```
 1 /** This class  ConBNum implements BNum methods
 2  * It performs a number of tests including add, subtract, clone, getDigit
 3   * lessThan, getSign and tests of the exceptions.
 4   *
 5   *  *Name: Long Nguyen: Student # 5427059
 6   *
 7   * @version  1.0 (Mar. 2014)                                    */
 8
 9 package BigNumbers;
10 import java.io.*;
13
14
15 public class ConBNum implements BNum, Serializable  {
16
17    //Class fields
18    public int num [];
19    public int sign;
20
21   /** This default constructor ConBNum , construct an object with positive
    zero                */
22
23 // The default constructor
24 public ConBNum ( ) {
25
26    //creates an object with a positive zero
27    this(0);
28
29    }; // constructor
30
31    /** This constructor produces  a Long of the form:  */
32
33  // The constructor that takes a long type
34 public ConBNum ( long n ) {
35
```

```
36    int sizeOfArray = 0; //variable for size of the array to be made
37    //converting the number to a char array
38    char[] characters = String.valueOf(n).toCharArray();
39
40    /*Check to see if the strings is a valid number
41     * If it not a valid number then throw a run time exception
42     * */
43    for (char c :characters) {
44       try {
45       if (Character.isLetter(c)) throw new BigNumbersException("This is not a
   valid "+ c + " Not a valid number!");
46       }
47       catch (BigNumbersException e) {
48          System.out.println("\n There seems to be a problem:  "+ "This is not a
   valid "+ c + e.getMessage());
49       }
50    }
51
52 //checks to see if the first value is a negative
53    if(characters[0] ==  '-'){
54       sizeOfArray = characters.length-1;  //getting the length of the array
   without getting the negative char value
55       num = new int[sizeOfArray]; //creating the new array with the size from
   get length
56       this.sign = -1; // setting the sign to negative
57
58    //looping through the char array to add the values to the class
59    for(int i = 1; i<characters.length; i++) {
60       //if its the last value of the array, then multiply it by negative one
61       if(i == 1){
62          this.num[i-1] = ((characters[i]-48));
63          //this.num[i-1] = ((characters[i]-48)*-1);
64       }else{
65       num[i-1] = characters[i]-48;//Subtract 48 because of the ASCII value
```

```
66      }
67     }//end for loop
68
69   }else{
70      sizeOfArray = characters.length;  //getting the length of the array
71      num = new int[sizeOfArray]; //creating the new array with the size from
   get length
72      this.sign = 1; //positive value will be positive one
73      for(int i = 0; i<characters.length; i++)
74       {
75         num[i] = characters[i]-48;//Subtract 48 because of the ASCII value
76           //System.out.println( characters[1]);
77       }
78   }//end else
79
80   }; // end ConBNum ( long n )  constructor
81
82   /** This constructor produces  a String of the form:  */
83
84 public ConBNum ( String s ) {
85
86   s = s.trim(); //Trimming the white spaces of the string
87   int sizeOfArray = 0; //variable for size of the array to be made
88   char[] characters = String.valueOf(s).toCharArray(); //converting the number
   to a char array
89
90   /*Check to see if the strings is a valid number
91    * If it not a valid number then throw a run time exception
92    * */
93   for (char c :characters) {
94      try {
95      if (Character.isLetter(c)) throw new BigNumbersException("Not a valid
   number!");
96      }
```

```
 97      catch (BigNumbersException e) {
 98         System.out.println("\n There seems to be a problem:  "+ "This is not a
    valid, "+ c +"! "+ e.getMessage());
 99      }
100   }
101
102   //if the number is negative
103   if(characters[0] ==  '-'){ //check the first character if it's a negative
104      sizeOfArray = characters.length-1; //getting the length of the array
    without the negative value
105      num = new int[sizeOfArray];  //setting the length of the array without the
    negative value
106      this.sign = -1;  //setting negative one as the sign
107
108      //looping through the char array to add the values to the class
