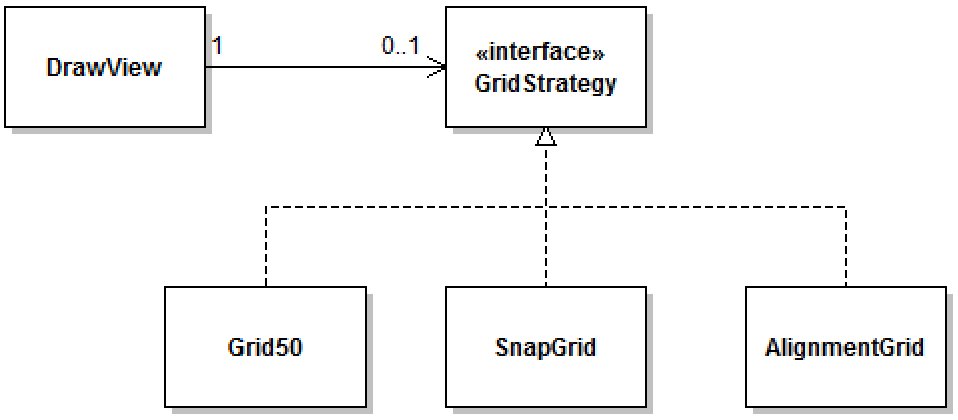
Module Design Patterns
Prof. Dr. Dominik Gruntz
Prof. Dr. Wolfgang Weck

Worksheet: Strategy Pattern

If you have to provide different algorithms to solve a particular problem, then the application of the strategy pattern is appropriate. The context accesses the different implementations over a common strategy interface.

The grid implementation in JDraw is an example of a strategy we looked at in class. The task of these grid algorithms is to map the mouse coordinates onto a "grid".



A first version of the `GridStrategy` interface we came up with looked as follows:

```
interface GridStrategy {
   Point mapPoint(Point p);
}
```

The question however is, whether this interface can serve all imaginable strategies. Try to answer this question with the help of the following imaginable algorithms. First think about which information is used by these different grid implementations to do their task and also think about how they obtain this information:

- **SnapGrid**
  This grid maps the mouse coordinates to the positions of the handles of the other figures if such a handle is sufficiently close to the current mouse position.

- **ModifierGrid**
  This grid behaves differently depending on whether the SHIFT-key is pressed or not. For example, if you move a figure in PowerPoint with the mouse while pressing the SHIFT-key, then the figure can only be moved horizontally or vertically.

- **HistoryGrid**
  A history grid uses the last ten mouse-down and mouse-up positions as the basis for the grid (i.e. it snaps to the last history points only).

How would you extend the signature of method `mapPoint` in order serve the above proposed (and also further) grid implementations?

Note that certain parameters needed by a particular grid implementation could also be passed to its instances using the constructor. What kind of parameters can be passed this way, and what kind of parameters must be passed over the signature of method `mapPoint`?

Advanced task: Conceptually think about how to implement the SnapGrid algorithms. According to your idea, would the implementation be stateful or stateless?