```java
import components.naturalnumber.NaturalNumber;

/**
 * {@code NaturalNumber} represented as a {@code String} with implementations of
 * primary methods.
 *
 * @convention <pre>
 * [all characters of $this.rep are '0' through '9']  and
 * [$this.rep does not start with '0']
 * </pre>
 * @correspondence <pre>
 * this = [if $this.rep = "" then 0
 *         else the decimal number whose ordinary depiction is $this.rep]
 * </pre>
 *
 * @author Shyam Sai Bethina and Yihone Chu
 *
 */
public class NaturalNumber3 extends NaturalNumberSecondary {

    /*
     * Private members
    --------------------------------------------------------
     */

    /**
     * Representation of {@code this}.
     */
    private String rep;

    /**
     * Creator of initial representation.
     */
    private void createNewRep() {

        /*
         * rep becomes an empty string
         */
        this.rep = "";
    }

    /*
```

```
43      * Constructors
   ------------------------------------------------------------
44      */
45
46     /**
47      * No-argument constructor.
48      */
49     public NaturalNumber3() {
50
51         this.createNewRep();
52     }
53
54     /**
55      * Constructor from {@code int}.
56      *
57      * @param i
58      *            {@code int} to initialize from
59      */
60     public NaturalNumber3(int i) {
61         assert i >= 0 : "Violation of: i >= 0";
62         this.createNewRep();
63         /*
64          * If the integer is 0, then the representation is an
   empty string,
65          * else, the representation is the string version of the
   integer.
66          */
67         if (i == 0) {
68             this.rep = "";
69         } else {
70             this.rep = Integer.toString(i);
71         }
72
73     }
74
75     /**
76      * Constructor from {@code String}.
77      *
78      * @param s
79      *            {@code String} to initialize from
80      */
81     public NaturalNumber3(String s) {
82         assert s != null : "Violation of: s is not null";
83         assert s.matches("0|[1-9]\\d*") : ""
```

```
 84                 + "Violation of: there exists n: NATURAL (s =
    TO_STRING(n))";
 85
 86         this.createNewRep();
 87         /*
 88          * If the string is equal to "0" or empty, then the
    representation is an
 89          * empty string, else, the representation is the input
    string.
 90          */
 91         if (s.equals("0") || s.equals("")) {
 92             this.rep = "";
 93         } else {
 94             this.rep = s;
 95         }
 96
 97     }
 98
 99     /**
100      * Constructor from {@code NaturalNumber}.
101      *
102      * @param n
103      *            {@code NaturalNumber} to initialize from
104      */
105     public NaturalNumber3(NaturalNumber n) {
106         assert n != null : "Violation of: n is not null";
107
108         this.createNewRep();
109
110         /*
111          * If the natural number is equal to zero, then the
    representation is an
112          * empty string, else, the representation is the string
    version of the
113          * natural number.
114          */
115         if (n.isZero()) {
116             this.rep = "";
117         } else {
118             this.rep = n.toString();
119         }
120
121     }
122
```

```
123     /*
124      * Standard methods
    -----------------------------------------------------------
125      */
126
127     @Override
128     public final NaturalNumber newInstance() {
129         try {
130             return this.getClass().getConstructor().newInstance();
131         } catch (ReflectiveOperationException e) {
132             throw new AssertionError(
133                     "Cannot construct object of type " +
    this.getClass());
134         }
135     }
136
137     @Override
138     public final void clear() {
139         /*
140          * Clears it by creating an empty representation.
141          */
142         this.createNewRep();
143     }
144
145     @Override
146     public final void transferFrom(NaturalNumber source) {
147         assert source != null : "Violation of: source is not
    null";
148         assert source != this : "Violation of: source is not
    this";
149         assert source instanceof NaturalNumber3 : ""
150                 + "Violation of: source is of dynamic type
    NaturalNumberExample";
151         /*
152          * This cast cannot fail since the assert above would have
    stopped
153          * execution in that case.
154          */
155         NaturalNumber3 localSource = (NaturalNumber3) source;
156         this.rep = localSource.rep;
157         localSource.createNewRep();
158     }
159
160     /*
```

```
161     * Kernel methods
    ------------------------------------------------------------
162     */
163
164    @Override
165    public final void multiplyBy10(int k) {
166        assert 0 <= k : "Violation of: 0 <= k";
167        assert k < RADIX : "Violation of: k < 10";
168        /*
169         * If the representation string is not empty or the input
    integer is not
170         * 0, then the representation string becomes the old
    representation plus
171         * k.
172         */
173        if (this.rep.length() != 0 || k != 0) {
174            this.rep = this.rep + k;
175        }
176    }
177
178    @Override
179    public final int divideBy10() {
180        /*
181         * The answer starts out as 0.
182         */
183        int answer = 0;
184        /*
185         * If the length of the representation is not 0, then the
    returned value
186         * is the last integer of the representation.
187         */
188        if (this.rep.length() != 0) {
189            answer = Integer.parseInt(
190
    Character.toString(this.rep.charAt(this.rep.length() - 1)));
191            /*
192             * The new representation is the old representation
    excluding the
193             * last digit.
194             */
195            this.rep = this.rep.substring(0, this.rep.length() -
    1);
196        }
197
```

```
198         return answer;
199     }
200 
201     @Override
202     public final boolean isZero() {
203 
204         /*
205          * If the representation is empty, then it returns true,
    else it returns
206          * false.
207          */
208         return this.rep.isEmpty();
209     }
210 
211 }
212 
```