109   for(int i = 1; i<characters.length; i++) {
110      if(i == 1){//multiply the last index of the array by negative one
111         this.num[i-1] = ((characters[i]-48));
112         //this.num[i-1] = ((characters[i]-48)*-1);
113      }else{
114      this.num[i-1] = characters[i]-48;//Subtract 48 because of the ASCII value
115      }
116   } //end for loop
117
118   }else{
119      sizeOfArray = characters.length;  //getting the length of the array
120      this.num = new int[sizeOfArray]; //setting the length of the array
121      this.sign = 1; //setting sign to positive one
122      for(int i = 0; i<characters.length; i++)//looping through the array and
    added the values to the num field
123      {
124         this.num[i] = characters[i]-48;//Subtract 48 because of the ASCII value
125      }
126   }
```

```
127   }; // end of the ConBNum ( String s ) constructor
128
129   /*Create a clone of this. Object.  The For loop will loop through the values of
      the this object
130    * and adding it to the string
131    *
132    */
133   public BNum clone(){
134
135      String cloneNumberValue = "";
136      //getting the sign of the number
137      int getSign = this.getSign();
138
139      //looping through the values of the object that called this
140      for(int i = 0; i < this.num.length; i++){
141
142         //Multiplying the first digital by the sign
143         if(i == 0 ){
144            cloneNumberValue  += (this.num[i])*getSign;
145         }else{
146         //adding the values to the cloneNumberValue string
147         cloneNumberValue  += this.num[i];
148         }
149      }//end for loop
150
151      //creating a new object with all the values that called the clone method
152      BNum copy = new ConBNum(cloneNumberValue);
153
154      //returning the object that called this method
155      return copy;
156   }
157
158   /* Returns true if 'this' =  n*/
159   public boolean equals( BNum n) {
```

```
160
161       boolean equals = false;
162
163       String valueOfNinString = BNumValue(n); //Getting the value of BNum
164       ConBNum copyingBNum =  new ConBNum(valueOfNinString); //creating
      a copy of the BNum n that's being passed in
165
166       int lengthOfBNum =  copyingBNum.num.length;//getting the length of
      BNum that's being passed in
167
168       /*If the length are equal then
169       loop through this check to see if the values are equal*/
170       for(int i = 0; i < this.num.length; i++){
171             if (lengthOfBNum == this.num.length && this.num[i] ==
      copyingBNum.getDigit(i)){
172                equals = true;
173             }else{
174                equals = false;
175             }
176       }//end for loop
177
178       return equals; //return the values if it's equal
179    }
180
181    /* Returns true if  'this' < n */
182    public boolean lessThan(BNum n ){
183
184       boolean lessThan = false;
185       String copyingThisBNumString = "";
186       String copyingBNumString = "";
187
188       String valueOfNinString = BNumValue(n); //Getting the value of BNum
189
190       ConBNum copyingBNum =  new ConBNum(valueOfNinString); //creating
```

```
    a copy of the BNum n that's being passed in
191     ConBNum copyingThisBNum =  (ConBNum)this.clone(); //creating a copy
    of the BNumThis that's calling the method
192
193     //find the different between the two lengths
194     int diferentLenghtBetweenBothValues=
    Math.abs(Math.abs(copyingThisBNum.num.length) -
    Math.abs(copyingBNum.num.length));
195
196     //adding the values to the copyingThisBNum string
197     for(int i = 0; i < copyingThisBNum.num.length; i++){
198       copyingThisBNumString  += copyingThisBNum.num[i];
199     }
200
201     //adding the values to the copyingBNum string
202     for(int i = 0; i < copyingBNum.num.length; i++){
203       copyingBNumString  += copyingBNum.num[i];
204     }
205
206     //check what length is longer, if one of the lengths is longer then pad the
    beginning with leading zeros
207     if(copyingThisBNumString.length() < copyingBNumString.length()){
208     for(int x = diferentLenghtBetweenBothValues-1; x >= 0; x--){
209       copyingThisBNumString = 0 + copyingThisBNumString;
