Hinweise zu Übung 5

Übung 5 ist viel leichter als die anderen Übungen. Trotzdem haben wir auch für diese Übung wieder ein Dokument bereitgestellt, welches aufzeigt, wie JDraw mit einer Grid-Implementierung interagiert.

A. Welche Klassen braucht es?

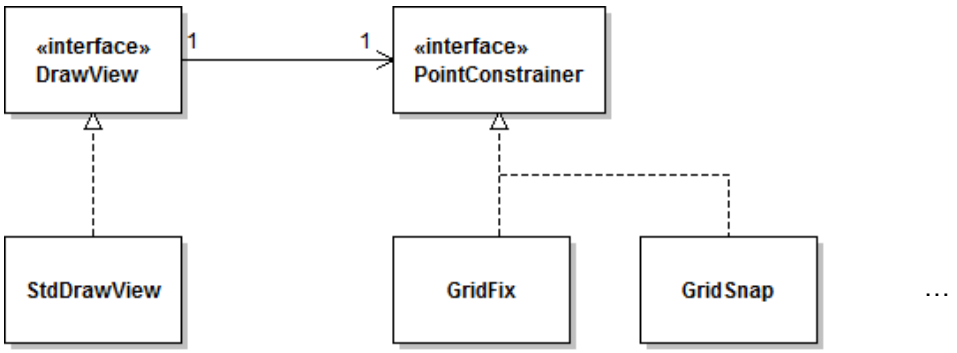
Das folgende Diagramm zeigt Ihnen die Situation, wenn Sie die *DrawView* mit Grids erweitern. Beachten Sie, dass ein Grid jeweils in einem Fenster (also in einer *DrawView*) registriert wird. Falls eine Grafik in mehreren Fenstern angezeigt wird, so kann in jedem Fenster ein anderes Grid registriert werden.

1. PointConstrainer-Klassen

Das Interface `PointConstrainer` definiert die Strategie-Schnittstelle die von jeder Grid-Klasse implementiert werden muss.

2. DrawView-Klasse

Damit *JDraw* Ihre Grids auch verwendet, müssen diese mit der Methode `setConstrainer` gesetzt werden. Das Setzen eines Grids erfolgt typischerweise über ein Menu.



