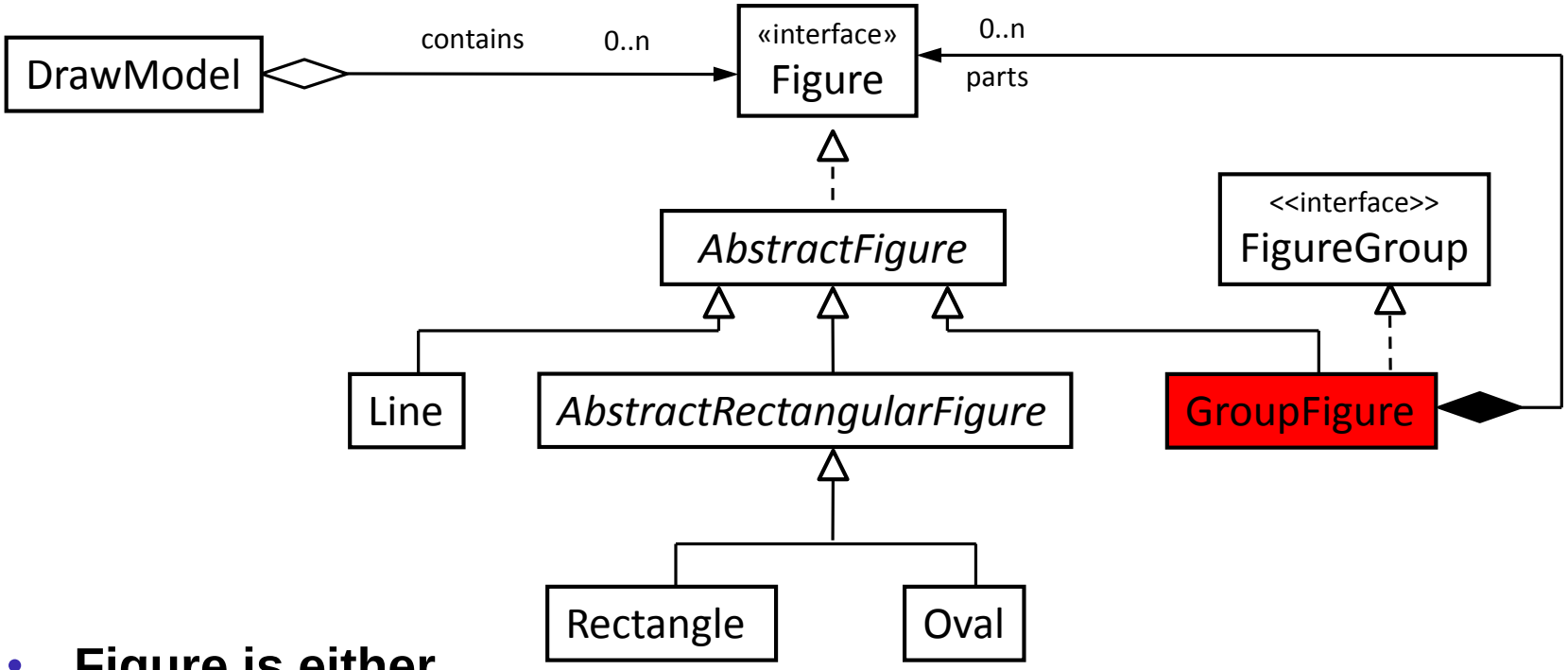
Assignment 6: Group Figures



- **Figure is either**
  - in the list of the draw model (ungrouped), or
  - in the list of a figure group (grouped) but not in the model

Class GroupFigure (1)

```
public class GroupFigure extends AbstractFigure implements FigureGroup {
   private final List<Figure> parts;

   public GroupFigure(List<Figure> selectedFigures) {
      if(selectedFigures == null || selectedFigures.size() == 0)
         throw new IllegalArgumentException();
      this.parts = new ArrayList<>(selectedFigures);
   }

   public Rectangle getBounds() {
      Rectangle bounds = parts.get(0).getBounds();
      parts.stream().skip(1).forEach(
         f -> bounds.add(f.getBounds())
      );
      return bounds;
   }
   …
```