210     }
211     }
212
213     //check what length is longer, if one of the lengths is longer then pad the
    beginning with leading zeros
214     if(copyingThisBNumString.length() > copyingBNumString.length()){
215       for(int x = diferentLenghtBetweenBothValues-1; x >= 0; x--){
216         copyingBNumString = 0 + copyingBNumString;
217       }
218       }
```

```
219
220       //Comparing both strings to see which one is bigger
221       if(copyingThisBNumString.compareTo( copyingBNumString ) < 0){
222          lessThan = true;
223       }
224       //
      System.out.println(copyingThisBNumString.compareTo( copyingBNumString
      ));
225
226       return lessThan;
227    }
228    /* returns 'this' + n*/
229
230    /*This add method add the to values together
231     * It adds it by checking which lenght is bigger
232     * */
233
234    public String BNumValue(BNum n){
235
236       int signOfN = n.getSign(); //Getting the sign of BNum N
237
238       int counter = 0;  //Counter for the while loop
239       String valueOfNString = ""; //Variable to hold the numbers of BNum N
240       int lenght = n.getDigit(-1);
241
242       //System.out.println("Lenght is "+ lenght);
243
244        // while(n.getDigit(counter) != -1){
245            while(counter <= lenght-1){
246            valueOfNString += n.getDigit(counter);
247          counter++;
248          //System.out.println(valueOfNinString );
249       }
250        int  firstCharacterOfN = (int) valueOfNString.charAt(0) -48;
```

```
251         valueOfNString =  (firstCharacterOfN*signOfN) +
    valueOfNString.substring(1, valueOfNString.length());
252
253       return valueOfNString;
254     }
255
256     public BNum add(BNum n){
257
258       String addingBothValuesString = ""; //String to adding all the digit
    together
259       boolean remainder = false; //Setting the remainder to false by default
260
261       String valueOfNinString = BNumValue(n); //Getting the value of BNum
262
263       ConBNum copyingBNum =   new ConBNum(valueOfNinString);
264       ConBNum copyingThisBNum =  (ConBNum)this.clone(); //creating a copy
    of the BNumThis that's calling the method
265
266       /*Getting the sign of both values before deciding to add or subtract */
267       int signOfBNum = copyingBNum.getSign();
268       int signofThisBNum = copyingThisBNum.getSign();
269
270       if(signOfBNum == signofThisBNum){
271
272       ////////////////////////////////////////////////////////////////////////
273       int  smallerLength = 0; // variable to hold the smaller length
274       int largerIndex = 0;  // variable to hold the bigger length
275       ConBNum  largerLengthConBNum = null;
276       ConBNum smallerLenghtConBNum = null;
277
278       /*Find out which BNum length is bigger
279        * length is less or greater then*/
280       if(copyingThisBNum.num.length <  copyingBNum.num.length){
281             smallerLength = copyingThisBNum.num.length;  // getting the
```

```
    lenght of the value
282             largerLengthConBNum = copyingBNum;
283             smallerLenghtConBNum = copyingThisBNum;
284             largerIndex  = largerLengthConBNum.num.length-1; //starting at the
    last index
285     }
286     /*Find out which BNum length is bigger
287      * length is less or greater then*/
288     if(copyingThisBNum.num.length >  copyingBNum.num.length){
289           smallerLength =  copyingBNum.num.length;
290           largerLengthConBNum = copyingThisBNum;
291           smallerLenghtConBNum = copyingBNum;
292           largerIndex  = largerLengthConBNum.num.length-1; //starting at the
    last index
293     }
294     /*Find out which BNum length is bigger
295      * length is less or greater then*/
296     if(copyingThisBNum.num.length ==  copyingBNum.num.length &&
    copyingThisBNum.num[0] < copyingBNum.num[0]){
297          smallerLength = copyingThisBNum.num.length;  // getting the lenght
    of the value
298           largerLengthConBNum = copyingBNum;
299           smallerLenghtConBNum = copyingThisBNum;
300           largerIndex  = largerLengthConBNum.num.length-1; //starting at the
    last index
301   }
302     /*Find out which BNum length is bigger
