Praktikum Computergrafik, Blatt 4

* Aufgabe 1 (Dreiecksoperationen)

Diese Aufgabe muss nicht abgegeben werden (aber die weiter unten stehende schon!). Sie sollten Sie jedoch bearbeiten, um Flächenberechnung, Orientierung für ein Dreieck sowie den „Punkt im Dreieck"-Test zu verinnerlichen.

Lesen Sie das Rahmenprogramm `Triangle.java` aus Moodle ☜ durch. Es speichert die Koordinaten von vier Mausklicks in der Liste `points`. Nach drei Mausklicks wird die Variable `setTriangle` auf `false` gesetzt, nach vier Mausklicks die Variable `setPoints`.

Fügen Sie Code in `paintComponent` ein, der

- die Punkte aus `points` als kleine Quadrate der Seitenlänge 5 Pixel zeichnet (mit der Methode `graphics.drawRect(...)`),
- ein Dreieck durch die ersten drei Punkte zeichnet (`graphics.drawLine(...)`), sobald das Attribut `setTriangle` den Wert `false` hat,
- weiter, sobald `setTriangle` den Wert `false` hat, die Fläche des Dreiecks errechnet[1] und im Fenster ausgibt (`graphics.drawString(...)`),
- ermittelt, ob der vierte Punkt inner- oder außerhalb des Dreiecks liegt, sobald `setPoints` den Wert `false` hat. Das Ergebnis des Tests soll im Fenster ausgegeben werden.

** Aufgabe 2 (Bézier-Kurven)

In Moodle ☜ finden Sie die Datei `bezier.zip` mit den Rahmenprogrammen:

`BezierPanel.java`: `JPanel`-Klasse mit `main`-Routine zur Darstellung , muss nicht editiert werden.

`Bezier.java`: Das ist die Klasse, in der Sie den Algorithmus zum Zeichnen von Bézier-Kurven implementieren müssen.

`BezierTest.java`: JUnit4-Test, muss nicht editiert werden. Identisch mit dem vom APA-Server ausgeführten Test.

`Point.java`: Einfache Klasse, die einen 2D-Punkt repräsentiert.

`BezierPanel` erlaubt es, durch Mausklicks eine beliebige Zahl von Kontrollpunkten $P_0, \dots, P_n$ zu wählen. Dabei werden die Punkte als kleine Quadrate gezeichnet sowie der verbindende Linienzug (etwas fälschlich als Kontrollpolygon bezeichnet).

Durch erneutes Klicken auf einen bereits gewählten Punkt wird die Auswahl von Kontrollpunkten beendet und die Berechnung der Bézier-Kurve gestartet. Hier kommen Sie ins Spiel.

a) Implementieren Sie in der Klasse `Bezier` den Konstruktor

[1] Plausibilitätstest: Wenn man bei einem 500×500-Fenster in die Nähe dreier Ecken klickt, welche Fläche ist zu erwarten?

```
Bezier(List<Point> points, double h)
```

Die Bedeutung der Parameter ist in den JavaDoc-Kommentaren angegeben. Speichern Sie die Parameter in Attribute. Sie können, falls Sie die Notwendigkeit dafür sehen, im weiteren Verlauf noch weitere Attribute hier initialisieren.

b) Implementieren Sie dann in der Klasse `Bezier` die Methode

```
Point casteljau(double t)
```

die den Casteljau-Algorithmus für einen Kurvenparameter t durchläuft, also den zu dem Parameter t gehörigen Punkt $\vec{x}(t)$ der Bezier-Kurve liefert.

Sie können hierzu z.B. ein zweidimensionales Array von `Points` verwenden.

Hilfreich ist der Konstruktor

```
Point(double alpha, Point p1, double beta, Point p2) {
```

der Klasse `Point`. Mit diesem können Sie zwei Punkte gewichtet addieren, so wie es im Casteljau-Algorithmus benötigt wird[2].

c) Implementieren Sie dann in der Klasse `Bezier` die Methode

```
void render(Graphics graphics)
```

die die Bézier-Kurve zeichnet. Sie berechnet im Parameterintervall $[0;1]$ für $t = h, 2h, 3h, \ldots$ mit der Methode `casteljau` Punkte $\vec{x}(t)$ der Bezier-Kurve und verbindet sie mit der `drawLine`-Methode des `graphics`-Objekts.

Vergessen Sie dabei nicht, den ersten Kontrollpunkt mit dem ersten berechneten Punkt zu verbinden und den letzten berechneten Punkt mit dem letzten Kontrollpunkt.

Nach dem Zeichnen der Bézier-Kurve können Sie Kontrollpunkte mit der Maus über *drag and drop* verschieben. Die Kurve folgt dem geänderten Kontrollpolygon.

[2]Sonderlich effizient ist das natürlich nicht, weil man viele Objekte erzeugt und der GC immer beschäftigt wird. Eine bessere Lösung benutzt ein einmalig alloziertes mehrdimensionales `double`-Array.

Fragen zur Übung (prüfungsrelevant)

1.) Seien (`x0, y0`) und (`x1, y1`) Koordinaten von Pixeln. Wie fragen Sie in C oder Java ab, ob der (Pixel-) Abstand zwischen den Pixeln maximal `r` ist?

2.) Java2D verwendet als Ursprung die linke obere Ecke des Fensters. Muss das bei der Flächenberechnung oder beim Test, ob ein Punkt im Dreieck liegt, berücksichtigt werden?

3.) Wie viele Punkte P_k^i müssen für eine Bézierkurve vom Grad n bei einer Schrittweite h berechnet werden? In Landauscher O-Notation?

4.) Gegeben sei eine Routine, die zu einem Kurvenparameter t die Koordinaten des zugehörigen Punkts auf der Bézierkurve errechnet:

$(x, y) \leftarrow \text{bezier}(t)$

sowie eine Routine

drawLine(x0, y0, x1, y1)

zum Linienzeichnen.

Geben Sie einen Pseudocode an, der die Bézierkurve mit einer Schrittweite von h auswertet und zeichnet.

5.) Welchen Kurvenverlauf hat eine Bézierkurve 4. Grades qualitativ für die folgenden Stützpolygone? Bitte einzeichnen.