```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 Micah Casey-Fusco & Andrew Wu
 *
 */
public class NaturalNumber3 extends NaturalNumberSecondary {

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

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

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

        this.rep = "";

    }

    /*
     * Constructors -----------------------------------------------------------
     */

    /**
     * No-argument constructor.
     */
    public NaturalNumber3() {

        this.createNewRep();

    }

    /**
     * Constructor from {@code int}.
     *
     * @param i
     *            {@code int} to initialize from
     */
```

```
59      public NaturalNumber3(int i) {
60          assert i >= 0 : "Violation of: i >= 0";
61
62          //check for empty rep case, otherwise set this.rep to string of i
63          if (i == 0) {
64              this.createNewRep();
65          } else {
66              this.rep = Integer.toString(i);
67          }
68
69      }
70
71      /**
72       * Constructor from {@code String}.
73       *
74       * @param s
75       *            {@code String} to initialize from
76       */
77      public NaturalNumber3(String s) {
78          assert s != null : "Violation of: s is not null";
79          assert s.matches("0|[1-9]\\d*") : ""
80                  + "Violation of: there exists n: NATURAL (s = TO_STRING(n))";
81
82          //check for empty rep case, otherwise set this.rep to s
83          if (s.equals("0")) {
84              this.rep = "";
85          } else {
86              this.rep = s;
87          }
88
89      }
90
91      /**
92       * Constructor from {@code NaturalNumber}.
93       *
94       * @param n
95       *            {@code NaturalNumber} to initialize from
96       */
97      public NaturalNumber3(NaturalNumber n) {
98          assert n != null : "Violation of: n is not null";
99
100         //check for empty rep case, otherwise set this.rep to s
101         if (n.isZero()) {
102             this.rep = "";
103         } else {
104             this.rep = n.toString();
105         }
106
107     }
108
109     /*
110      * Standard methods ---------------------------------------------------
111      */
112
113     @Override
114     public final NaturalNumber newInstance() {
115         try {
```

```
116             return this.getClass().getConstructor().newInstance();
117         } catch (ReflectiveOperationException e) {
118             throw new AssertionError(
119                     "Cannot construct object of type " + this.getClass());
120         }
121     }
122
123     @Override
124     public final void clear() {
125         this.createNewRep();
126     }
127
128     @Override
129     public final void transferFrom(NaturalNumber source) {
130         assert source != null : "Violation of: source is not null";
131         assert source != this : "Violation of: source is not this";
132         assert source instanceof NaturalNumber3 : ""
133                 + "Violation of: source is of dynamic type NaturalNumberExample";
134         /*
135          * This cast cannot fail since the assert above would have stopped
136          * execution in that case.
137          */
138         NaturalNumber3 localSource = (NaturalNumber3) source;
139         this.rep = localSource.rep;
140         localSource.createNewRep();
141     }
142
143     /*
144      * Kernel methods ------------------------------------------------------
145      */
146
147     @Override
148     public final void multiplyBy10(int k) {
149         assert 0 <= k : "Violation of: 0 <= k";
150         assert k < RADIX : "Violation of: k < 10";
151         // converts k to a string and concatenates to end(rightside) of this.rep
152         this.rep = this.rep + Integer.toString(k);
153
154     }
155
156     @Override
157     public final int divideBy10() {
158         int remainder = 0;
159         // if rep.length == 0; it is an empty string -> return 0
160         // v checks for non empty string
161         if (this.rep.length() > 0) {
162             // remainder = the last char in rep
163             remainder = Integer
164                     .parseInt(this.rep.substring(this.rep.length() - 1));
165             // rep updated, dropping the last char
166             this.rep = this.rep.substring(0, this.rep.length() - 1);
167         }
168         return remainder;
169     }
170
171     @Override
172     public final boolean isZero() {
```

```
173         // empty string == 0
174         return this.rep.isEmpty();
175     }
176
177 }
178
```