```
1 /** This class  LinkBNum 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
11 import java.io.*;
12
13 public class LinkBNum implements BNum, Serializable {
14
15    public int num;
16    public LinkBNum nextLink;
17    public LinkBNum head = null;
18    public int lenghtOfNumber;
19    public  int sign;
20
21    // method to insert the values into the linkLists
22    public void insert(char[] s) {
23
24       // converting the object that's being passed in as an character array
25       char[] characters = String.valueOf(s).toCharArray();
26       if (characters[0] == '-') { // checking to see if the number is negative
27          this.sign = -1; // Setting the sign to a negative one because the
28                          // value is negative
29          this.lenghtOfNumber = characters.length - 1; // getting the length
30                                                       // of the number;
31
32          /*
33           * Starting at the end of the character array and loop backward to
34           * create the linklisted with the values
```

```
35          */
36          for (int i = characters.length - 1; i >= 1; i--) {
37             LinkBNum link = new LinkBNum(characters[i]); // a new linklisted
38                                                 // one by one
39             link.nextLink = head;
40             head = link;
41
42          }// end for loop
43       } else {// if the character is not negative
44          this.sign = 1; // if the value is positive then set the sign to
45                      // positive 1
46          this.lenghtOfNumber = characters.length; // getting the length of
47                                             // the number;
48          // putting the numbers in the opposite order
49          for (int i = characters.length - 1; i >= 0; i--) {
50             LinkBNum link = new LinkBNum(characters[i]); // a new linklisted
51                                                 // one by one
52             link.nextLink = head;
53             head = link;
54          }
55       }// end else
56    }// end insert method
57
58    ////////////////////////////////////////////////////
59    ///optional method
60    public void printList() {
61       LinkBNum currentLink = head;
62       System.out.print("List: ");
63       while (currentLink != null) {
64          currentLink.printLink();
65          currentLink = currentLink.nextLink;
66       }
67       System.out.println("");
68    }
```

```
69 ////////////////////////////////////////////////////////////////
70    /**
71     * This default constructor ConBNum , construct an object with positive zero
72     */
73
74    // Print Link data
75    public void printLink() {
76       System.out.print(num + " ");
77    }
78
79    public LinkBNum() {
80
81       this(0); // creates an object with a positive zero
82
83    }; // constructor
84
85    /** This constructor produces a Long of the form: */
86
87    public LinkBNum(long n) {
88
89       char[] characters = String.valueOf(n).toCharArray();
90
91       /*Check to see if the strings is a valid number
92        * If it not a valid number then throw a run time exception
93        * */
94       for (char c :characters) {
95          try {
96             if (Character.isLetter(c)) throw new BigNumbersException(c + " Not a
   valid number!");
97          }
98          catch (BigNumbersException e) {
99             System.out.println("\n There seems to be a problem:  "+ "This is not a
   valid "+ c + e.getMessage());
100         }
```

```
101       }
102
103       insert(characters);
104
105    }; // constructor
106
107    //part of the linked Listed when creating the lists with the values
108    public LinkBNum(char n) {
109
110       num = (int) n - 48; // Subtract 48 because of assiic value
111
112    }
113
114    /** This constructor produces a String of the form: */
115
116    public LinkBNum(String s) {
117
118       s = s.trim(); // Trimming the white spaces of the string
119       char[] characters = String.valueOf(s).toCharArray();
120
121       /*Check to see if the strings is a valid number
122        * If it not a valid number then throw a run time exception
123        * */
124       for (char c :characters) {
