Wiederholung Grafik und OO

Auf Wunsch der Schüler sollte ein Projekt – Vorschlag zu diesem Bereich im Unterricht bearbeitet werden. Der zu bearbeitende konkrete Vorschlag ist, die OO Grafik zu einem einfachen Brettspiel zu realisieren. Dieses Beispiel ist ein Dame – Spiel.

Textliche Analyse:

Dame wird auf einem Spielfeld mit 8 mal 8 quadratischen Feldern gespielt. Die Spielsteine sind kreisförmige Scheiben, in der Aufsicht also Kreise.
Die beiden Spieler besitzen unterschiedlich gefärbte Steine, die sie abwechselnd nacheinander auf das Spielfeld setzen.
Ein Stein kann nach dem Ende des Setzens bewegt werden und zwar ...
Dabei kann ein gegnerischer Stein geschlagen werden, wenn ...
Ein Spieler erhält eine Dame, also einen anders gekennzeichneten Spielstein, wenn ...
Eine Dame kann ...

Identifikation von Objekten

In der Reihenfolge des Auftretens im o.a. Text:

- Spielfeld
- Feld
- Spielsteine
- Kreise
- Spieler
- Dame

Identifikation von Methoden

In der Reihenfolge des Auftretens im o.a. Text:

- abwechselnd versteckt das Verb abwechseln
- setzen
- bewegen
- schlagen
- Dame erhalten

Hinter dem nicht ausgeführten Text „Eine Dame kann ...“ verstecken sich weitere oder spezielle Methoden von Damen.

Identifikation von Attributen und Beziehungen

Auch Attribute und Beziehungen lassen sich im o.a. Text schon identifizieren:
Aus dem Textabschnitt „*mit 8 mal 8 quadratischen Feldern*“ können wir einmal herauslesen, dass es eine Nutzer – Beziehung von Spielfeld zu Feld gibt. es handelt sich offensichtlich um eine Komposition, da „besteht aus“ zu formulieren ist, was eine Aggregation bedeutet (Teil – Ganzes Beziehung) und außerdem Felder für sich keine Bedeutung haben.
Ein Spieler hat Steine, hier sicher kein „besteht aus“, also eine Assoziation. Da diese Steine auf das Spielfeld gesetzt werden, besteht also auch eine Assoziation zwischen dem Spielfeld und den Steinen.