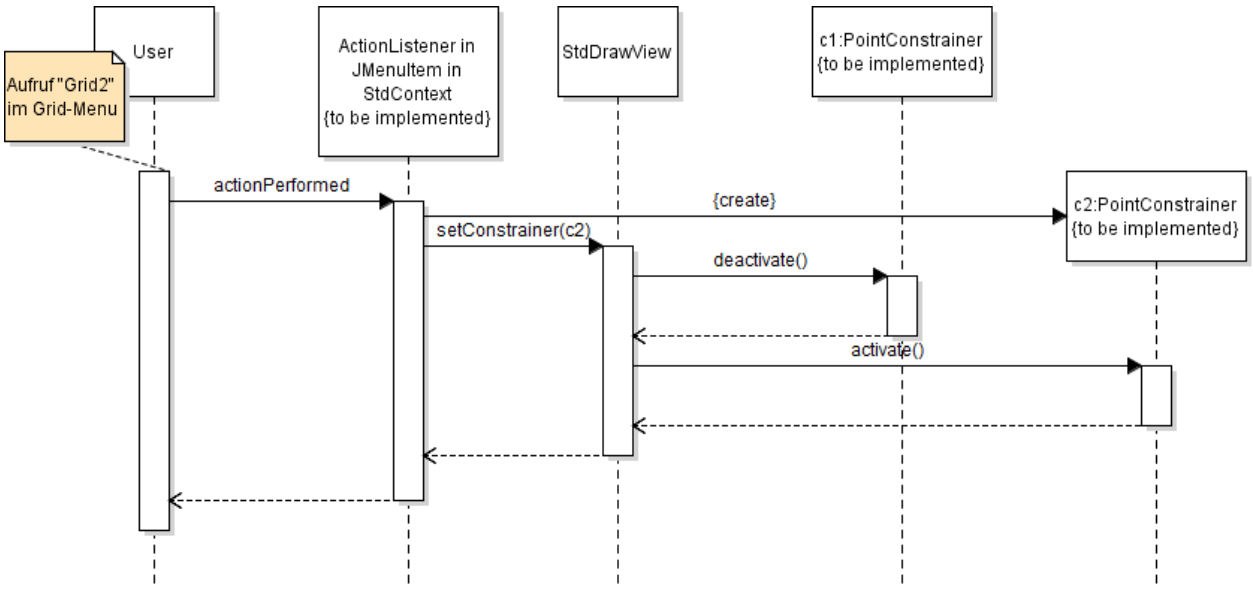
B. Interaktion mit JDraw

In *JDraw* sind die Menus in der Klasse `StdContext` definiert. Ab Zeile 129 ist das Grid-Menu festgelegt, ein Untermenu des Edit-Menüs. Alle Action-Listener, welche die Grid-Strategie festlegen sollen, müssen ein entsprechendes Strategie-Objekt erzeugen und dann die Methode

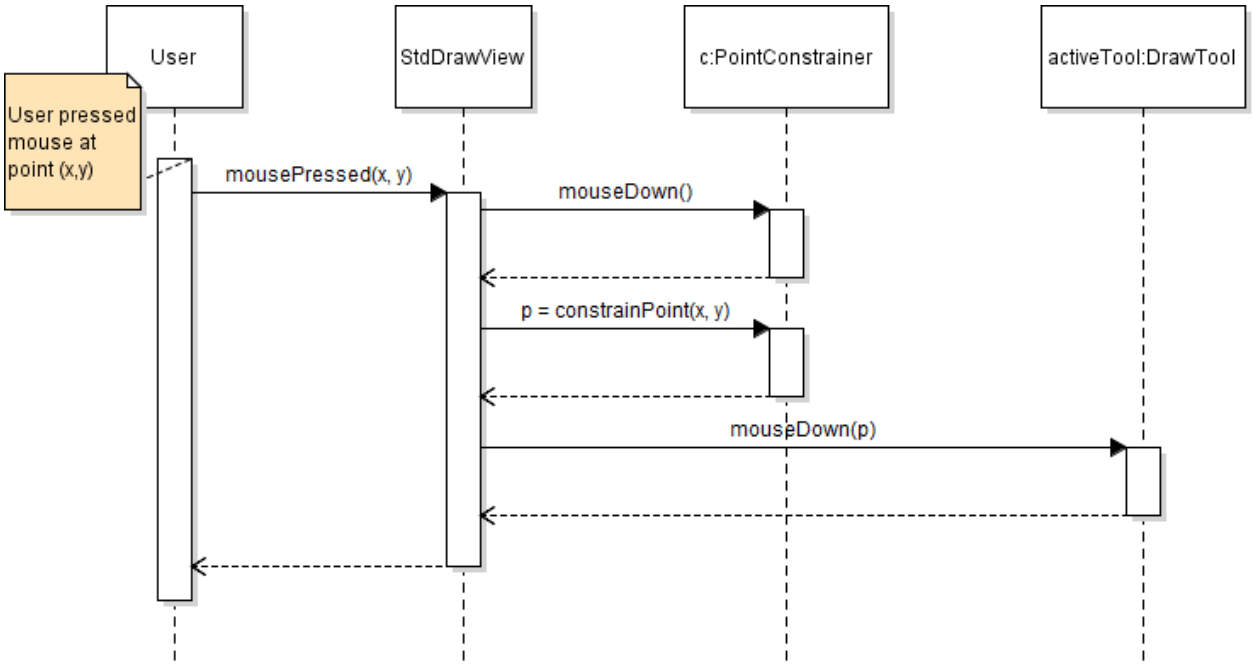
```
getView().setContrainer(…)
```

aufrufen.

