```java
/**
 * Klasse zur Repr&auml;sentation eines gek&uuml;rzten Dezimalbruchs.
 */
public class Bruch {

    /**
     *
     */
    private int zaehler, nenner;


    /**
     * Erzeugt einen neuen Bruch mit dem Zaehler 1 und dem Nenner 1.
     * Der Bruch representiert somit den Wert 1.
     */
    public Bruch() {
        this.zaehler = this.nenner = 1;
    }

    /**
     * Erzeugt einen neuen Bruch mit dem Zaehler zaehler und dem Nenner 1.
     * Der Bruch representiert den Wert des Zaehlers.
     * @param zaehler
     */
    public Bruch(int zaehler) {
        this.zaehler = zaehler;
        this.nenner = 1;
    }

    /**
     * Erzeugt einen neuen Bruch mit dem Zaehler zaehler und dem Nenner nenner.
     * @param zaehler
     * @param nenner
     */
    public Bruch(int zaehler, int nenner) {
        if(nenner == 0)
            throw new IllegalArgumentException("Nenner muss von Null verschieden sein.");
        this.zaehler = zaehler;
        setNenner(nenner); // Kuertzt den Bruch direkt.
    }

    /**
     * Addiert den Bruch b2 zum Bruch.
     * @param b2 Anderer Summand
     * @return Referenz auf einen Bruch welcher die Summe enthaelt.
     */
    public Bruch add(Bruch b2){
        int n3 = this.nenner*b2.getNenner();
        int z3 = this.zaehler*b2.getNenner() + b2.getZaehler() * this.nenner;
        return new Bruch(z3, n3);
```

```
51      }
52      /**
53       * Substituiert b2 vom Bruch.
54       * @param b2 der Substituent
55       * @return Differrenz Bruch-b2.
56       */
57      public Bruch sub(Bruch b2) {
58          Bruch b3 = new Bruch(b2.zaehler, b2.nenner * -1);
59          return add(b3);
60      }
61      /**
62       * Berechent das Produkt des Bruches mit einem Anderen Bruch b2.
63       * @param b2 anderer Bruch.
64       * @return Referenz auf ein neuen Bruch welcher das Produkt enthaelt.
65       */
66      public Bruch mul(Bruch b2) {
67          return new Bruch(
68                  this.zaehler * b2.zaehler,
69                  this.nenner * b2.nenner
70                  );
71      }
72
73      /**
74       * Berechent den Quotienten des Bruches mit einem Anderen Bruch b2.
75       * @param b2 Bruch durch den geteilt werden soll.
76       * @return Referenz auf ein neuen Bruch welcher den Quotienten
    enthaelt.
77       */
78      public Bruch div(Bruch b2) {
79          return mul(b2.getKehrWert());
80      }
81
82      /**
83       * Berechent den Kehrwert des Bruches.
84       * @return Referenz auf ein neuen Bruch welcher den Kehrwert des
    Bruches enthaelt.
85       */
86      public Bruch getKehrWert() {
87          return new Bruch(nenner, zaehler);
88      }
89
90      /**
91       * Berechnet den groeßten gemeinsamen Teiler von a, b.
92       * @param a Zahl 1
93       * @param b Zahl 2
94       * @return ggt
95       */
96      int ggt(int a, int b) {
97          return b == 0 ? a : ggt(b, a % b);
98      }
99
100     /**
101      * Kuertzt den Bruch
```

```
102      */
103     void kuerze() {
104         int ggt =  ggt(nenner, zaehler);
105         nenner  /= ggt;
106         zaehler /= ggt;
107     }
108 
109     public int getZaehler() {
110         return zaehler;
111     }
112 
113     public void setZaehler(int zaehler) {
114         this.zaehler = zaehler;
115     }
116 
117     public int getNenner() {
118         return nenner;
119     }
120 
121     /**
122      * Setzt den Nenner des Bruchs und k&uuml;rzt dann ggfs.
123      * @param Nenner Der neue Nenner des Bruchs.
124      */
125     public void setNenner(int nenner) {
126         // Beispiel zur Behandlung eines ungueltigen Parameters.
127         if (nenner == 0) {
128             throw new IllegalArgumentException("Nenner muss von Null
    verschieden sein.");
129         }
130         this.nenner = nenner;
131         kuerze();
132     }
133 
134     @Override
135     public boolean equals(Object arg0) {
136 //      System.out.println(zaehler + " / " + nenner);
137         if (!(arg0 instanceof Bruch) ){
138             return false;
139         }else if(nenner != ((Bruch) arg0).getNenner()) {
140             return false;
141         }else if(zaehler != ((Bruch) arg0).getZaehler()) {
142             return false;
143         }
144         return true;
145     }
146 
147 
148 
149 }
150 
```