125          try {
126          if (Character.isLetter(c)) throw new BigNumbersException(c + " Not a
   valid number!");
127          }
128          catch (BigNumbersException e) {
129             System.out.println("\n There seems to be a problem:  "+ "This is not a
   valid, "+ c +"! "+ e.getMessage());
130          }
131       }
132
```

```
133      insert(characters);
134
135    }; // constructor
136
137    public String BNumValue(BNum n){
138
139      int signOfN = n.getSign(); //Getting the sign of BNum N
140
141      int counter = 0;  //Counter for the while loop
142      String valueOfNString = ""; //Variable to hold the numbers of BNum N
143      int lenght = n.getDigit(-1);
144
145        while(counter <= lenght-1){
146          valueOfNString += n.getDigit(counter);
147         counter++;
148         //System.out.println(valueOfNinString );
149      }
150        int  firstCharacterOfN = (int) valueOfNString.charAt(0) -48;
151        valueOfNString =  (firstCharacterOfN*signOfN) +
    valueOfNString.substring(1, valueOfNString.length());
152
153      return valueOfNString;
154    }
155
156    // Create a clone of this
157    public BNum clone() {
158
159      //variable to hold the thisNumberValue in string form
160      String thisNumberValue = "";
161      int getSign = this.getSign();
162
163      LinkBNum referenceLink = this.head; // reference pointer to head
164
165      //looping through the linkedlisted to get the values
```

```
166       while (this.head != null) {
167          thisNumberValue += this.head.num;
168          this.head = this.head.nextLink;
169       }
170       this.head = referenceLink; // setting the pointer back to the head
171
172       //casting the first character to an int and subtracting 48 because of the
    ASSIIC value
173       int  firstCharacter = (int)thisNumberValue.charAt(0) -48;
174
175       //Multiplying the first part of the string by the sign
176       thisNumberValue =  (firstCharacter*getSign) +
    thisNumberValue.substring(1, thisNumberValue.length());
177
178       //creating a new object with that same value
179       BNum cloneLink = new LinkBNum(thisNumberValue);
180       return cloneLink;
181    }
182
183    /* Returns true if 'this' = n */
184    public boolean equals(BNum n) {
185
186    boolean equals = false;
187
188       String valueOfNinString = BNumValue(n); //Getting the value of BNum
189       LinkBNum copyingBNum =  new LinkBNum(valueOfNinString); //creating
    a copy of the BNum n that's being passed in
190       int lengthOfBNum =  copyingBNum.lenghtOfNumber;//getting the length
    of BNum that's being passed in
191
192       /*If the length are equal then
193       loop through this check to see if the values are equal*/
194       for(int i = 0; i < this.lenghtOfNumber; i++){
195             if (lengthOfBNum == this.lenghtOfNumber && this.getDigit(i) ==
```

```
    copyingBNum.getDigit(i)){
196                 equals = true;
197             }else{
198                 equals = false;
199             }
200     }//end for loop
201 
202     return equals; //return the values if it's equal
203 
204 }
205 
206 /* Returns true if 'this' < n */
207 public boolean lessThan(BNum n) {
208 
209     boolean lessThan = false;
210     String copyingThisBNumString = "";
211     String copyingBNumString = "";
212 
213     String valueOfNinString = BNumValue(n); //Getting the value of BNum
214     LinkBNum copyingBNum =  new LinkBNum(valueOfNinString); //creating
    a copy of the BNum n that's being passed in
215 
216     LinkBNum copyingThisBNum =  (LinkBNum)this.clone(); //creating a copy
    of the BNumThis that's calling the method
217 
218     //find the different between the two lengths
219     int diferentLenghtBetweenBothValues=
    Math.abs(Math.abs(copyingThisBNum.lenghtOfNumber) -
    Math.abs(copyingBNum.lenghtOfNumber));
220 
221     //adding the values to the copyingThisBNum string
222     for(int i = 0; i < copyingThisBNum.lenghtOfNumber; i++){
223         copyingThisBNumString  += copyingThisBNum.getDigit(i);
224     }
```

```
225
226       //adding the values to the copyingBNum string
227       for(int i = 0; i < copyingBNum.lenghtOfNumber; i++){
