

```

1 package Bi gNumbers;
2
3 /**
4  * This class is an implementation of the BNum interface using a linked-
5  * list.
6  * @author Matt Laidman
7  * @version 1.0 (March 2014)
8  */
9
10 public class LinkBNum implements BNum {
11
12     private Node lNum;
13
14
15     public LinkBNum() { // default constructor creates LinkBNum object with
        value 0
16         this(0);
17     }
18
19
20     public LinkBNum (long n) { // Takes long value and converts to integer
        node list
21         toList(String.valueOf(n));
22     }
23
24
25     public LinkBNum (String n) { // Takes String and converts to integer
        node list
26         try {
27             toList(n);
28         } catch (Exception E) { // throws BadNumberFormatException if string
            contains invalid chars
29             throw new BadNumberFormatException("Invalid number format");
30         }
31     }
32
33
34     public void toList (String n) { // adds given string to node list
35         for (char c : reverse(checkSign(n)).toCharArray()) { // assign LSD
            + reverse
36             lNum = new Node(Character.getNumericValue(c), lNum); // add to
            node list
37         }
38     }
39
40
41     public String toString ( ) { // returns string representation of 'this'
42         String t = "";
43         Node p = lNum;
44         while (p != null) {
45             t = t + p.digit;
46             p = p.next;
47         }
48         return t;
49     }
50
51
52     public BNum clone() { // returns a clone of 'this'
53         if (lNum.digit == 0) {
54             return new LinkBNum(this.toString());

```

```

55     } else {
56         return new LinkBNum("-" + this.toString().substring(1));
57     }
58 }
59
60
61 public boolean equals(BNum n) { // returns true if 'this' == n
62     return this.toString().equals(n.toString());
63 }
64
65
66 public boolean lessThan(BNum n) { // true if 'this' < n
67     Node p;
68     if (INum.digit == 0 & n.getSign() == 0) { // if both positive
69         if (this.toString().length() != n.toString().length()) { // if
different lengths
70             return this.toString().length() < n.toString().length(); /
/ true if 'this' shorter than n
71         } else { // if same length
72             p = INum.next;
73             for (int i = 1 ; i < this.toString().length() ; i++) {
74                 if (p.digit != n.getDigit(i)) { // if digits from left
to right aren't equal
75                     return p.digit < n.getDigit(i); // true if 'this'
digit less than n digit
76                 }
77                 p = p.next;
78             }
79         }
80     } else if (INum.digit == 1 & n.getSign() == 1) { // if both
negative
81         if (this.toString().length() != n.toString().length()) { // if
different lengths
82             return this.toString().length() > n.toString().length(); /
/ true if 'this' longer than n
83         } else { // if same length
84             p = INum.next;
85             for (int i = 1 ; i < this.toString().length() ; i++) {
86                 if (p.digit != n.getDigit(i)) { // if digits from left
to right aren't equal
87                     return p.digit > n.getDigit(i); // true if 'this'
digit greater than n digit
88                 }
89                 p = p.next;
90             }
91         }
92     } else { // if different signs
93         return INum.digit > n.getSign(); // true if 'this' negative and
n positive
94     }
95     return false;
96 }
97
98
99 public BNum add(BNum n) { // returns 'this' + n
100     if (INum.digit == n.getSign()) { // if same sign
101         if (INum.digit == 0) { // both are positive
102             return new LinkBNum(doAdd(this, n));
103         } else { // both are negative
104             return new LinkBNum('-' + doAdd(this, n));
105         }

