Assignment 7: Copy & Paste

- **Implementation of the Figure.clone() methods**
  - AbstractFigure
  - Line
  - AbstractRectangularFigure
  - GroupFigure
- **Implementation of Cut-, Copy- and Paste-Actions**

Clone in class AbstractFigure

```
@Override
public Object clone() {
  return null;
}
```

```
@Override
public AbstractFigure clone() {
  try {
    AbstractFigure copy = (AbstractFigure) super.clone();
    copy.listeners = new ArrayList<>();
    return copy;
  } catch (CloneNotSupportedException e) {
    throw new InternalError();
  }
}
```

Listeners must not be cloned, they do not belong to the cloneable state of the figure, they are set by the model (i.e. by the context in which the figure is used).

Clone needs own (empty) list of listeners.

Clone in class Line

```
/* List of handles, lazily allocated in getHandles. */
private List<FigureHandle> handles;

/* internal representation for the coordinates of the line. */
private Line2D line;

@Override
public Line clone() {
  Line copy = (Line) super.clone();
  copy.line = (Line2D) copy.line.clone();
  copy.handles = null;
  return copy;
}
```

The `CloneNotSupportedException` needs not be catched as this exception is already handled in class `AbstractFigure`.

Line2D-object must be copied (deep copy)

If a handle cache has been defined, then this cache must be cleared, will be recreated on the next invocation of method `getHandles`.

Clone in class AbstractRectangularFigure

```
@Override
public AbstractRectangularFigure clone() {
   AbstractRectangularFigure copy =
        (AbstractRectangularFigure) super.clone();
   copy.rectangle = (Rectangle) copy.rectangle.clone();
   copy.handles = null;
   return copy;
}
```

Deep copy: internal rectangle must be cloned.

Handle cache is cleared; it will be created upon the next invocation of method `getHandles`.

- Method clone only needs be overridden in the concrete rectangular figures (i.e. `Oval` and `Rectangle`), …
  - if these contain additional state or
  - if the result type needs to be adjusted.

Clone in class GroupFigure

```
@Override
public GroupFigure clone() {
  GroupFigure copy = (GroupFigure) super.clone();
  copy.parts = new ArrayList<>(parts.size());
  for (Figure f : parts) {
    copy.parts.add(f.clone());
  }
  return copy;
}
```

All figures of the group have to be cloned

- Additional information (like original size constraints) not necessarily have to be cloned, as they remain the same for all copies (as long as no figures can be added to or removed from a group)

Simple Clipboard

```
public final class SimpleClipboard {
  private final List<Figure> figures = new ArrayList<>();

  public void add(Figure figure) { figures.add(figure); }

  public List<Figure> get() { return figures; }

  public void clear() { figures.clear(); }
}
```

```
private SimpleClipboard clipboard = new SimpleClipboard();
```

Cut / Copy

```
cut.addActionListener(e -> {
   clipboard.clear();
   for(Figure f : getView().getSelection()) {
      clipboard.add(f);

      getModel().removeFigure(f);
   }
});

copy.addActionListener(e -> {
   clipboard.clear();
   for(Figure f : getView().getSelection()) {
      clipboard.add(f.clone());
   }
});
```

At a cut operation, the selection does not have to be copied.
The selection is removed from the model (and thus also from the selection).

Figures must be cloned if they are copied, as the original figure might be changed after the copy was made.

Paste

```
paste.addActionListener(e -> {
   getView().clearSelection();
   for(Figure f : clipboard.get()) {
      Figure f2 = f.clone();
      getModel().addFigure(f2);
      getView().addToSelection(f2);
   }
});
```

1) Clear current selection

2) Add a ***copy*** of the figures in the clipboard into the draw model

3) Add the copied figure to the selection