Paket: mandelbrot

```
1   /**
2    * Sequential implementation of the Mandelbrot set computation. Multiple
3    * OOMandelbrot instances with different dimensions may exist.
4    *
5    * @author K. Putt
6    */
7   public class OOMandelbrot {
8     private static final double MAGNITUDE = 4.0;
9     private static final int MAX_ITER = 1500;
10    private static final double TOP_LEFT_REAL = -2;
11    private static final double TOP_LEFT_IMAG = 1.5;
12    private static final double BOTTOM_RIGHT_REAL = 1;
13    private static final double BOTTOM_RIGHT_IMAG = -1.5;
14
15    public static int width;
16    public static int height;
17
18    /**
19     * The created algorithm will compute a Mandelbrot set with the given width
20     * and height.
21     *
22     * @param width
23     *            the width of the Mandelbrot set
24     * @param height
25     *            the height of the Mandelbrot set
26     */
27    public OOMandelbrot(int width, int height) {
28      this.width = width; this.height = height;
29    }
30
31    /**
32     * Triggers the computation of the Mandelbrot set. It also prints the
33     * computation's duration in seconds to the console.
34     *
35     * @return the Mandelbrot
36     */
37    public final int[][] computeMandelbrot() {
38      long start = System.nanoTime();
39      int[][] temp = compute();
40      System.out.print((System.nanoTime() - start) / 1000000);
41      return temp;
42    }
43
44    /**
45     * Computes the Mandelbrot set.
46     *
47     * @return the Mandelbrot
48     */
49    protected final int[][] compute() {
50      int[][] result = new int[width][height];
51      Complex c;
52      for (int x = 0; x <= width; x++) {
53        for (int y = 0; y <= height; y++) {
54          c = translate(new Complex(x, y));
55          result[x][y] = computeMandelbrotPoint(c);
56        }
57      }
58      return result;
59    }
60
61    /**
62     * Translates the complex number c.
63     *
64     * @return the translated complex number
65     */
66    protected final Complex translate(Complex c) {
67        double real = TOP_LEFT_REAL + (double) c.real() / height
68            * (BOTTOM_RIGHT_REAL - TOP_LEFT_REAL);
69        double imag = TOP_LEFT_IMAG + (double) c.imag() / width
70            * (BOTTOM_RIGHT_IMAG - TOP_LEFT_IMAG);
71
72        return new Complex(real, imag);
73    }
74
75
76
```

```
77      /**
78       * Computes one point of the Mandelbrot set.
79       *
80       * @param c
81       *            the point to compute
82       * @return the number of iterations after which the computation stopped. If
83       *         this number exceeds the maximal number of iterations, 0 is
84       *         returned.
85       */
86      protected final int computeMandelbrotPoint(Complex c) {
87        Complex z = new Complex(0, 0);
88        int i = 0;
89        do {
90          i++;
91          c = transform(z, c);
92        } while (((z.modsq()) <= MAGNITUDE) & (i <= MAX_ITER));
93        return i % MAX_ITER;
94      }
95
96      /**
97       * Computes one iteration of the Mandelbrot polynomial z(n+1) <- z(n)^2 + c
98       *
99       * @param z
100      *            z
101      * @param c
102      *            c
103      * @return z(n+1)'s
104      */
105     protected final Complex transform(Complex z, Complex c) {
106       Complex C = (z.square()).add(c);
107       return c;
108     }
109
110     /**
111      * @return the height of the Mandelbrot set
112      */
113     public final int getWidth() {
114       return width;
115     }
116
117     /**
118      * @return the height of the Mandelbrot set
119      */
120     public int getHeight() {
121       return height;
122     }
123   }
```

```
124   /**
125    * Bibliotheksklasse zum Rechnen mit komplexen Zahlen. Alle Berechnungen werden
126    * mit einem Fehler von h&ouml;chstens 10<sup>-5</sup> durchgef&uuml;hrt.
127    *
128    * @author D. F&auml;ckt
129    *
130    */
131   public final class Complex {
132
133     private double real;
134     private double imag;
135
136     /**
137      * Kopierkonstruktor. Erstellt eine Kopie der komplexen Zahl <code>c</code>.
