CSE 143 Assignment 1 (LetterInventory) Score Sheet

Student(s): akshit <akshit@uw.edu>
Graded by: Melissa Medsker <medskm@cs.washington.edu>

29 / 30 : Total Score

29 / 30 : Correctness

3 / 3 : constructor works
1 / 1 : empty constructor
1 / 1 : nonempty constructor works sometimes
1 / 1 : nonempty constructor works for all cases

2 / 2 : set works
1 / 1 : set attempt
1 / 1 : set works in all cases

3 / 3 : add works
1 / 1 : add attempt
1 / 1 : add works perfectly in all cases
1 / 1 : does not modify original LetterInventory

5 / 5 : subtract works
2 / 2 : subtract attempt for non-null cases
1 / 1 : subtract attempt for null cases
1 / 1 : subtract works perfectly in all cases
1 / 1 : does not modify original LetterInventory

2 / 2 : getLetterPercentage works
1 / 1 : works in all cases
1 / 1 : decodedCryptogram is correct

1 / 1 : toString works
1 / 1 : isEmpty works (or is consistent w/ size after set)
1 / 1 : get works

2 / 2 : get, set and getLetterPercentage throw proper exceptions
1 / 1 : attempt: at least 1 out of the 3 correctly throws the exceptions
1 / 1 : correct: all 3 throw the proper exceptions

3 / 3 : size field used for fast size()
1 / 1 : removes magic values

2 / 3 : comments
1 / 1 : attempt to comment
1 / 2 : well documented code
-1: See yellow
-0: You should comment somewhere in your program that letter-casing is ignored when setting/constructing/getting letter counts.

3 / 3 : otherwise good style
-0: Lines should be no more than 80 characters long (see 68, 162, 164)

Lateness and Other Deductions

	Thu 2016/10/06 11:30pm	Due
	Wed 2016/10/05 11:17pm	Submitted (on time)
0	Late days used on this assignment	
0	Lateness deduction	
	Other deductions	

Overall comments:
Great work on writing an efficient implementation and handling the different possible edge cases for the LetterInventory constructors and methods! For future assignments, be careful about explicitly mentioning clients and avoiding comments on unnecessary details (such as a precondition that a parameter must be the same as the type in the method header)

Annotations: LetterInventory.java

```java
/**
 * @author Akshit Patel
 * @Date 09/29/2016
 * CSE 143D DC
 * TA: Melissa Medsker
 * HW #1 LetterInventory
 */

/**
 * LetterInventory Class creates, updates & modifies an inventory of the no. of
 * alphabets in a string passed and provides useful methods to get its size,
 * adding or subtracting or returning a useful string of the inventory.
 */
public class LetterInventory {

    public static final int INVENTORY_SIZE = 26;// fixed size of inventory.
    private int[] inventory;// Reference to the letter inventory.
    private int size;// keeps check of the total alphabets in inventory.

    /**
     * This Constructor creates an empty inventory for the client of the fixed
     * size.
     */
    public LetterInventory() {
        this("");// create empty inventory.
    }

    /**
     * This constructor creates an inventory for the client based of the string
     * passed by client.
     *
     * preCondition: parameter passed is a string.
     *
     * @param data is a string to be passed to be processed in inventory.
     */
    public LetterInventory(String data) {
        this.inventory = new int[INVENTORY_SIZE];// create empty inventory.
        data = data.toLowerCase();// converts the string into lower case.
        // for-loop to process the string alphabets into inventory.
        for (int i = 0; i < data.length(); i++) {
            char currentChar = data.charAt(i);
            // if statement to check for only alphabets.
            if (currentChar >= 'a'
                    && currentChar <= ('a' + INVENTORY_SIZE - 1)) {
                this.inventory[currentChar - 'a']++;// adding to inventory
                this.size++;// updating the new size.
            }
        }
    }

    /**
     * This method helps the client to get data from inventory of the specific
     * letter.
     *
     * preCondition: the letter is an alphabet.
     *
     * @param letter Character whose info is needed.
     * @throws IllegalArgumentException when letter is not an alphabet.
     * @return int value of the letter in inventory.
     */
    public int get(char letter) {
        letter = Character.toLowerCase(letter);
        this.errorCheck(letter);
        return this.inventory[letter - 'a'];
    }

    /**
     * This method helps the client to set a specific value of the letter in the
     * inventory.
     *
     * preCondition: the letter is an alphabet & value is non-negative.
     *
     * @param letter Character which is needed to update its associated value.
     * @param value int data of the letter to be updated.
     * @throws IllegalArgumentException when letter is not an alphabet & value
     * is a negative number.
     *
     */
    public void set(char letter, int value) {
        letter = Character.toLowerCase(letter);
        this.errorCheck(letter);
        if (value < 0) {
            throw new IllegalArgumentException("No negative values accepted!");
        }
        int temp = this.inventory[letter - 'a'];
        this.size -= temp;// updates size by removing the value before.
        this.inventory[letter - 'a'] = value;// updates the inventory.
        this.size += value;// updates size by adding the provided value .
    }

    /**
     * This method provides the client with the total size of the inventory
     * which is the sum of the letters in inventory.
     *
     * @return the size of inventory.
     */
    public int size() {
        return this.size;
    }

    /**
     * This methods helps the client to know if the inventory is empty.
     *
     * @return true if this inventory is empty.
     */
    public boolean isEmpty() {
        return size == 0;
    }

    /**
     * This methods helps the client to get a string representation of the
     * inventory.
     *
     * @return String representation of the inventory of all letters in
     * lowercase, sorted and inside square brackets.
     */
    @Override
    public String toString() {
        String sortString = "";// empty string to store the data.
        // for loop to add data from the inventory to the string
        for (int i = 0; i < INVENTORY_SIZE; i++) {
            for (int j = 0; j < inventory[i]; j++) {
                char alphabet = (char) (i + 'a');// getting the alphabet.
                sortString += alphabet;// adding the alphabet to the string.
            }
        }
        return "[" + sortString + "]";// return the updated string.
    }
```

