Homework Turnin

Name:	Xuqing Wu
Account:	xw88 (xw88@uw.edu)
Student ID:	1933202
Section:	AD
Course:	CSE 143 20wi
Assignment:	a1
Receipt ID:	323924cba69fd114fabda7d047d5cedf

Turnin script completed with output:

Turnin Successful!

The following file(s) were received:

LetterInventory.java (4286 bytes, sha256: bff962b2daaac011c3adb64bd92ec0a1)

```
1.  // Xuqing Wu
2.  // 1/14/2020
3.  // CSE143
4.  // TA: Eric Fan
5.  // Assignment #1
6.  //
7.  // Class LetterInventory can be used to keep track of an inventory of letters of the alphabet.
8.
9.  public class LetterInventory {
10.    private int[] elementData;  //inventory of letters
11.    private int size;  //current number of alphabetic letters in the list
12.    public static final int NUM = 26;
13.
14.    //post: Constructs an inventory of the alphabetic letters in the given string,
15.    //ignoring the case of letters and ignoring any non-alphabetic characters.
16.    public LetterInventory(String data) {
17.       elementData = new int[NUM];
18.       data = data.toLowerCase();
19.       for(int i = 0; i < data.length(); i++) {
20.          char ch = data.charAt(i);
21.          if(Character.isLetter(ch)) {
22.             elementData[ch - 97]++;
23.             size++;
24.          }
25.       }
26.    }
27.
28.    //pre: alphabetic character is passed(throw IllegalArgumentException if not)
29.    //post: Returns a count of how many of the letter given as
30.    //      parameter are in the inventory.
31.    public int get(char letter) {
32.       checkAlphabetic(letter);
33.       letter = Character.toLowerCase(letter);
34.       return elementData[letter - 97];
35.    }
36.
37.    //pre: check that the given character is alphabetic,
38.    //     throw IllegalArgumentException if not.
39.    public void checkAlphabetic(char letter) {
40.       if(!Character.isLetter(letter)){
41.          throw new IllegalArgumentException(letter + "is not letter");
```

```
42.       }
43.    }
44.
45.    //pre: alphabetic character is passed(throw IllegalArgumentException if not)
46.    //     value >= 0(throw an IllegalArgumentException if not)
47.    //post: Sets the count for the given letter to the given value.
48.    public void set(char letter, int value) {
49.       checkAlphabetic(letter);
50.       checkValue(value);
51.       letter = Character.toLowerCase(letter);
52.       size = size - elementData[letter - 97] + value;
53.       elementData[letter - 97] = value;
54.    }
55.
56.    //pre: check that the value given is non negative
57.    //     (throw IllegalArgumentException if not)
58.    public void checkValue(int value) {
59.       if(value < 0){
60.          throw new IllegalArgumentException("value: " + value);
61.       }
62.    }
63.
64.    //post: Returns the sum of all of the counts in this inventory
65.    public int size() {
66.       return size;
67.    }
68.
69.    //post: Returns true if this inventory is empty (all counts are 0)
70.    public boolean isEmpty() {
71.       return size == 0;
72.    }
73.
74.    //post: Returns a String representation of the inventory with the letters
75.    //      all in lowercase and in sorted order and surrounded by square brackets.
76.    public String toString() {
77.       String result = "[";
78.       for(int i = 0; i < NUM; i++) {
79.          for(int j = 0; j < elementData[i]; j++) {
80.             char grade = (char) ('a' + i);
81.             result += grade;
82.          }
83.       }
84.       result += "]";
85.       return result;
86.    }
87.
88.    //post: Constructs and returns a new LetterInventory object
89.    //      that represents the sum of this letter inventory and the other
90.    //      given LetterInventory.
91.    public LetterInventory add(LetterInventory other) {
92.       LetterInventory added = new LetterInventory("");
93.       for(int i = 0; i < NUM; i++){
94.          added.elementData[i] = this.elementData[i] + other.elementData[i];
95.       }
96.       added.size = this.size + other.size;
97.       return added;
98.    }
99.
100.   //post: Constructs and returns a new LetterInventory object
101.   //      that represents the result of subtracting the other
102.   //      inventory from this inventory. If any resulting count
103.   //      is negative, return null.
104.   public LetterInventory subtract(LetterInventory other) {
105.      LetterInventory subtracted = new LetterInventory("");
106.      for(int i = 0; i < NUM; i++){
107.         if(this.elementData[i] < other.elementData[i]){
108.            size = 0;
109.            return null;
110.         }
111.         else{
112.            subtracted.elementData[i] = this.elementData[i] - other.elementData[i];
113.            subtracted.size = this.size - other.size;
114.         }
115.      }
116.      return subtracted;
117.   }
```

```
118. }
```