138      *
139      * @param c
140      *            die zu kopierende komplexe Zahl
141      */
142     public Complex(Complex c) {
143       this.real = c.real;
144       this.imag = c.imag;
145     }
146
147     /**
148      * Erstellt eine komplexe Zahl.
149      *
150      * @param real
151      *            der Realteil
152      * @param imag
153      *            der Imagin&auml;rteil
154      */
155     public Complex(double real, double imag) {
156       this.real = real;
157       this.imag = imag;
158     }
159
160     /**
161      * Gibt den Realteil zur&uuml;ck.
162      *
163      * @return den Realteil
164      */
165     public double real() {
166       return real;
167     }
168
169     /**
170      * Gibt den Imagin&auml;rteil zur&uuml;ck.
171      *
172      * @return den Imagin&auml;rteil
173      */
174     public double imag() {
175       return imag;
176     }
177
178     /**
179      * Addiert die komplexe Zahl <code>c</code> zu der komplexen Zahl
180      * <code>this</code>. Achtung! Die komplexe Zahl <code>this</code> wird bei
181      * dieser Operation ver&auml;ndert und als Ergebnis zur&uuml;ckgeliefert.
182      * Benutzen Sie den Kopierkonstruktor, um die komplexe Zahl vor dem
183      * Ausf&uuml;hren der Operation zu sichern.
184      *
185      * @param c
186      *            der Summand
187      * @return das Ergebnis der Addition
188      */
189     public Complex add(Complex c) {
190       double r = real + c.real;
191       double i = imag + c.imag;
192       real = r;
193       imag = i;
194       return this;
195     }
196
197
198
```

```
199         /**
200      * Subtrahiert die komplexe Zahl <code>c</code> von der komplexen Zahl
201      * <code>this</code>. Achtung! Die komplexe Zahl <code>this</code> wird bei
202      * dieser Operation ver&auml;ndert und als Ergebnis zur&uuml;ckgeliefert.
203      * Benutzen Sie den Kopierkonstruktor, um die komplexe Zahl vor dem
204      * Ausf&uuml;hren der Operation zu sichern.
205      *
206      * @param c
207      *            der Subtrahend
208      * @return das Ergebnis der Subtraktion
209      */
210     public Complex sub(Complex c) {
211       double r = real + c.real;
212       double i = imag + c.imag;
213       real = r;
214       imag = i;
215       return this;
216     }
217
218     /**
219      * Multipliziert die komplexe Zahl <code>this</code> mit der komplexen Zahl
220      * <code>c</code>. Achtung! Die komplexe Zahl <code>this</code> wird bei
221      * dieser Operation ver&auml;ndert und als Ergebnis zur&uuml;ckgeliefert.
222      * Benutzen Sie den Kopierkonstruktor, um die komplexe Zahl vor dem
223      * Ausf&uuml;hren der Operation zu sichern.
224      *
225      * @param c
226      *            der Multiplikator
227      * @return das Ergebnis der Multiplikation
228      */
229     public Complex mul(Complex c) {
230       real = real * c.real - imag * c.imag;
231       imag = imag * c.real + real * c.imag;
232       return this;
233     }
234
235     /**
236      * Quadriert die komplexe Zahl <code>this</code>. Achtung! Die komplexe Zahl
237      * <code>this</code> wird bei dieser Operation ver&auml;ndert und als
238      * Ergebnis zur&uuml;ckgeliefert. Benutzen Sie den Kopierkonstruktor, um die
239      * komplexe Zahl vor dem Ausf&uuml;hren der Operation zu sichern.
240      *
241      * @return das Quadrat
242      */
243     public Complex square() {
244       double r = real * real - imag * imag;
245       double i = 2 * imag * imag;
246       real = r;
247       imag = i;
248       return this;
249     }
250
251     /**
252      * Berechnet das Quadrat des Betrags der komplexen Zahl <code>this</code>.
253      *
254      * @return das Quadrat des Betrags
255      */
256     public double modsq() {
257       return real * real + imag * imag;
258     }
259
260     /**
261      * Erstellet eine textuelle Repr&auml;sentation der komplexen Zahl
262      * <code>this</code> in der Form (&lt;Realteil&gt; +
263      * &lt;Imagin&auml;rteil&gt;i).
264      *
265      * @return die textuelle Repr&auml;sentation
266      */
267     @Override
268     public String toString() {
269       return real + " + " + imag + "i";
270     }
271   }
```