```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 Zhuoyang Li + Xinci Ma
 *
 */
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.
```

```
45      */
46     public NaturalNumber3() {
47
48         this.createNewRep();
49
50     }
51
52     /**
53      * Constructor from {@code int}.
54      *
55      * @param i
56      *            {@code int} to initialize from
57      */
58     public NaturalNumber3(int i) {
59         assert i >= 0 : "Violation of: i >= 0";
60
61         this.createNewRep();
62
63         if (i != 0) {
64             this.rep = Integer.toString(i);
65         }
66
67     }
68
69     /**
70      * Constructor from {@code String}.
71      *
72      * @param s
73      *            {@code String} to initialize from
74      */
75     public NaturalNumber3(String s) {
76         assert s != null : "Violation of: s is not null";
77         assert s.matches("0|[1-9]\\d*") : ""
78                 + "Violation of: there exists n: NATURAL (s =
   TO_STRING(n))";
79
80         this.createNewRep();
81         if (!s.equals("0")) {
82             this.rep = s;
83         }
84     }
85
86     /**
87      * Constructor from {@code NaturalNumber}.
88      *
89      * @param n
90      *            {@code NaturalNumber} to initialize from
```

```
 91      */
 92     public NaturalNumber3(NaturalNumber n) {
 93         assert n != null : "Violation of: n is not null";
 94
 95         this.createNewRep();
 96         if (!n.isZero()) {
 97             this.rep = n.toString();
 98         }
 99
100     }
101
102     /*
103      * Standard methods
    -----------------------------------------------------------
104      */
105
106     @Override
107     public final NaturalNumber newInstance() {
108         try {
109             return
    this.getClass().getConstructor().newInstance();
110         } catch (ReflectiveOperationException e) {
111             throw new AssertionError(
112                     "Cannot construct object of type " +
    this.getClass());
113         }
114     }
115
116     @Override
117     public final void clear() {
118         this.createNewRep();
119     }
120
121     @Override
122     public final void transferFrom(NaturalNumber source) {
123         assert source != null : "Violation of: source is not
    null";
124         assert source != this : "Violation of: source is not
    this";
125         assert source instanceof NaturalNumber3 : ""
126                 + "Violation of: source is of dynamic type
    NaturalNumberExample";
127         /*
128          * This cast cannot fail since the assert above would
    have stopped
129          * execution in that case.
130          */
```

```
131         NaturalNumber3 localSource = (NaturalNumber3) source;
132         this.rep = localSource.rep;
133         localSource.createNewRep();
134     }
135
136     /*
137      * Kernel methods
    ------------------------------------------------------------
138      */
139
140     @Override
141     public final void multiplyBy10(int k) {
142         assert 0 <= k : "Violation of: 0 <= k";
143         assert k < RADIX : "Violation of: k < 10";
144
145         this.rep += Integer.toString(k);
146     }
147
148     @Override
149     public final int divideBy10() {
150
151         if (this.rep.isEmpty()) {
152             return 0;
153         }
154         int remainder = Integer
155                 .parseInt(this.rep.substring(this.rep.length(
    ) - 1));
156         this.rep = this.rep.substring(0, this.rep.length() -
    1);
157         return remainder;
158     }
159
160     @Override
161     public final boolean isZero() {
162         return this.rep.isEmpty();
163     }
164
165 }
166
```