303      * length is less or greater then*/
304     if(copyingThisBNum.num.length ==  copyingBNum.num.length &&
    copyingThisBNum.num[0] >= copyingBNum.num[0]){
305         smallerLength =  copyingBNum.num.length;
306         largerLengthConBNum = copyingThisBNum;
307         smallerLenghtConBNum = copyingBNum;
308         largerIndex  = largerLengthConBNum.num.length-1; //starting at the
```

```
    last index
309   }
310
311       //getting the different between the two values and calling the Math
    absolute value
312       int diferentLenghtBetweenBothValues=
    Math.abs(Math.abs(copyingThisBNum.num.length) -
    Math.abs(copyingBNum.num.length));
313       //System.out.println(diferentLenghtBetweenBothValues);
314       /*This while loop add the numbers together in both Arrays, each element
    at a time
315        * If the smallerLenght is less then zero, then it will break out of the loop
316        *
317        * */
318
319       while (smallerLength > 0 ){
320
321          /*This if statement checks if adding both values together is less then 10,
322           * and BNumLenght is not equal to 0
323           * */
324          if(remainder ==  false && largerLengthConBNum.num[largerIndex] +
    smallerLenghtConBNum.getDigit(smallerLength-1) < 10 && smallerLength !=
    0){
325             addingBothValuesString = largerLengthConBNum.num[largerIndex]
    + smallerLenghtConBNum.getDigit(smallerLength-1) +
    addingBothValuesString;
326             if(smallerLength >0){
327                smallerLength--;
328                }
329             if(largerIndex >0){
330                largerIndex--;
331                }
332             remainder = false;
333          }
```

```
334
335         /*This if statement checks if adding both values together plus a 1 is less
    then 10,
336          * and BNumLenght is not equal to 0
337          * */
338         if(remainder ==  true && largerLengthConBNum.num[largerIndex] +
    smallerLenghtConBNum.getDigit(smallerLength-1) < 10 && smallerLength !=
    0){
339             addingBothValuesString = ((largerLengthConBNum.num[largerIndex]
    + smallerLenghtConBNum.getDigit(smallerLength-1))+1) +
    addingBothValuesString;
340
341             if(smallerLength >0){
342                 smallerLength--;
343                 }
344             if(largerIndex >0){
345                 largerIndex--;
346                 }
347             remainder = false;
348         }
349
350         /*This if statement checks if adding both values together  is greater
    then 10,
351          * and BNumLenght is not equal to 0
352          * */
353         if(remainder ==  false && largerLengthConBNum.num[largerIndex] +
    smallerLenghtConBNum.getDigit(smallerLength-1) >= 10 && smallerLength !=
    0){
354             addingBothValuesString = ((largerLengthConBNum.num[largerIndex]
    + smallerLenghtConBNum.getDigit(smallerLength-1))%10) +
    addingBothValuesString;
355
356             if(smallerLength >0){
357                 smallerLength--;
```

```
358             }
359           if(largerIndex >0){
360             largerIndex--;
361             }
362           remainder = true;
363         }//end if
364
365         /*This if statement checks if adding both values together  plus the
    remainder is greater then 10,
366          * and BNumLenght is not equal to 0
367          * */
368         if(remainder ==  true && largerLengthConBNum.num[largerIndex] +
    n.getDigit(smallerLength-1) >= 10 && smallerLength !=  0){
369           addingBothValuesString =
    (((largerLengthConBNum.num[largerIndex] +
    smallerLenghtConBNum.getDigit(smallerLength-1))+1) %10) +
    addingBothValuesString;
370
371           if(smallerLength >0){
372             smallerLength--;
373             }
374           if(largerIndex >0){
375             largerIndex--;
376             }
377           remainder = true;
378         }
379
380         if(largerIndex == 0 && remainder == true && smallerLength !=  0){
381           addingBothValuesString = (((largerLengthConBNum.num[0] +
    smallerLenghtConBNum.getDigit(0))%10)+1) +  addingBothValuesString;
382           addingBothValuesString =1 + addingBothValuesString;