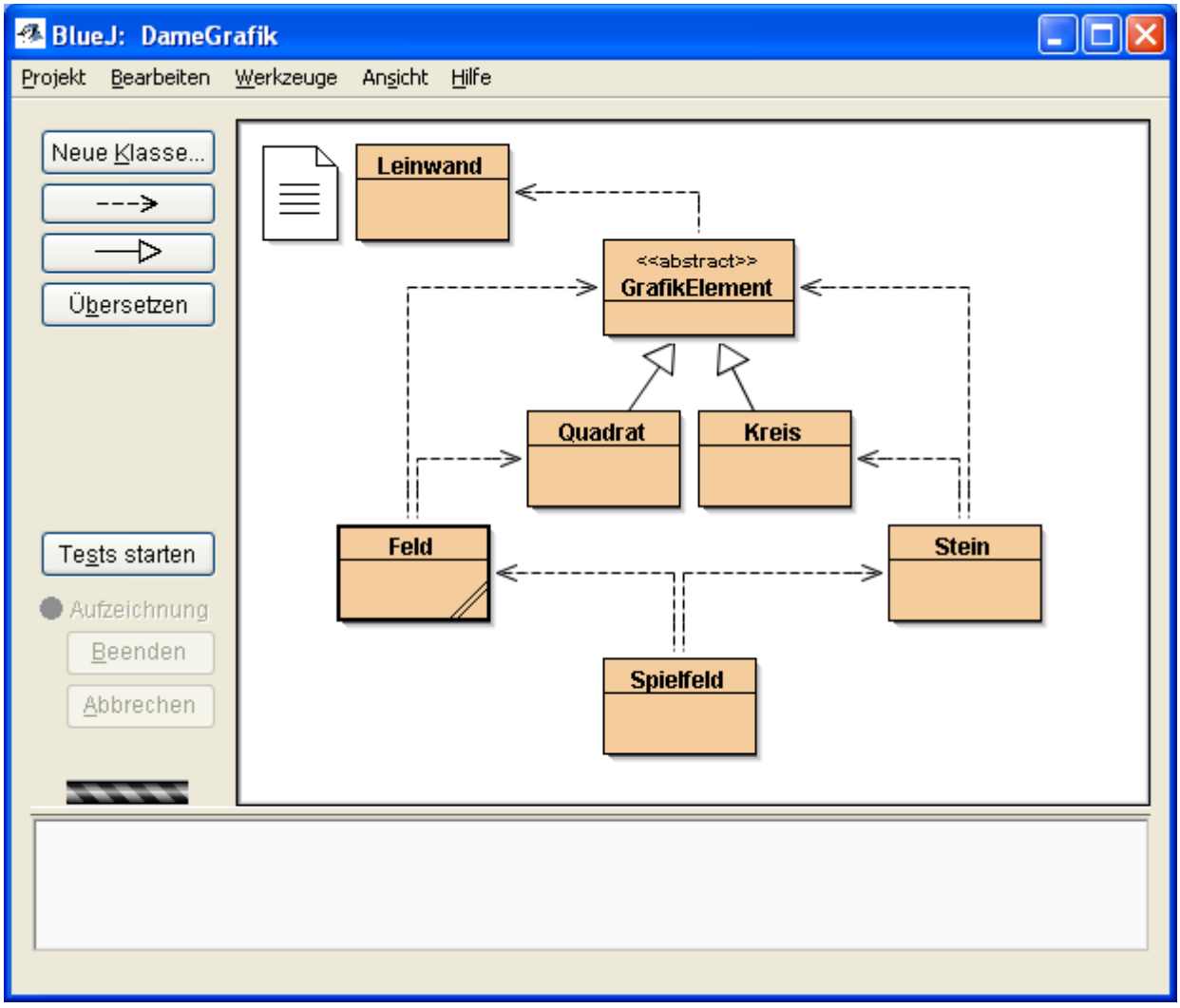
UML – Diagramme und Ergänzungen

Da dieses Projekt im Gegensatz zu den von uns realisierten Möbelprojekten nicht statisch

ist, wären auch Modellierungen mit Hilfe von Sequenzdiagrammen angemessen. Weiterhin interessant ist die Untersuchung der möglichen Zustände des Systems und seine möglichen Zustandsänderungen, womit wir beim bereich der Automatentheorie sind, die wir aus den behandelten Inhalten leider ausgeklammert haben.

Klassendiagramm

Ein Klassendiagramm mit BlueJ könnte so aussehen:



Aufgabe:

Untersuchen Sie gewählte (Teil-) Modellierung und machen Sie Vorschläge zur Ergänzung oder Veränderung[1].

1 Abschnitte aus dem Programmcode auf den folgenden Seiten

Abschnitte aus dem Programmcode

Aus der Klasse Leinwand

```
  /*************************************************************
   * Interne Klasse ShapeMitFarbe - Da die Klasse Shape des JDK nicht auch
   * eine Farbe mitverwalten kann, muss mit dieser Klasse die Verknüpfung
   * modelliert werden.
   * graphic.fill() wurde in der bisher verwendeten Variante durch
...* graphic.draw() ersetzt, hier wird über ein zusätzlichs Attribut
   * gefuellt gesteuert, ob voll oder nur Umriss gezeichnet werden soll.
   */
  private class ShapeMitFarbe
  {
    private Shape shape;
    private String farbe;
    private boolean gefuellt;

    public ShapeMitFarbe(Shape shape, String farbe, boolean gefuellt)
    {
      this.shape = shape;
      this.farbe = farbe;
      this.gefuellt = gefuellt;
    }

    public void draw(Graphics2D graphic)
    {
      setzeZeichenfarbe(farbe);
      if (gefuellt) graphic.fill(shape); else graphic.draw(shape);
    }
  }
```

Die Klasse GrafikElement entspricht in wesentlichen Teilen unserer alten Klasse Moebel. Daher ist hier nur der Konstruktor angegeben:

```
    protected GrafikElement(int x, int y, String farbe, boolean gefuellt) {
        xPosition = x;
        yPosition = y;
        this.farbe = farbe;
        this.gefuellt = gefuellt;
        istSichtbar = false;
    }
```

Der Text der Klasse Quadrat steht hier beispielhaft auch für Kreis

```
import java.awt.Shape;
import java.awt.geom.Rectangle2D;

public class Quadrat extends GrafikElement{
    private int kante;

    // Konstruktor für Objekte der Klasse Quadrat
    public Quadrat(int x, int y, int kante, String farbe, boolean gefuellt)
    {
        super(x, y, farbe, gefuellt);
        this.kante=kante;
    }

    protected Shape gibAktuelleFigur() {
        Shape quadrat=new Rectangle2D.Double(0, 0, kante, kante);
        return transformiere(quadrat);
    }
}
```

Ausschnitte aus der Klasse Spielfeld:

```
import java.util.ArrayList;

/**
 * Die Klasse Spielfeld.
 */
public class Spielfeld
{
    private ArrayList<Feld> felder;
    private ArrayList<Stein> steine;

    /**
     * Konstruktor für Objekte der Klasse Spielfeld
     */
    public Spielfeld()
    {
        super();
        felder=new ArrayList<Feld>();
        steine=new ArrayList<Stein>();
        for (int i=0; i<8; i++) {
            for (int j=0; j<8; j++) {
                Feld feld=new Feld(i,j, 40);
                felder.add(feld);
                feld.zeige();
            }
        }
    }

    /**
     * setzt einen Stein auf das Feld.
     */
    public void setzeStein(int xPos, int yPos) {
        if (xPos<1) return;
        if (xPos>8) return;
        if (yPos<1) return;
        if (yPos>8) return;
        Stein stein=new Stein(xPos-1, yPos-1, 40);
        steine.add(stein);
        stein.zeige();
       }
}
```