Copy constructor

Initialization of bounding box, no NPE as parts.size() > 0

Union of the bounding boxes of all figures

Class GroupFigure (2)

```
public boolean contains(int x, int y) {
   return getBounds().contains(x, y);
}

public boolean contains(int x, int y) {
   for (Figure f : parts) {
      if (f.contains(x, y)) { return true; }
   }
   return false;
}

public boolean contains(int x, int y) {
   return parts
            .stream()
            .anyMatch(f -> f.contains(x, y));
}

…
```

Variants for the implementation of contains

Class GroupFigure (3)

```
public void draw(Graphics g) {
   parts.forEach(f -> f.draw(g));
}

public void move(int dx, int dy) {
   if(dx != 0 || dy != 0) {
      parts.forEach(f -> f.move(dx, dy));
      notifyChange();
   }
}


public void setBounds(Point origin, Point corner) {
   // if setBounds is implemented, then original size
   // relations must be stored
}

…
```

Necessary as model has not registered a listener in the figure parts, only in the group

Class GroupFigure (4)

```
   public List<FigureHandle> getHandles() {
      // if setBounds is supported then RectangularFigure Handles
      // may be used;
      // as an alternative the handles may be used to only move
      // the group.
   }

   public Iterable<Figure> getFigureParts() {
      return Collections.unmodifiableList(parts);
   }
}
```

Used for the ungrouping operation

Group Action

```
JMenuItem group = new JMenuItem("Group");
editMenu.add(group);
group.addActionListener(e -> {
   List<Figure> selection = getView().getSelection();
   if (selection != null && selection.size() >= 2) {
      GroupFigure g = new GroupFigure(new ArrayList<>(selection));
      DrawModel m = getView().getModel();
      for (Figure f : selection) {
         m.removeFigure(f);
      }
      m.addFigure(g);

      getView().addToSelection(g);
   }
});
```

Removes parts from the model

Adds group figure to the model and to the selection

Ungroup Action

```
JMenuItem ungroup = new JMenuItem("Ungroup");
editMenu.add(ungroup);
ungroup.addActionListener(e -> {
   for (Figure g : getView().getSelection()){
      if (g instanceof FigureGroup) {
         getModel().removeFigure(g);
         for (Figure f : ((FigureGroup)g).getFigureParts()) {
            getModel().addFigure(f);
            getView().addToSelection(f);
         }
      }
   }
});
```

Removes group figure

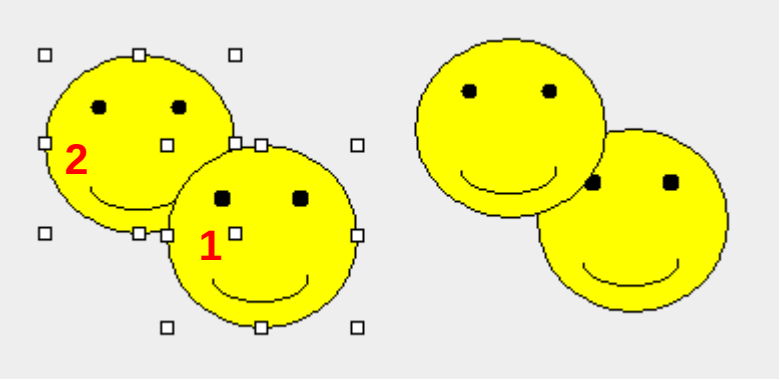
Adds figure parts to the model and to the selection

Order of Figures

```
public class GroupFigure extends AbstractFigure implements FigureGroup {
   private final List<Figure> parts;

   public GroupFigure(List<Figure> selectedFigures) {
      this.parts = new ArrayList<>(selectedFigures);
   }
```

- The order of the figures in the selection is the selection order
- Z-order of the figures may be changed



Order of Figures

```
public class GroupFigure extends AbstractFigure implements FigureGroup {
   private final List<Figure> parts;

   public GroupFigure(List<Figure> figures, DrawModel model) {
      if (figures == null || figures.size() == 0) {
         throw new IllegalArgumentException();
      }
      this.parts = new ArrayList<>();
      for(Figure f : model.getFigures()) {
         if(figures.contains(f)) { parts.add(f); }
      }
   }

   …
```

Same order of the figure parts as in the model