383           break;
384         }
385
```

```
386         if(largerIndex == 0 && remainder == false && smallerLength !=  0){
387               addingBothValuesString = (((largerLengthConBNum.num[0] +
    smallerLenghtConBNum.getDigit(0)))) + addingBothValuesString;
388             break;
389         }
390         //end if
391
392    }//while  loop
393
394       //System.out.println(remainder);
395
396       /*adding the rest of the number if the BNumLeght is smaller then the
    this.Lenght;
397        * This for loop check if there's carrying over, that needs to been adding
398        *
399        * */
400         for(int x = diferentLenghtBetweenBothValues-1; x >= 0; x--){
401           if(remainder ==  true  &&  (largerLengthConBNum.num[x] +1) >= 10){
402               addingBothValuesString = (largerLengthConBNum.num[x] +1)%10
    +  addingBothValuesString;
403
404               //adding the very last digit to the end of the string
405              if(x == 0){
406                 addingBothValuesString = 1 +  addingBothValuesString;
407              }
408               remainder =  true;
409            }//end if
410             if(remainder ==  true  &&  (largerLengthConBNum.num[x] +1) < 10){
411                 addingBothValuesString = (largerLengthConBNum.num[x] +1) +
    addingBothValuesString;
412                  remainder =  false;
413          }if(remainder ==  false){
414               addingBothValuesString = (largerLengthConBNum.num[x]) +
    addingBothValuesString;
```

```
415             }//end else
416
417         }//end for loop
418
419         //casting the first character to an int and subtracting 48 because of the
    ASSIIC value
420         int  firstCharacter = (int)addingBothValuesString.charAt(0) -48;
421
422         //Multiplying the first part of the string by the sign
423          addingBothValuesString =  (firstCharacter*signofThisBNum) +
    addingBothValuesString.substring(1, addingBothValuesString.length());
424
425        // System.out.println( "addingBothValuesString " +
    addingBothValuesString);
426
427       //creating a new object with the new values and returning it
428
429      }/**********end of  if very top *********/
430
431      else{
432         //else if the sign match up then call the subtraction method
433         ConBNum addingConBNum =  (ConBNum)
    copyingThisBNum.sub(copyingBNum);
434
435         for(int i = 0; i < addingConBNum.num.length; i++){
436            addingBothValuesString += addingConBNum.getDigit(i);
437         }
438
439      }
440      ConBNum addingConBNum = new ConBNum(addingBothValuesString);
441
442
443      return addingConBNum;
444   }
```

```
445
446    /*returns 'this' - n*/
447    public BNum sub(BNum n){
448
449        String subtractingBothValuesString = ""; //String to adding all the digit
       together
450        boolean regrouping = false; //Setting the remainder to false by default
451
452        String valueOfNinString = BNumValue(n); //Getting the value of BNum
453
454            ConBNum copyingBNum =   new ConBNum(valueOfNinString); //
       creating a copy of the BNum n that's being passed in
455
456        ConBNum copyingThisBNum =  (ConBNum)this.clone(); //creating a copy
       of the BNumThis that's calling the method
457
458        /*Getting the sign of both values before deciding to add or subtract
459         * if the sign are the same then add, if they are different the substact */
460        int signOfBNum = (copyingBNum.getSign())*-1;
461        int signofThisBNum = copyingThisBNum.getSign();
462
463        if(signOfBNum != signofThisBNum){
464
465        ////////////////////////////////////////////////////////////////////////
466        int  smallerLength = 0; // variable to hold the smaller length
467        int largerIndex = 0;  // variable to hold the bigger length
468        ConBNum  largerLengthConBNum = null;
469        ConBNum smallerLenghtConBNum = null;
470
471        /*Find out which BNum length is bigger
472         * length is less or greater then*/
473        if(copyingThisBNum.num.length <  copyingBNum.num.length){
474                smallerLength = copyingThisBNum.num.length;  // getting the
       lenght of the value
