Übung 10: Design Patterns 1

Abgabetermin: 11. 6. 2015, 8:15

Übung 11: Design Patterns 2

Abgabetermin: 18. 6. 2015, 8:15

Informatik: ☐ G1 (Prähofer) ☐ G2 (Prähofer) ☐ G3 (Grimmer) ☐ G4 (Grimmer)

WIN: ☐ G1 (Khalil) ☐ G2 (Kusel) ☐ G3 (Kusel)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 10	24	Java-Programm inkl JavaDoc	Java-Programm inkl JavaDoc	☐	
Übung 11	24	Java-Programm inkl JavaDoc	Java-Programm inkl JavaDoc	☐	

MicroDraw

MicroDraw ist eine einfache GUI-Applikation mit der man Graphiken aus primitiven Shapes erstellen kann. Abbildung 1 zeigt die Applikation.

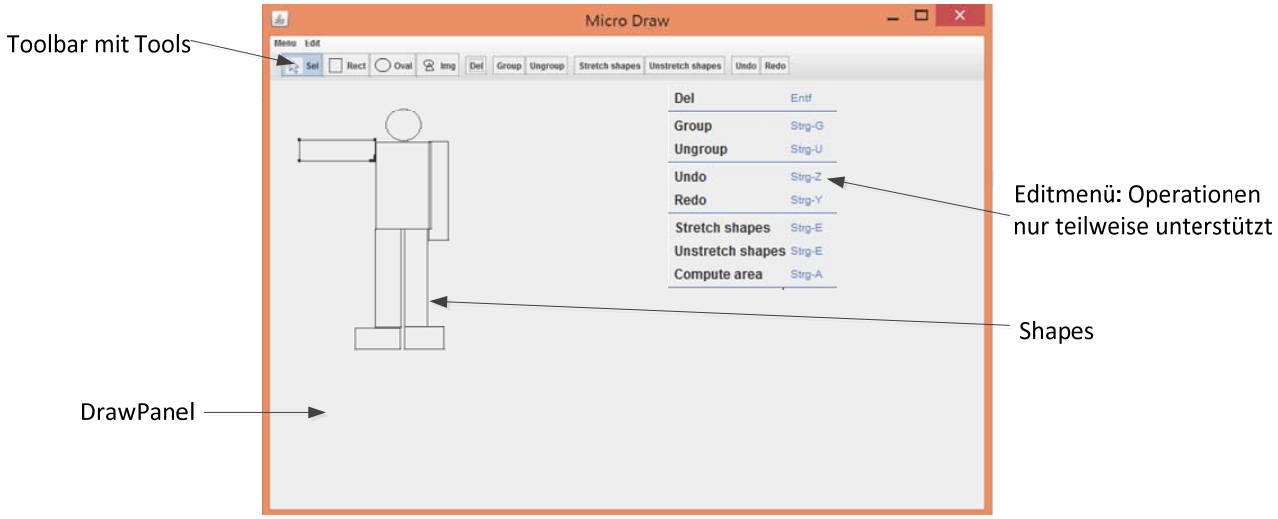

Abbildung 1: Screeshot von MicroDraw

Die Applikation hat eine Toolbar mit Tools zum Anfügen von Rechtecken und Ovalen und ein Selektionstools. Mit dem Selektionstool kann man Elemente selektieren, verschieben und vergrößern. Des Weiteren kann man Gruppen von selektierten Objekten bilden (Menüeintrag „Group“).

Die Implementierung der Applikation hat eine Architektur wie in Abbildung 2 gezeigt.

- Die angezeigten graphischen Objekte sind im Package `mdraw.shapes` implementiert. Sie sind alle von Interface `Shape` abgeleitet, welches Methoden für Zugriff auf und Setzen von Position und Größe, Zeichnen und Füllen, Selektion und Erzeugen einer Kopie definiert. Es gibt konkrete `Shape`-Klassen `Rect`, `Oval` und `Group`. `Groups` sind nach dem Composite-Pattern gebildet.
- Klasse `ShapeModel` zusammen mit Listeners und Event-Objekten aus dem Package `mdraw.model` implementiert ein Modell nach dem MVC-Prinzip. `ShapeModel` verwaltet eine Reihe von Shapes und eine Menge von aktuell selektierten Shapes. Es benachrichtigt Änderungen in der Menge der Shapes mit `shapeChanged` Ereignissen und Änderungen bei der Selektion mit `shapeSelection` Ereignissen.

- `ShapeApp` und `ShapePanel` implementieren die graphische Anzeige. Sie horchen auf `shapeChanged` und Änderungen `shapeSelection` Ereignisse des Modells. In `ShapeApp` gibt es eine Reihe von `Action`-Implementierungen, die die Reaktionen auf die Menüoperationen realisieren.
- Wesentlich bei der Applikation ist die Behandlung der Mouse-Ereignisse im `ShapePanel`. Die Reaktion auf die Mouse-Ereignisse ist abhängig vom im Toolbar ausgewählten Tool. Das `ShapePanel` leitet daher alle Mouse-Ereignisse an das in der Toolbar ausgewählte Tool weiter. Für die Implementierung von Tools gibt es eine abstrakte Basisklasse `Tool` (im Package `mdraw.tools`). `Tool` leitet von `AbstractAction` ab (kann also als `Action` verwendet werden) und implementiert `MouseListener` und `MouseMotionListener`. Zusätzlich definiert `Tool` auch eine Methode `paintToolFeedBack(Graphics g)`, die bei `paintComponent` vom `ShapePanel` aufgerufen wird, um eine visuelles Feedback für die Operation beim Tool zu geben (z.B. beim Verschieben von Shapes). Die Verwaltung der Tools erfolgt in der `ToolPalette`, die direkt als Toolbar verwendet wird (erweitert `JToolBar`). Man kann der `ToolPalette` Tools hinzufügen, die dann im Toolbar der Applikation erscheinen.