228          copyingBNumString  += copyingBNum.getDigit(i);
229       }
230
231       //check what length is longer, if one of the lengths is longer then pad the
      beginning with leading zeros
232       if(copyingThisBNumString.length() < copyingBNumString.length()){
233       for(int x = diferentLenghtBetweenBothValues-1; x >= 0; x--){
234          copyingThisBNumString = 0 + copyingThisBNumString;
235       }
236       }//end for loop
237
238       //check what length is longer, if one of the lengths is longer then pad the
      beginning with leading zeros
239       if(copyingThisBNumString.length() > copyingBNumString.length()){
240          for(int x = diferentLenghtBetweenBothValues-1; x >= 0; x--){
241             copyingBNumString = 0 + copyingBNumString;
242          }
243          }
244
245       //Comparing both strings to see which one is bigger
246       if(copyingThisBNumString.compareTo( copyingBNumString ) < 0){
247          lessThan = true;
248       }
249
      System.out.println(copyingThisBNumString.compareTo( copyingBNumString )
      );
250
251       return lessThan;
252    }
253
254    /* returns 'this' + n */
```

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

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

```
308   }
309
310      //getting the different between the two values and calling the Math
      absolute value
311      int diferentLenghtBetweenBothValues=
      Math.abs(Math.abs(copyingThisBNum.lenghtOfNumber) -
      Math.abs(copyingBNum.lenghtOfNumber));
312      //System.out.println(diferentLenghtBetweenBothValues);
313      /*This while loop add the numbers together in both Arrays, each element
      at a time
314       * If the smallerLenght is less then zero, then it will break out of the loop
315       *
316       * */
317
318      while (smallerLength > 0 ){
319
320         /*This if statement checks if adding both values together is less then 10,
321          * and BNumLenght is not equal to 0
322          * */
323         if(remainder ==  false && largerLengthConBNum.getDigit(largerIndex) +
      smallerLenghtConBNum.getDigit(smallerLength-1) < 10 && smallerLength !=
      0){
324            addingBothValuesString =
      largerLengthConBNum.getDigit(largerIndex) +
      smallerLenghtConBNum.getDigit(smallerLength-1) +
      addingBothValuesString;
325            smallerLength--;
326            if(largerIndex >0){
327               largerIndex--;
328               }
329            remainder = false;
330         }
331
332         /*This if statement checks if adding both values together plus a 1 is less
```

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

```
355             remainder = true;
356         }//end if
357 
358         /*This if statement checks if adding both values together  plus the
    remainder is greater then 10,
359          * and BNumLenght is not equal to 0
360          * */
361         if(remainder ==  true && largerLengthConBNum.getDigit(largerIndex) +
    n.getDigit(smallerLength-1) >= 10 && smallerLength !=  0){
362             addingBothValuesString =
    (((largerLengthConBNum.getDigit(largerIndex) +
    smallerLenghtConBNum.getDigit(smallerLength-1))+1) %10) +
    addingBothValuesString;
363 
364             smallerLength--;
365             if(largerIndex >0){
366                 largerIndex--;
367                 }
368             remainder = true;
369         }
370 
371         if(largerIndex == 0 && remainder == true && smallerLength !=  0){
372             addingBothValuesString = (((largerLengthConBNum.getDigit(0)+
    smallerLenghtConBNum.getDigit(0))%10)+1) +  addingBothValuesString;
373             addingBothValuesString =1 + addingBothValuesString;
374             break;
375         }
376 
377         if(largerIndex == 0 && remainder == false && smallerLength !=  0){
378                 addingBothValuesString = (((largerLengthConBNum.getDigit(0) +
    smallerLenghtConBNum.getDigit(0)))) + addingBothValuesString;
379             break;
380         }
381         //end if
```

```
382
383   }//while  loop
384
385       //System.out.println(remainder);
386
387       /*adding the rest of the number if the BNumLeght is smaller then the
    this.Lenght;
