Homework Turnin

Name:	Ameya Singh
Account:	ameyas (ameyas@uw.edu)
Student ID:	1868457
Section:	AQ
Course:	CSE 143 18au
Assignment:	a1
Receipt ID:	b5a5b50cd3021840f11cf74c9e4c4b9a

Turnin script completed with output:

Turnin Successful!

The following file(s) were received:

LetterInventory.java (5058 bytes)

```
1. /**
2.  * LetterInventory represents the count of each letter of the alphabet
3.  * within a specified input string.
4.  *
5.  * @author Ameya Singh, CSE143 A, TA: Soham P.
6.  */
7. public class LetterInventory {
8.     public static final int LETTER_COUNTS_ARRAY_SIZE = 26;
9.
10.     private int[] letterCounts;  // Holds counters for all alphabets
11.     private int size;            // Current size of inventory
12.
13.     /**
14.      * Constructs a new letter inventory using the provided string
15.      *
16.      * @param data Input String whose characters will be inventoried
17.      */
18.     public LetterInventory(String data) {
19.         letterCounts = new int[LETTER_COUNTS_ARRAY_SIZE];
20.         size = 0;
21.
22.         setLetterCounts(data);
23.     }
24.
25.     /**
26.      * Helper method: Inventories the provided String
27.      *
28.      * @param data String to be inventoried
29.      */
30.     private void setLetterCounts(String data) {
31.         data = data.toLowerCase();
32.         char[] dataArr = data.toCharArray();
33.         for (char c : dataArr) {
34.             if (Character.isAlphabetic(c)) {
35.                 int index = (int) c - 'a';
36.                 letterCounts[index] = letterCounts[index] + 1;
37.                 size++;
38.             }
39.         }
40.     }
41.
42.     /**
43.      * Gets the current count of passed character in the inventory
44.      *
45.      * @param letter Alphabetic character whose count to return
46.      * @return Count of 'letter' in inventory
```

```java
     * @throws IllegalArgumentException if non-alphabetic letter passed
     */
    public int get(char letter) {
        letter = Character.toLowerCase(letter);
        checkCharInput(letter);

        int index = (int) letter - 'a';
        return letterCounts[index];
    }


    /**
     * Sets the count of passed character in the inventory
     *
     * @param letter Alphabetic character whose count is to be set
     * @param value  Positive integer value to set count of 'letter' to
     * @throws IllegalArgumentException if non-alphabetic letter passed
     */
    public void set(char letter, int value) {
        letter = Character.toLowerCase(letter);
        checkCharInput(letter);
        if (value < 0)
            throw new IllegalArgumentException();

        int index = (int) letter - 'a';
        int before = letterCounts[index];
        letterCounts[index] = value;
        int delta = value - before;
        size = size + delta;
    }

    /**
     * Private Helper: Checks if passed char is valid
     *
     * @param c char to check
     * @throws IllegalArgumentException Thrown if char is not valid
     */
    private void checkCharInput(char c) {
        if (!Character.isAlphabetic(c)) {
            throw new IllegalArgumentException();
        }
    }

    /**
     * Gets the current size of the inventory
     *
     * @return Size of the LetterInventory
     */
    public int size() {
        return size;
    }

    /**
     * Returns whether the inventory is currently empty
     *
     * @return Returns true if the LetterInventory is empty
     */
    public boolean isEmpty() {
        return (size == 0);
    }

    /**
     * Creates a alphabetic list of the letters in the inventory
     * Repeats the letter for each occurrence in the inventory
     *
     * @return Square bracketed String of letters in inventory
     */
    @Override
    public String toString() {
        StringBuilder out = new StringBuilder();
        out.append("[");
        for (int i = 0; i < LETTER_COUNTS_ARRAY_SIZE; i++) {
            for (int j = 0; j < letterCounts[i]; j++) {
                out.append((char) ('a' + i));
            }
        }
        out.append("]");
        return out.toString();
    }

    /**
```

```
128.      * Returns a LetterInventory with the sum of this inventory and the
129.      * passed in inventory
130.      *
131.      * @param other LetterInventory to be summed with current inventory
132.      * @return LetterInventory of the sum of this and other
133.      */
134.     public LetterInventory add(LetterInventory other) {
135.         LetterInventory newInv = new LetterInventory("");
136.         for (int i = 0; i < LETTER_COUNTS_ARRAY_SIZE; i++) {
137.             int sum = other.letterCounts[i] + this.letterCounts[i];
138.             newInv.letterCounts[i] = sum;
139.             newInv.size += sum;
140.         }
141.         return newInv;
142.     }
143.
144.     /**
145.      * Returns a LetterInventory resultant of the subtraction of the passed
146.      * inventory with this inventory
147.      * Returns null if the subtraction cannot be completed
148.      *
149.      * @param other LetterInventory to be subtracted from current inventory
150.      * @return LetterInventory of result of subtraction, null if subtraction
151.      * fails
152.      */
153.     public LetterInventory subtract(LetterInventory other) {
154.         LetterInventory newInv = new LetterInventory("");
155.         for (int i = 0; i < LETTER_COUNTS_ARRAY_SIZE; i++) {
156.             int difference = this.letterCounts[i] - other.letterCounts[i];
157.             if (difference < 0) {
158.                 return null;
159.             }
160.             newInv.letterCounts[i] = difference;
161.             newInv.size += difference;
162.         }
163.         return newInv;
164.     }
165. }
166.
```