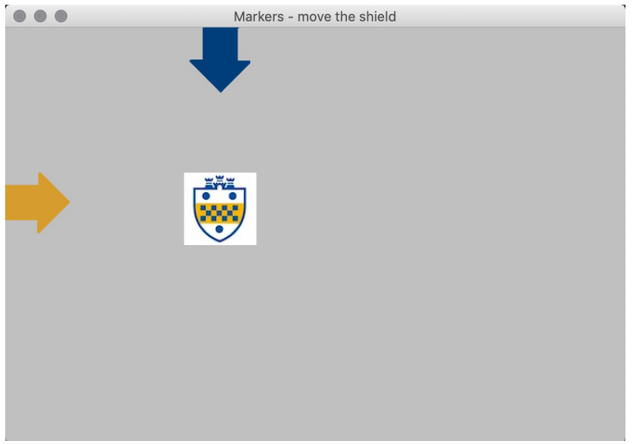
```
public void addFigureChangeListener(FigureChangeListener l) { listenerList.add(l); }
```

```
public void changed() {
  invalidate();
  for (FigureChangeListener l : listenerList) {
    FigureChangeEvent e = new FigureChangeEvent( source: this);
    l.figureChanged(e);
  }
}
```

MiniDraw: Markers


```
34      public class MarkerFigureDecorator implements Figure {
35
36        private Figure decoratee;
37        private boolean horizontal;
38
39    @   public MarkerFigureDecorator(Figure decoratee,
40                                     Figure target,
41                                     boolean horizontal ) {
42          this.decoratee = decoratee;
43          this.horizontal = horizontal;
44          Observer o = new Observer();
45          target.addFigureChangeListener(o);
46        }
47
48        public void draw(Graphics g) { decoratee.draw(g); }
49        public Rectangle displayBox() { return decoratee.displayBox(); }
50        public void moveBy(int dx, int dy) { decoratee.moveBy(dx,dy); }
51        public void invalidate() { decoratee.invalidate(); }
52        public void changed() { decoratee.changed(); }
53        public void addFigureChangeListener(FigureChangeListener l) {
54          decoratee.addFigureChangeListener(l);
55        }
56        public void removeFigureChangeListener(FigureChangeListener l) {
57          decoratee.removeFigureChangeListener(l);
58        }
59
```

MiniDraw: Markers




```
60      private class Observer implements FigureChangeListener {
61        public void figureInvalidated(FigureChangeEvent e){
62        }
63        public void figureChanged(FigureChangeEvent e){
64          Figure f = e.getFigure();
65          Rectangle target_r = f.displayBox();
66          Rectangle marker_r = decoratee.displayBox();
67          int dx = 0, dy = 0;
68          if ( horizontal ) {
69            dx = target_r.x - marker_r.x;
70          } else {
71            dy = target_r.y - marker_r.y;
72          }
73          decoratee.moveBy(dx, dy);
74        }
75        public void figureRemoved(FigureChangeEvent e){}
76        public void figureRequestRemove(FigureChangeEvent e){}
77        public void figureRequestUpdate(FigureChangeEvent e){}
78      }
79    }
```

MiniDraw: Variability Points

Images

→Custom images, loaded from a specific folder (src/main/resources/minidraw-images)

Tools

→Tell the editor which tool to use to handle figures in different ways

Figures

→Custom Figures that do their own graphical rendering

Views

→Custom DrawingViews that provide special rendering

Drawings

→Customize collection implementation to store and remove figures

Observers

→ Make other objects listen to state changes in figures (including other figures)

Next Time: MiniDraw + HotCiv


```
44  task show(type: JavaExec) {
45    group 'HotCiv Demonstration'
46    description 'Demonstrate MapView'
47
48    main = 'hotciv.visual.ShowWorld'
49    classpath = sourceSets.main.runtimeClasspath
50  }
51
52  task text(type: JavaExec) {
53    group 'HotCiv Demonstration'
54    description 'Demonstrate TextFigure'
55
56    main = 'hotciv.visual.ShowText'
57    classpath = sourceSets.main.runtimeClasspath
58  }
59
60  task city(type: JavaExec) {
61    group 'HotCiv Demonstration'
62    description 'Demonstrate CityFigure'
63
64    main = 'hotciv.visual.ShowCity'
65    classpath = sourceSets.main.runtimeClasspath
66  }
```