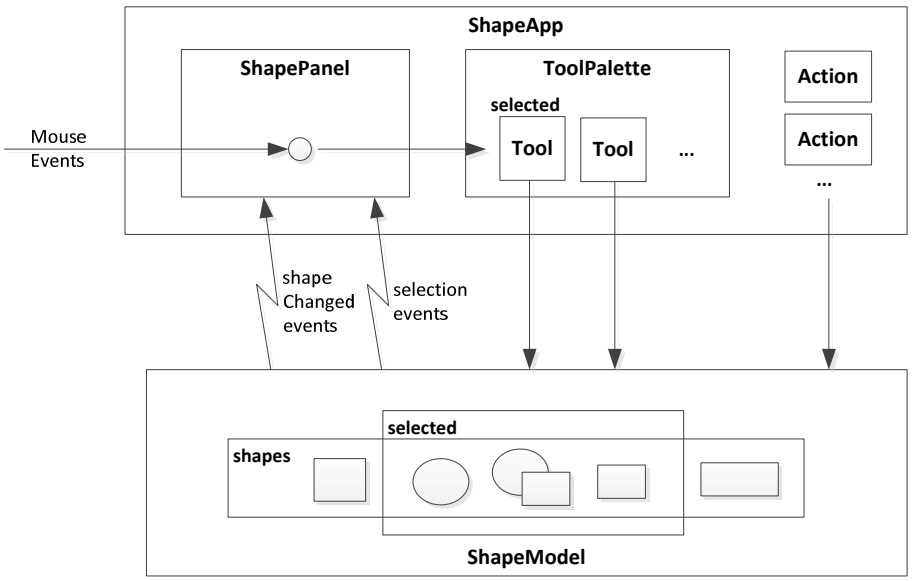

Abbildung 2: Architektur

Sie bekommen die Implementierung dieser Applikation als Download zur Verfügung gestellt. In Directory *doc* befindet sich eine JavaDoc-Dokumentation aller Klassen der Implementierung.

UE10: Erweiterung von MicroDraw mit ShapeAdapter und ShapeVisitor (24 Punkte)

Der Projektleiter möchte nun, dass Sie die Applikation im Funktionsumfang erweitern. Unser Projektleiter möchte, dass Sie dabei Design Patterns anwenden und formuliert die Aufgabe folgend:

Adaptor: Es sollen nun nicht nur Rechtecke und Ovale als elementare Shapes verwendet werden können, sondern auch weitere mit eventuell unterschiedlicher Schnittstelle. Dies soll nach dem Prinzip des Adapter-Pattern realisiert werden. Bauen Sie entsprechende Strukturen in das Klassensystem von Shapes ein.

Konkret soll nun Images über das Adapter-Pattern als Shapes verwendet werden können. Implementieren Sie dazu eine `ImageAdapter` Klasse. Implementieren Sie ein Tool und erweitern Sie die Toolbar für die Erzeugung von `ImageShapes`. Verwenden Sie einen `JFileChooser`, um eine Image-Datei für die Erzeugung des Images auszuwählen.

Hinweis: Folgende Anweisungen zeigen die Auswahl einer Image-Datei und das Erzeugen eines Image.

```
JFileChooser chooser = new JFileChooser(".");
int code = chooser.showOpenDialog(null);
if (code == 0) {
       File imgFile = chooser.getSelectedFile();
       Image img = javax.imageio.ImageIO.read(imgFile);
       …
}
```

Visitor: Erweitern Sie das System mit einem Visitor-Pattern. Das Visitor-Interface soll einen generischen Rückgabewert haben und folgend definiert sein:

```
public interface ShapeVisitor<T> {
   public T visitRect(Rect r);
   public T visitOval(Oval o);
   public T visitGroup(Group g);
   public T visitAdapter(ShapeAdapter a);
}
```

Implementieren Sie mit dem Visitor folgende Operationen:

- Einen `ComputeAreaVisitor`, der die Summe der Flächeninhalte aller Shapes berechnet. Dabei sollen für Gruppen die Flächeninhalte aller Elemente addiert werden, ohne auf Überlappungen Rücksicht zu nehmen. Der Flächeninhalt von Ovalen berechnet sich nach der Formel

  $$A = \pi ab$$

  wobei a und b die Halbachsen sind. Der berechnete Flächeninhalt soll auf der Standardausgabe ausgegeben werden.
- Einen `StretchVisitor`, der die Shapes folgend in der Größe verändert:
  - Rechtecke werden in der Höhe verdopplet
  - Oval werden in der Breite verdoppelt
  - Gruppen werden in beide Richtungen verdoppelt
  - Adapters bleiben unverändert.
- Einen `UnstretchVisitor`, der die inverse Operation zum `StretchVisitor` ausführt.

Führen Sie für die Operationen entsprechende Menüeinträge ein.

Diese Implementierung ist mit JavaDoc-Kommentaren zu dokumentieren!!

UE11: Erweiterung von MicroDraw mit Undo/Redo (24 Punkte)

In Übung 11 soll nun die Applikation so erweitert werden, dass Undo/Redo von Operationen ermöglicht werden. Implementieren Sie nach dem Command-Pattern Undo/Redo aller Operationen, die die Shapes im `ShapeModel` verändern.

Für Undo/Redo gibt es in der Klasse `DrawApp` bereits die entsprechenden Menüoperationen und die `Action`-Objekte `undoAction` und `redoAction`.

Ändern Sie alle Codestellen, die Veränderungen beim Modell bewirken.

Diese Implementierung ist mit JavaDoc-Kommentaren zu dokumentieren!!