```

```
475             largerLengthConBNum = copyingBNum;
476             smallerLenghtConBNum = copyingThisBNum;
477             largerIndex  = largerLengthConBNum.num.length-1;
478     }
479     /*Find out which BNum length is bigger
480      * length is less or greater then*/
481     if(copyingThisBNum.num.length >  copyingBNum.num.length){
482           smallerLength =  copyingBNum.num.length;
483           largerLengthConBNum = copyingThisBNum;
484           smallerLenghtConBNum = copyingBNum;
485           largerIndex  = largerLengthConBNum.num.length-1;
486     }
487     /*Find out which BNum length is bigger
488      * length is less or greater then*/
489     if(copyingThisBNum.num.length ==  copyingBNum.num.length &&
    copyingThisBNum.num[0] < copyingBNum.num[0]){
490          smallerLength = copyingThisBNum.num.length;  // getting the lenght
    of the value
491             largerLengthConBNum = copyingBNum;
492             smallerLenghtConBNum = copyingThisBNum;
493             largerIndex  = largerLengthConBNum.num.length-1;
494   }
495     /*Find out which BNum length is bigger
496      * length is less or greater then*/
497     if(copyingThisBNum.num.length ==  copyingBNum.num.length &&
    copyingThisBNum.num[0] >= copyingBNum.num[0]){
498         smallerLength =  copyingBNum.num.length;
499         largerLengthConBNum = copyingThisBNum;
500         smallerLenghtConBNum = copyingBNum;
501         largerIndex  = largerLengthConBNum.num.length-1;
502   }
503
504     //getting the different between the two values and calling the Math
    absolute value
```

```
505      int diferentLenghtBetweenBothValues=
    Math.abs(Math.abs(copyingThisBNum.num.length) -
    Math.abs(copyingBNum.num.length));
506
507      /*This while loop add the numbers together in both Arrays, each element
    at a time
508       * If the smallerLenght is less then zero, then it will break out of the loop
509       *
510       * */
511
512      while (smallerLength > 0 ){
513
514         /*This if statement checks if adding both values together is less then 10,
515          * and BNumLenght is not equal to 0
516          * */
517         //System.out.println(largerLengthConBNum.num[largerIndex]);
518         if(regrouping ==  false && largerLengthConBNum.num[largerIndex] -
    smallerLenghtConBNum.getDigit(smallerLength-1) < 0 && smallerLength !=  0)
    {
519            subtractingBothValuesString =
    (largerLengthConBNum.num[largerIndex]+10) -
    smallerLenghtConBNum.getDigit(smallerLength-1) +
    subtractingBothValuesString;
520            if(smallerLength >0){
521               smallerLength--;
522               }
523            if(largerIndex >0){
524            largerIndex--;
525            }
526            regrouping = true;
527         }
528
529         if(regrouping ==  true && largerLengthConBNum.num[largerIndex] -
    smallerLenghtConBNum.getDigit(smallerLength-1) < 0 && smallerLength !=  0)
```

```
    {
530             subtractingBothValuesString =
    (((largerLengthConBNum.num[largerIndex]-1)+10) -
    smallerLenghtConBNum.getDigit(smallerLength-1)) +
    subtractingBothValuesString;
531
532             if(smallerLength >0){
533               smallerLength--;
534               }
535             if(largerIndex >0){
536               largerIndex--;
537               }
538             regrouping = true;
539         }
540
541         if(regrouping ==  true && largerLengthConBNum.num[largerIndex] -
    smallerLenghtConBNum.getDigit(smallerLength-1) >=  0 && smallerLength !=
    0){
542             subtractingBothValuesString =
    (((largerLengthConBNum.num[largerIndex])-1) -
    smallerLenghtConBNum.getDigit(smallerLength-1)) +
    subtractingBothValuesString;
543
544             if(smallerLength >0){
545               smallerLength--;
546               }
547             if(largerIndex >0){
548               largerIndex--;
549               }
550             regrouping = false;
551         }
552
553         if(regrouping ==  false && largerLengthConBNum.num[largerIndex] -
    smallerLenghtConBNum.getDigit(smallerLength-1) >=  0 && smallerLength !=
```

```
    0){
554             subtractingBothValuesString =