Das folgende Sequenzdiagramm zeigt die Abläufe bei der Auswahl des Grids „Grid 2“ im Edit-Menu (wenn aktuell „Grid 1“ gesetzt ist):



Die Methoden, die jedes Strategie-Objekt implementieren muss, sind im Interface `PointConstrainer` definiert und dokumentiert. Ihre Verwendung im Kontext der *JDraw*-Implementierung zeigt das folgende Sequenzdiagramm exemplarisch:



C. Mögliches Vorgehen

In der folgenden Anleitung beschreiben wir, wie Sie ein *SimpleGrid* realisieren und im Menu registrieren können.

1. Grid Klasse programmieren

Erzeugen Sie (z.B. im Paket *jdraw.grid*) die Klasse *SimpleGrid* und implementieren Sie in dieser Klasse das Interface *PointConstrainer*. Sie können auch gleich die fehlenden Methoden von ihrer IDE generieren lassen.

Point constrainPoint(Point p):

In dieser Methode sollen die Koordinaten, an welchen sich die Maus aktuell befinden, auf das Grid abgebildet werden. Geben Sie hier vorerst einfach den Parameter p zurück (das entspricht „keinem Grid“) und geben Sie auf der Konsole aus, dass die Methode aufgerufen worden ist.

```
@Override
public Point constrainPoint(Point p) {
        System.out.println("SimpleGrid:constrainPoint: " + p);
        return p;
}
```

int getStepX(boolean right):

Diese Methode gibt an, um wieviel Pixel die Figur verschoben werden soll wenn die Verschiebung über die Pfeiltasten erfolgt. Hier können Sie ebenfalls ausgeben dass die Methode aufgerufen worden ist und als Resultat z.B. 1 (entspricht dem Normalfall ohne Grid) zurückgeben.

void activate() / void deactivate():
void mouseDown() / void mouseUp():

Geben Sie auch bei diesen Methoden auf der Konsole aus, dass sie aufgerufen worden sind. Für die Methode `activate` sieht das dann wie folgt aus:

```
@Override
public void activate() {
        System.out.println("SimpleGrid:activate()");
}
```

2. **Grid in der DrawView registrieren**
   Das Grid muss jetzt über das Menu gesetzt werden können. Definieren Sie dazu in der Klasse `StdContext` ab Zeile 129 einen entsprechenden Menu-Listener und registrieren sie diesen im Grid-Menu:

   ```
   JMenuItem simpleGrid = new JMenuItem("Simple Grid");
   simpleGrid.addActionListener(e -> getView().setConstrainer(new SimpleGrid()));
   grid.add(simpleGrid);
   ```

   Wenn Sie nun das Simple-Grid auswählen, dann können sie die oben beschriebenen Sequenzdiagramme reproduzieren.

3. **Grid in der DrawView wieder deregistrieren**
   Falls Sie das Simple-Grid einmal gewählt haben, so können Sie dieses Grid nicht mehr entfernen. Sie können das Menu mehrfach ausführen und dabei wird immer eine neue Instanz des Simple-Grids gesetzt (und dabei werden die Methoden `passivate` und `activate` aufgerufen), aber es ist nicht mehr möglich das Grid wieder zu entfernen.

   Dies können Sie korrigieren, indem sie einen weiteren Menu Eintrag im Grids-Menu vorsehen, in welchem in der DrawView der Constrainer *null* gesetzt wird (`getView().setConstrainer(null)`). Sobald in der DrawView der Constrainer *null* gesetzt ist, werden die Mauskoordinaten nicht mehr mit einem *PointConstrainer* abgebildet.

4. **Konkrete Grids implementieren**
   Sie haben jetzt gesehen wie man Grid-Klasse implementieren und in JDraw registrieren kann. Zeit also, ihre eigenen Grids zu implementieren. In der Vorlesung haben Sie bereits das Grid gesehen welches die x- und y-Koordinaten auf Vielfache einer gegebenen Gridgrösse rundet (z.B. auf Vielfache von 20 Pixeln oder 50 Pixeln). Eine weitere Möglichkeit wäre, die Koordinaten auf ein gegebenes Rechteck zu beschränken, oder (und das ist dann etwas schwieriger) an die Koordinaten der Handles der anderen Figuren zu springen, falls diese genügend nahe bei der Mausposition liegen.