

```

1 import components.naturalnumber.NaturalNumber;
2
3 /**
4  * {@code NaturalNumber} represented as a {@code String} with implementations of
5  * primary methods.
6  *
7  * @convention <pre>
8  * [all characters of $this.rep are '0' through '9'] and
9  * [$this.rep does not start with '0']
10 * </pre>
11 * @correspondence <pre>
12 * this = [if $this.rep = "" then 0
13 *         else the decimal number whose ordinary depiction is $this.rep]
14 * </pre>
15 *
16 * @author Qينو Shi & Yiming Cheng
17 */
18
19 public class NaturalNumber3 extends NaturalNumberSecondary {
20
21     /**
22      * Private members -----
23      */
24
25     /**
26      * Representation of {@code this}.
27      */
28     private String rep;
29
30     /**
31      * Creator of initial representation.
32      */
33     private void createNewRep() {
34         /**
35          * create an empty representation
36          */
37         this.rep = "";
38     }
39
40     /**
41      * Constructors -----
42      */
43
44     /**
45      * No-argument constructor.
46      */
47     public NaturalNumber3() {
48         this.createNewRep();
49     }
50
51     /**
52      * Constructor from {@code int}.
53      */
54
55

```

```

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

```

```

116     }
117
118     /*
119     * Standard methods -----
120     */
121
122     @Override
123     public final NaturalNumber newInstance() {
124         try {
125             return this.getClass().getConstructor().newInstance();
126         } catch (ReflectiveOperationException e) {
127             throw new AssertionError(
128                 "Cannot construct object of type " + this.getClass());
129         }
130     }
131
132     @Override
133     public final void clear() {
134         this.createNewRep();
135     }
136
137     @Override
138     public final void transferFrom(NaturalNumber source) {
139         assert source != null : "Violation of: source is not null";
140         assert source != this : "Violation of: source is not this";
141         assert source instanceof NaturalNumber3 : ""
142             + "Violation of: source is of dynamic type NaturalNumberExample";
143         /*
144         * This cast cannot fail since the assert above would have stopped
145         * execution in that case.
146         */
147         NaturalNumber3 localSource = (NaturalNumber3) source;
148         this.rep = localSource.rep;
149         localSource.createNewRep();
150     }
151
152     /*
153     * Kernel methods -----
154     */
155
156     @Override
157     public final void multiplyBy10(int k) {
158         assert 0 <= k : "Violation of: 0 <= k";
159         assert k < RADIX : "Violation of: k < 10";
160
161         this.rep = this.rep.concat(Integer.toString(k));
162     }
163
164     @Override
165     public final int divideBy10() {
166
167         int number = 0;
168         if (!this.rep.equals("")) {
169             /*
170             * find the last digit of the number
171             */

```

```
173         number = Integer
174             .parseInt(this.rep.substring(this.rep.length() - 1));
175         this.rep = this.rep.substring(0, this.rep.length() - 1);
176     }
177
178     return number;
179 }
180
181 @Override
182 public final boolean isZero() {
183
184     return this.rep.length() == 0;
185
186 }
187
188 }
189
```