The Ohio State University

PROJECT 2: NATURALNUMBER ON STRING

Daniil Gofman

Ansh Pachauri

SW 2: Dev & Dsgn

Paolo Bucci

Yiyang Chen

Shivam Gupta

September 12, 2023

```java
import components.naturalnumber.NaturalNumber;
import components.naturalnumber.NaturalNumberSecondary;

/**
 * {@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 Daniil Gofman
 *
 */
public class NaturalNumber3 extends NaturalNumberSecondary {

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

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

    /**
     * Creator of initial representation.
     */
    private void createNewRep() {
        // Create the representation the corresponds to the data representation
        this.rep = "";
    }

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

    /**
     * No-argument constructor.
     */
    public NaturalNumber3() {
        this.rep = "";
    }

    /**
     * Constructor from {@code int}.
     *
     * @param i
     *            {@code int} to initialize from
     */
    public NaturalNumber3(int i) {
        assert i >= 0 : "Violation of: i >= 0";
        // Check parameter i and set up proper data representation
        if (i == 0) {
            this.rep = "";
        } else {
            this.rep = Integer.toString(i);
```

```
 63         }
 64
 65     }
 66
 67     /**
 68      * Constructor from {@code String}.
 69      *
 70      * @param s
 71      *            {@code String} to initialize from
 72      */
 73     public NaturalNumber3(String s) {
 74         assert s != null : "Violation of: s is not null";
 75         assert s.matches("0|[1-9]\\d*") : ""
 76                 + "Violation of: there exists n: NATURAL (s = TO_STRING(n))";
 77         // Check parameter s and set up proper data representation
 78         if (s.equals("0")) {
 79             this.rep = "";
 80         } else {
 81             this.rep = s;
 82         }
 83     }
 84
 85     /**
 86      * Constructor from {@code NaturalNumber}.
 87      *
 88      * @param n
 89      *            {@code NaturalNumber} to initialize from
 90      */
 91     public NaturalNumber3(NaturalNumber n) {
 92         assert n != null : "Violation of: n is not null";
 93         // Check parameter n and set up proper data representation
 94         if (n.isZero()) {
 95             this.rep = "";
 96         } else {
 97             this.rep = n.toString();
 98         }
 99     }
100
101     /*
102      * Standard methods ---------------------------------------------------------
103      */
104
105     @Override
106     public final NaturalNumber newInstance() {
107         try {
108             return this.getClass().getConstructor().newInstance();
109         } catch (ReflectiveOperationException e) {
110             throw new AssertionError(
111                     "Cannot construct object of type " + this.getClass());
112         }
113     }
114
115     @Override
116     public final void clear() {
117         this.createNewRep();
118     }
119
120     @Override
121     public final void transferFrom(NaturalNumber source) {
122         assert source != null : "Violation of: source is not null";
123         assert source != this : "Violation of: source is not this";
124         assert source instanceof NaturalNumber3 : ""
```

```
125                 + "Violation of: source is of dynamic type NaturalNumberExample";
126         /*
127          * This cast cannot fail since the assert above would have stopped
128          * execution in that case.
129          */
130         NaturalNumber3 localSource = (NaturalNumber3) source;
131         this.rep = localSource.rep;
132         localSource.createNewRep();
133     }
134
135     /*
136      * Kernel methods ----------------------------------------------------------
137      */
138
139     @Override
140     public final void multiplyBy10(int k) {
141         // Check the preconditions: k should be non-negative and less than 10 (RADIX)
142         assert 0 <= k : "Violation of: 0 <= k";
143         assert k < RADIX : "Violation of: k < 10";
144
145         /*
146          * Concatenate the integer k to the current representation, effectively
147          * multiplying by 10
148          */
149         this.rep = this.rep + Integer.toString(k);
150     }
151
152     @Override
153     public final int divideBy10() {
154         // Variable to store current value
155         String numberStr = this.rep;
156         // Variable to store new representation
157         String newStr = "";
158         // Variable to store result
159         int result = 0;
160
161         if (!this.rep.isEmpty()) {
162             // Get the new value
163             newStr = numberStr.substring(0, numberStr.length() - 1);
164             // Get the remainder
165             result = Character
166                     .getNumericValue(numberStr.charAt(numberStr.length() - 1));
167             // Update the number to contain only the quotient
168             this.rep = newStr;
169         }
170         // Return the result (remainder)
171         return result;
172     }
173
174     @Override
175     public final boolean isZero() {
176         // A natural number is zero if its representation is "".
177         return this.rep.equals("");
178     }
179
180 }
181
```