388        * This for loop check if there's carrying over, that needs to been adding
389        *
390        * */
391          for(int x = diferentLenghtBetweenBothValues-1; x >= 0; x--){
392             if(remainder ==  true  &&  (largerLengthConBNum.getDigit(x) +1) >=
    10){
393                 addingBothValuesString = (largerLengthConBNum.getDigit(x)
    +1)%10 +  addingBothValuesString;
394
395                 //adding the very last digit to the end of the string
396                if(x == 0){
397                   addingBothValuesString = 1 +  addingBothValuesString;
398                }
399                 remainder =  true;
400             }//end if
401              if(remainder ==  true  &&  (largerLengthConBNum.getDigit(x) +1) <
    10){
402                   addingBothValuesString = (largerLengthConBNum.getDigit(x)
    +1) +  addingBothValuesString;
403                    remainder =  false;
404           }if(remainder ==  false){
405                addingBothValuesString = (largerLengthConBNum.getDigit(x)) +
    addingBothValuesString;
406             }//end else
407
408          }//end for loop
409
```

```
410         //casting the first character to an int and subtracting 48 because of the
    ASSIIC value
411         int  firstCharacter = (int)addingBothValuesString.charAt(0) -48;
412
413         //Multiplying the first part of the string by the sign
414          addingBothValuesString =  (firstCharacter*signofThisBNum) +
    addingBothValuesString.substring(1, addingBothValuesString.length());
415
416        //System.out.println( "addingBothValuesString " +
    addingBothValuesString);
417
418       //creating a new object with the new values and returning it
419
420      }/**********end of  if very top *********/
421
422      else{
423          //else if the sign match up then call the subtraction method
424           LinkBNum addingConBNum =  (LinkBNum)
    copyingThisBNum.sub(copyingBNum);
425
426          for(int i = 0; i < addingConBNum.lenghtOfNumber; i++){
427             addingBothValuesString += addingConBNum.getDigit(i);
428          }
429
430      }
431      //creating a new LinkBNum object and passing it back
432      LinkBNum addingConBNum = new LinkBNum(addingBothValuesString);
433
434
435      return addingConBNum;
436   }
437
438   /* returns 'this' - n */
439   public BNum sub(BNum n) {
```

```
440
441       String subtractingBothValuesString = ""; //String to adding all the digit
      together
442       boolean regrouping = false; //Setting the remainder to false by default
443
444       String valueOfNinString = BNumValue(n); //Getting the value of BNum
445       LinkBNum copyingBNum =  new LinkBNum(valueOfNinString); //creating
      a copy of the BNum n that's being passed in
446
447       LinkBNum copyingThisBNum =  (LinkBNum)this.clone(); //creating a copy
      of the BNumThis that's calling the method
448
449       /*Getting the sign of both values before deciding to add or subtract
450        * if the sign are the same then add, if they are different the substact */
451       int signOfBNum = copyingBNum.getSign();
452       int signofThisBNum = copyingThisBNum.getSign();
453
454       if(signOfBNum == signofThisBNum){
455
456       ////////////////////////////////////////////////////////////////////////
457       int  smallerLength = 0; // variable to hold the smaller length
458       int largerIndex = 0;  // variable to hold the bigger length
459       LinkBNum  largerLengthConBNum = null;
460       LinkBNum smallerLenghtConBNum = null;
461
462       /*Find out which BNum length is bigger
463        * length is less or greater then*/
464       if(copyingThisBNum.lenghtOfNumber <  copyingBNum.lenghtOfNumber)
      {
465             smallerLength = copyingThisBNum.lenghtOfNumber;  // getting the
      lenght of the value
466             largerLengthConBNum = copyingBNum;
467             smallerLenghtConBNum = copyingThisBNum;
468             largerIndex  = largerLengthConBNum.lenghtOfNumber-1;
```

```
469      }
470      /*Find out which BNum length is bigger
471       * length is less or greater then*/
472      if(copyingThisBNum.lenghtOfNumber >  copyingBNum.lenghtOfNumber)
    {