[“Hip”,“Hip”]

Note that an alphabet is defined as a collection of letters - so "letters" would be a better term to use here

-0: Comments should not explicitly mention anything about clients

What does "fixed size" refer to here?

-0: This condition is implicit by the public method header, which requires a String data parameter to be called - you should only comment on what should be expected as a result of passing in the String, not that the parameter must be a String

"in the alphabet" or "alphabetic"

This comment should more clearly specify that the "letter count" of the given character is returned

"in the alphabet" or "alphabetic"

-0: This comment should specify that the "letter count" of the given character is updated

-0: Unnecessary to specify client in comments

-0: This comment should specify that the letters are repeated according to their letter count in the inventory

```java

    /**
     * This method constructs and returns a new LetterInventory object that
     * represents the SUM of this(current) LetterInventory and the other given
     * LetterInventory.
     *
     * preCondition: this & other are LetterInventory object. postCondition:
     * this & other remain the same.
     *
     * @param other LetterInventory object that needs to be added to this
     * LetterInventory.
     * @return LetterInventory object that represents the sum of this and other
     * LetterInventory.
     */
    public LetterInventory add(LetterInventory other) {
        LetterInventory sum = new LetterInventory();// creates empty Inventory.
        // for loop to add the two inventory data in new one.
        for (int i = 0; i < INVENTORY_SIZE; i++) {
            // adding the data.
            sum.inventory[i] = this.inventory[i] + other.inventory[i];
            sum.size += sum.inventory[i];// updating size of the new Inventory.
        }
        return sum;// processed inventory is returned.
    }
```

-0: These add and subtract comments should more clearly specify what is meant by "sum" and "difference" of two LetterInventory objects

```java

    /**
     * This method constructs and returns a new LetterInventory object that
     * represents the DIFFERENCE of this(current) LetterInventory and the other
     * given LetterInventory.
     *
     * preCondition: this & other are LetterInventory object. postCondition:
     * this & other remain the same.
     *
     * @param other LetterInventory object that needs to be subtracted from this
     * LetterInventory.
     * @return LetterInventory object that represents the difference of this and
     * other LetterInventory. Returns null if the difference is negative.
     */
    public LetterInventory subtract(LetterInventory other) {
        LetterInventory remove = new LetterInventory();// empty Inventory made.
        // for loop to get the difference of two inventory data in newer one.
        for (int i = 0; i < INVENTORY_SIZE; i++) {
            // subtracting the data.
            remove.inventory[i] = this.inventory[i] - other.inventory[i];
            // if statement to check for negative values.
            if (remove.inventory[i] < 0) {
                return null;
            }
            remove.size += remove.inventory[i];// updates the size.
        }
        return remove;// processed inventory is returned.
    }

    /**
     * This method helps the client to know the percentage of given letter in
     * the inventory.
     *
     * preCondition: The letter is an alphabet character only.
     *
     * @param letter Character whose info is needed.
     * @throws IllegalArgumentException if letter is not an alphabet.
     * @return a double from 0.0 to 1.0.
     */
    public double getLetterPercentage(char letter) {
        letter = Character.toLowerCase(letter);
        this.errorCheck(letter);
        // if statement to check if the inventory has any letters.
        if (this.isEmpty()) {
            return 0;
        }
        // calculate and return the percent.
        return (double) (this.inventory[letter - 'a']) / size;
    }
```

-0: This comment should specify that the percentage returned is represented as a value between 0.0 and 1.0

```java

    /**
     * This method checks for error in the letter passed by client.
     *
     * @param letter the Character passed by the client.
     * @throws IllegalArgumentException if letter does not meet the
     * preconditions.
     */
    private void errorCheck(char letter) {
        // if statement that checks for letters not in between 'a' to 'z'.
        if (letter < 'a' || letter > ('a' + INVENTORY_SIZE - 1)) {
            throw new IllegalArgumentException("Non-Alphabets not accepted!");
        }
    }
}

```

-1: This comment should specify the cases in which exceptions are thrown (here, the passed character must be alphabetic)