```

```

106     } else { // if different signs
107         if (new LinkBNum(this.toString().substring(1)).lessThan(new LinkBNum(n.toString().substring(1)))) { // |this| < |n|
108             if (n.getDigit(0) == 1) { // if n is negative
109                 return new LinkBNum('-' + doSub(n, this));
110             } else { // if n is positive
111                 return new LinkBNum(doSub(n, this));
112             }
113         } else { // |n| < |this|
114             if (lNum.digit == 1) { // if 'this' is negative
115                 return new LinkBNum('-' + doSub(this, n));
116             } else { // if 'this' is positive
117                 return new LinkBNum(doSub(this, n));
118             }
119         }
120     }
121 }
122
123
124 private String doAdd (BNum t, BNum n) { // adds the two given BNums
125     int a = t.toString().length()-1;
126     int b = n.toString().length()-1;
127     int carry = 0;
128     int sum;
129     String total = "";
130     while (a > 0 | b > 0) { // while both have digits
131         if (a <= 0) { // if t has no more digits
132             sum = n.getDigit(b) + carry;
133         } else if (b <= 0) { // if n has no more digits
134             sum = t.getDigit(a) + carry;
135         } else { // if both still have digits
136             sum = t.getDigit(a) + n.getDigit(b) + carry;
137         }
138         if (sum > 9) {
139             sum = sum % 10;
140             carry = 1;
141         } else {
142             carry = 0;
143         }
144         total = total + sum;
145         a--;
146         b--;
147     }
148     if (carry == 1) {
149         return 1 + reverse(total);
150     } else {
151         return reverse(total);
152     }
153 }
154
155
156
157 public BNum sub(BNum n) { // returns 'this' - n
158     if (lNum.digit == n.getSign()) { // same sign
159         if (!new LinkBNum(this.toString().substring(1)).lessThan(new LinkBNum(n.toString().substring(1)))) { // |this| > |n|
160             if (lNum.digit == 0) { // both are positive
161                 return new LinkBNum(doSub(this, n));
162             } else { // both are negative
163                 return new LinkBNum('-' + doSub(this, n));
164             }

```

```

165         } else { // |this| < |n|
166             if (lNum.digit == 0) { // both are positive
167                 return new LinkBNum('-' + doSub(n, this));
168             } else { // both are negative
169                 return new LinkBNum(doSub(n, this));
170             }
171         }
172     } else { // different signs
173         if (this.lessThan(n)) { // negative sub positive
174             return new LinkBNum('-' + doAdd(this, n));
175         } else { // positive sub negative
176             return new LinkBNum(doAdd(this, n));
177         }
178     }
179 }
180
181
182 private String doSub (BNum t, BNum n) { // subtracts the two given
BNums
183     int a = t.toString().length()-1;
184     int b = n.toString().length()-1;
185     int carry = 0;
186     int diff;
187     String total = "";
188     while (a > 0 | b > 0) { // while both have digits
189         if (a <= 0) { // if t has no more digits
190             diff = n.getDigit(b) - carry;
191         } else if (b <= 0) { // if n has no more digits
192             diff = t.getDigit(a) - carry;
193         } else { // if both have digits
194             diff = t.getDigit(a) - n.getDigit(b) - carry;
195         }
196         if (diff < 0) {
197             diff = 10 + diff;
198             carry = 1;
199         } else {
200             carry = 0;
201         }
202         total = total + diff;
203         a--;
204         b--;
205     }
206     return reverse(total);
207 }
208
209
210 public int getSign() { // returns the LSD of the BNum
211     return lNum.digit;
212 }
213
214
215 public int getDigit(int i) { // returns the digit at the given index, 0
is LSD
216     Node p;
217     try {
218         p = lNum;
219         for (int j = 0 ; j < i ; j++) {
220             p = p.next;
221         }
222         return p.digit;
223     } catch (Exception E) { // throws DigitOutOfRangeException if index

```

```

223 out of range
224         throw new DigitOutOfRangeException("Index out of range");
225     }
226 }
227
228
229 private String reverse (String n) { // returns the reverse of given
    string
230     String rTotal = "";
231     for (int i = n.length()-1 ; i >= 0 ; i--) {
232         rTotal = rTotal + n.charAt(i);
233     }
234     return rTotal;
235 }
236
237
238 private String checkSign (String n) { // checks string and applies LSD
239     if (n.charAt(0) == '-') {
240         return 1 + n.substring(1);
241     } else if (n.charAt(0) == '0') {
242         return n;
243     } else {
244         return 0 + n;
245     }
246 }
247 }

```