473              smallerLength =  copyingBNum.lenghtOfNumber;
474              largerLengthConBNum = copyingThisBNum;
475              smallerLenghtConBNum = copyingBNum;
476              largerIndex  = largerLengthConBNum.lenghtOfNumber-1;
477      }
478      /*Find out which BNum length is bigger
479       * length is less or greater then*/
480      if(copyingThisBNum.lenghtOfNumber ==
    copyingBNum.lenghtOfNumber && copyingThisBNum.getDigit(0) <
    copyingBNum.getDigit(0)){
481              smallerLength = copyingThisBNum.lenghtOfNumber;  // getting the
    lenght of the value
482              largerLengthConBNum = copyingBNum;
483              smallerLenghtConBNum = copyingThisBNum;
484              largerIndex  = largerLengthConBNum.lenghtOfNumber-1;
485    }
486      /*Find out which BNum length is bigger
487       * length is less or greater then*/
488      if(copyingThisBNum.lenghtOfNumber ==
    copyingBNum.lenghtOfNumber && copyingThisBNum.getDigit(0) >=
    copyingBNum.getDigit(0)){
489          smallerLength =  copyingBNum.lenghtOfNumber;
490          largerLengthConBNum = copyingThisBNum;
491          smallerLenghtConBNum = copyingBNum;
492          largerIndex  = largerLengthConBNum.lenghtOfNumber-1;
493    }
494
495      //getting the different between the two values and calling the Math
    absolute value
```

```
496      int diferentLenghtBetweenBothValues=
    Math.abs(Math.abs(copyingThisBNum.lenghtOfNumber) -
    Math.abs(copyingBNum.lenghtOfNumber));
497
498      /*This while loop add the numbers together in both Arrays, each element
    at a time
499       * If the smallerLenght is less then zero, then it will break out of the loop
500       *
501       * */
502
503      while (smallerLength > 0 ){
504
505         /*This if statement checks if adding both values together is less then 10,
506          * and BNumLenght is not equal to 0
507          * */
508         //System.out.println(largerLengthConBNum.num[largerIndex]);
509         if(regrouping ==  false && largerLengthConBNum.getDigit(largerIndex)
    - smallerLenghtConBNum.getDigit(smallerLength-1) < 0 && smallerLength !=
    0){
510            subtractingBothValuesString =
    (largerLengthConBNum.getDigit(largerIndex)+10) -
    smallerLenghtConBNum.getDigit(smallerLength-1) +
    subtractingBothValuesString;
511            smallerLength--;
512            if(largerIndex >0){
513            largerIndex--;
514            }
515            regrouping = true;
516         }
517
518         if(regrouping ==  true && largerLengthConBNum.getDigit(largerIndex) -
    smallerLenghtConBNum.getDigit(smallerLength-1) < 0 && smallerLength !=  0)
    {
519            subtractingBothValuesString =
```

```
    (((largerLengthConBNum.getDigit(largerIndex)-1)+10) -
    smallerLenghtConBNum.getDigit(smallerLength-1)) +
    subtractingBothValuesString;
520
521             smallerLength--;
522             if(largerIndex >0){
523                largerIndex--;
524                }
525             regrouping = true;
526          }
527
528          if(regrouping ==  true && largerLengthConBNum.getDigit(largerIndex) -
    smallerLenghtConBNum.getDigit(smallerLength-1) >=  0 && smallerLength !=
    0){
529             subtractingBothValuesString =
    (((largerLengthConBNum.getDigit(largerIndex)-1)) -
    smallerLenghtConBNum.getDigit(smallerLength-1)) +
    subtractingBothValuesString;
530
531             smallerLength--;
532             if(largerIndex >0){
533                largerIndex--;
534                }
535             regrouping = false;
536          }
537
538          if(regrouping ==  false && largerLengthConBNum.getDigit(largerIndex)-
    smallerLenghtConBNum.getDigit(smallerLength-1) >=  0 && smallerLength !=
    0){
539             subtractingBothValuesString =
    (((largerLengthConBNum.getDigit(largerIndex))) -
    smallerLenghtConBNum.getDigit(smallerLength-1)) +
    subtractingBothValuesString;
540
```

```
541             smallerLength--;
542             if(largerIndex >0){
543                largerIndex--;
544                }
545             regrouping = false;
546          }
547
548    }//while  loop
549
550       /*adding the rest of the number if the BNumLeght is smaller then the
