```java
package BigNumbers;

/**
 * This class is an implementation of the BNum interface using a contiguous
 *
 * @author Matt Laidman
 * @version 1.0 (March 2014)
 */

public class ConBNum implements BNum {

    private int[] aNum;


    public ConBNum ( ) { // default constructor creates ConBNum object with value 0
        this(0);
    }


    public ConBNum (long n) { // converts long to int array
        toArray(String.valueOf(n));
    }


    public ConBNum (String n) { // converts string to int array
        try {
            toArray(n);
        } catch (Exception E) {
            throw new BadNumberFormatException("Invalid number format");
        }
    }


    public void toArray (String n) { // converts given string to int array
        n = checkSign(n);
        aNum = new int[n.length()];
        for (int i = 0 ; i < n.length() ; i++) {
            aNum[i] = Integer.parseInt(String.valueOf(n.charAt(i)));
        }
    }


    public String toString ( ) { // returns string representation of 'this'
        String t = "";
        for (int i : aNum) {
            t = t + i;
        }
        return t;
    }


    public BNum clone ( ) { // returns a clone of 'this'
        if (aNum[0] == 0) {
            return new ConBNum(this.toString());
        } else {
            return new ConBNum('-' + this.toString().substring(1));
        }
    }


```

```
61      public boolean equals (BNum n) { // returns 'this' == n
62          return this.toString().equals(n.toString());
63      }
64
65
66      public boolean lessThan (BNum n) { // returns 'this' < n
67          if (aNum[0] ==  0 & n.getSign() == 0) { // if both positive
68              if (aNum.length != n.toString().length()) { // if different
    lengths
69                  return aNum.length < n.toString().length(); // true if '
    this' shorter than n
70              } else { // if same length
71                  for (int i = 1 ; i < aNum.length ; i++) {
72                      if (aNum[i] != n.getDigit(i)) { // if digits from left
     to right aren't equal
73                          return aNum[i] < n.getDigit(i); // true if 'this'
    digit less than n digit
74                      }
75                  }
76                  return false;
77              }
78          } else if (aNum[0] == 1 & n.getSign() == 1) { // if both negative
79              if (aNum.length != n.toString().length()) { // if different
    lengths
80                  return aNum.length > n.toString().length(); // true if '
    this' longer than n
81              } else { // if same length
82                  for (int i = 1 ; i < aNum.length ; i++) {
83                      if (aNum[i] != n.getDigit(i)) { // if digits from left
     to right aren't equal
84                          return aNum[i] > n.getDigit(i); // true if 'this'
    digit greater than n digit
85                      }
86                  }
87                  return false;
88              }
89          } else { // if different signs
90              return aNum[0] > n.getSign(); // true if 'this' negative and n
    positive
91          }
92      }
93
94
95      public BNum add (BNum n){ // returns 'this' + n
96          if (aNum[0] == n.getSign()) { // if same sign
97              if (aNum[0] == 0) { // both are positive
98                  return new ConBNum(doAdd(this, n));
99              } else { // both are negative
100                 return new ConBNum('-' + doAdd(this, n));
101             }
102         } else { // if different signs
103             if (new ConBNum(this.toString().substring(1)).lessThan(new Con
    BNum(n.toString().substring(1)))) { // |this| < |n|
104                 if (n.getDigit(0) == 1) { // if n is negative
105                     return new ConBNum('-' + doSub(n, this));
106                 } else { // if n is positive
107                     return new ConBNum(doSub(n, this));
108                 }
109             } else { // |n| < |this|
110                 if (aNum[0] == 1) { // if 'this' is negative
111                     return new ConBNum('-' + doSub(this, n));
```

```
112                 } else { // if 'this' is positive
113                     return new ConBNum(doSub(this, n));
114                 }
115             }
116         }
117     }
118 
119 
120     private String doAdd (BNum t, BNum n) { // adds the two given BNums
121         int a = t.toString().length()-1;
122         int b = n.toString().length()-1;
123         int carry = 0;
124         int sum;
125         String total = "";
126         while (a > 0 | b > 0) { // while both have digits
127             if (a <= 0) { // if t has no more digits
128                 sum = n.getDigit(b) + carry;
129             } else if (b <= 0) { // if n has no more digits
130                 sum = t.getDigit(a) + carry;
131             } else { // if both still have digits
132                 sum = t.getDigit(a) + n.getDigit(b) + carry;
133             }
134             if (sum > 9) {
135                 sum = sum % 10;
136                 carry = 1;
137             } else {
138                 carry = 0;
139             }
140             total = total + sum;
141             a--;
142             b--;
143         }
144         if (carry == 1) {
145             return 1 +  reverse(total);
146         } else {
147             return reverse(total);
148 
149         }
150     }
151 
152 
153     public BNum sub (BNum n){ // returns 'this' - n
154         if (aNum[0] == n.getSign()) { // same sign
155             if (!new ConBNum(this.toString().substring(1)).lessThan(new Co
    nBNum(n.toString().substring(1)))) { // |this| > |n|
156                 if (aNum[0] == 0) { // both are positive
157                     return new ConBNum(doSub(this, n));
158                 } else { // both are negative
159                     return new ConBNum('-' + doSub(this, n));
160                 }
161             } else { // |this| < |n|
162                 if (aNum[0] == 0) { // both are positive
163                     return new ConBNum('-' + doSub(n, this));
164                 } else { // both are negative
165                     return new ConBNum(doSub(n, this));
166                 }
167             }
168         } else { // different signs
169             if (this.lessThan(n)) { // negative sub positive
170                 return new ConBNum('-' + doAdd(this, n));
171             } else { // positive sub negative
```

```
172                 return new ConBNum(doAdd(this, n));
173             }
174         }
175     }
176 
177 
178     private String doSub (BNum t, BNum n) { // subtracts the two given
    BNums
179         int a = t.toString().length()-1;
180         int b = n.toString().length()-1;
181         int carry = 0;
182         int diff;
183         String total = "";
184         while (a > 0 | b > 0) { // while both have digits
185             if (a <= 0) { // if t has no more digits
186                 diff = n.getDigit(b) - carry;
187             } else if ( b <= 0) { // if n has no more digits
188                 diff = t.getDigit(a) - carry;
189             } else { // if both have digits
190                 diff = t.getDigit(a) - n.getDigit(b) - carry;
191             }
192             if (diff < 0) {
193                 diff = 10 + diff;
194                 carry = 1;
195             } else {
196                 carry = 0;
197             }
198             total = total + diff;
199             a--;
200             b--;
201         }
202         return reverse(total);
203     }
204 
205 
206     public int getSign() { // returns sign of BNum
207         return aNum[0];
208     }
209 
210 
211     public int getDigit(int i) { // returns digit at index i, 0 is LSD
212         try {
213             return aNum[i];
214         } catch (Exception E) {
215             throw new DigitOutOfRangeException("Index out of range");
216         }
217     }
218 
219 
220     private String reverse (String n) { // returns the reverse of the
    passed string
221         String rTotal = "";
222         for (int i = n.length()-1 ; i >= 0 ; i--) {
223             rTotal = rTotal + n.charAt(i);
224         }
225         return rTotal;
226     }
227 
228 
229     private String checkSign (String n) { // checks the given string and
    applies appropriate LSD if necessary
```

```
230         if (n.charAt(0) == '-') {
231             return 1 + n.substring(1);
232         } else if (n.charAt(0) == '0') {
233             return n;
234         } else {
235             return 0 + n;
236         }
237     }
238 }
```