    (((largerLengthConBNum.num[largerIndex])) -
    smallerLenghtConBNum.getDigit(smallerLength-1)) +
    subtractingBothValuesString;
555
556             if(smallerLength >0){
557                smallerLength--;
558                }
559             if(largerIndex >0){
560                largerIndex--;
561                }
562             regrouping = false;
563          }
564
565    }//while  loop
566
567       /*adding the rest of the number if the BNumLeght is smaller then the
    this.Lenght;
568        * This for loop check if there's carrying over, that needs to been adding
569        *
570        * */
571       for(int x = diferentLenghtBetweenBothValues-1; x >= 0; x--){
572             if(regrouping ==  true  &&  (largerLengthConBNum.num[x] - 1) < 0){
573                subtractingBothValuesString = ((largerLengthConBNum.num[x]
    -1)+10) +  subtractingBothValuesString;
574
575                 //adding the very last digit to the end of the string
576                if(x == 0){
577                    subtractingBothValuesString = ((largerLengthConBNum.num[x]
    -1)) +  subtractingBothValuesString;
578                }
579                regrouping =  true;
580             }//end if
```

```
581             if(regrouping == true  &&  (largerLengthConBNum.num[x] - 1) > 0){
582                 subtractingBothValuesString = (largerLengthConBNum.num[x] -1)
    +  subtractingBothValuesString;
583                 regrouping =  false;
584             }
585       //check if the second last digit is not zero so it doesn't take away from the
    first and add that at the end
586       if(regrouping ==  false && largerLengthConBNum.num[1] != 0){
587           subtractingBothValuesString = (largerLengthConBNum.num[x]) +
    subtractingBothValuesString;
588       }//end else
589
590         }//end for loop
591
592
593       //casting the first character to an int and subtracting 48 because of the
    ASSIIC value
594       int  firstCharacter = (int)subtractingBothValuesString.charAt(0) -48;
595
596       //Multiplying the first part of the string by the sign
597       subtractingBothValuesString =  (firstCharacter*signofThisBNum) +
    subtractingBothValuesString.substring(1,
    subtractingBothValuesString.length());
598
599         //System.out.println( "subtractingBothValuesString " +
    subtractingBothValuesString);
600
601       }/*end of the long if statement */
602       else{
603         //else if the sign match up then call the addition method
604         ConBNum addingConBNum =  (ConBNum)
    copyingThisBNum.add(copyingBNum);
605
606         for(int i = 0; i < addingConBNum.num.length; i++){
```

```
607             subtractingBothValuesString += addingConBNum.getDigit(i);
608           }//end for loop
609
610       }//end else
611
612           //creating a new object with the new values and returning it
613
614       ConBNum subtractingConBNum = new
      ConBNum(subtractingBothValuesString);
615
616       return subtractingConBNum;
617
618   }
619
620   /*Returns the sign of this BigNumber object */
621   public int getSign(){
622
623       //variable to hold the sign
624       int signOfNumber = 0;
625
626       //if sign is positive
627       if(this.sign == 1){
628           signOfNumber = 1;
629       }
630       //if sign is negative
631       if(this.sign == -1){
632           signOfNumber = -1;
633       }
634       //return the value of this sign
635       return signOfNumber;
636   }
637
638   //Returns the digit i of this object, digit 0 is LSD.
639   public int getDigit(int i){
```

```
640
641       //variable to store the number
642       int number = 0;
643
644       //if it's -1 then return the length
645       if(i == -1){
646          return this.num.length;
647       }
648
649       try{
650          number = this.num[i]; //getting the number at the location requested
651       }
652       catch(ArrayIndexOutOfBoundsException e){
653          System.out.println("\nOops, array out of bounds went to far, better go
    back to 0! , Early Termination \nReason:\t " + e.getMessage());
654          return -1;
655
656       }
657       return number;  //returning that number
658
659    }; // constructor
660
661
662 }
663
```