    this.Lenght;
551        * This for loop check if there's carrying over, that needs to been adding
552        *
553        * */
554       for(int x = diferentLenghtBetweenBothValues-1; x >= 0; x--){
555             if(regrouping ==  true  &&  (largerLengthConBNum.getDigit(x) - 1) < 0)
    {
556                subtractingBothValuesString = ((largerLengthConBNum.getDigit(x)
    -1)+10) +  subtractingBothValuesString;
557
558                //adding the very last digit to the end of the string
559                if(x == 0){
560                    subtractingBothValuesString =
    ((largerLengthConBNum.getDigit(x) -1)) +  subtractingBothValuesString;
561                }
562                regrouping =  true;
563             }//end if
564              if(regrouping ==  true  &&  (largerLengthConBNum.getDigit(x) - 1) >
    0){
565                 subtractingBothValuesString = (largerLengthConBNum.getDigit(x)
    -1) +  subtractingBothValuesString;
566                 regrouping =  false;
567             }
568       //check if the second last digit is not zero so it doesn't take away from the
```

```
    first and add that at the end
569      if(regrouping ==  false && largerLengthConBNum.getDigit(x) != 0){
570          subtractingBothValuesString = (largerLengthConBNum.getDigit(x)) +
    subtractingBothValuesString;
571      }//end else
572
573        }//end for loop
574
575
576      //casting the first character to an int and subtracting 48 because of the
    ASSIIC value
577      int  firstCharacter = (int)subtractingBothValuesString.charAt(0) -48;
578
579      //Multiplying the first part of the string by the sign
580      subtractingBothValuesString =  (firstCharacter*signofThisBNum) +
    subtractingBothValuesString.substring(1,
    subtractingBothValuesString.length());
581
582        //System.out.println( "subtractingBothValuesString " +
    subtractingBothValuesString);
583
584      }/*end of the long if statement */
585      else{
586        //else if the sign match up then call the addition method
587        LinkBNum addingConBNum =  (LinkBNum)
    copyingThisBNum.add(copyingBNum);
588
589        for(int i = 0; i < addingConBNum.lenghtOfNumber; i++){
590          subtractingBothValuesString += addingConBNum.getDigit(i);
591        }//end for loop
592
593      }//end else
594
595        //creating a new object with the new values and returning it
```

```
596
597       LinkBNum subtractingConBNum = new
      LinkBNum(subtractingBothValuesString);
598
599       return subtractingConBNum;
600
601    }
602
603    /* Returns the sign of this BigNumber object */
604    public int getSign() {
605       //variable to hold the sign
606       int signOfNumber = 0;
607
608       //if sign is positive
609       if(this.sign == 1){
610          signOfNumber = 1;
611       }
612       //if sign is negative
613       if(this.sign == -1){
614          signOfNumber = -1;
615       }
616       //return the value of this sign
617       return signOfNumber;
618    }
619
620    // Returns the digit i of this object, digit 0 is LSD.
621    public int getDigit(int i) {
622
623       int toDigit = 0;
624       int counter = 0;
625
626       //if it's -1 then return the length
627       if(i == -1){
628          return this.lenghtOfNumber;
```

```
629       }
630
631        LinkBNum referenceLink = this.head; // reference pointer to head
632
633       try {
634
635          //while (this.head != null && counter <= i) {
636           while (counter <= i) {
637              //System.out.print(this.head.num);
638              toDigit = this.head.num;
639              this.head = this.head.nextLink;
640              counter++;
641          }//end while
642          this.head = referenceLink; // return the reference pointer to head
643
644       } catch (NullPointerException e) {
645          this.head = referenceLink; // return the reference pointer to head if the
    Exception has been called
646          // System.out.print("Early Termination \nReason:\t" +
647          // mc.getMessage());
648          //System.out.println("Linklisted out of bounds!");
649          return -1;
650       }
651
652       return toDigit;
653     }; // constructor
654
655 }
```