1. Assignment/9

Masato Ito	1. Assignment/9. task	31st March 2023
YUUU2V		
yuuu2v@INF.ELTE.HU		
Group 4		

Task

9. Implement the bag type which contains integers. Represent the bag as a sequence of (element, frequency) pairs. Implement as methods: inserting an element, removing an element, returning the frequency of an element, returning the number of elements which occur only once in the bag (suggestion: store the number of these elements and update it when the bag changes), printing the bag.

Bag type

Set of values

$$Bag(n) = \{a : Item* \in \mathbb{Z}^n \mid \forall i, j[1..|n|] : a[i].d \in \mathbb{Z} \land a[j].d \in \mathbb{Z} \rightarrow a[i].d \neq a[j].d \land a.f > 0\}$$

$$Item :< d : data : \mathbb{Z}, f : frequency : \mathbb{N} >$$

Operations

1. Insert an element

This operation adds a new element to the bag, potentially increasing the frequency count of an existing element if the same item was already present in the bag.

Formality:

$$A = a : Bag(n) \times m : \mathbb{Z} \times elem : \mathbb{Z}$$

$$Pre = (m = m \land elem = elem' \land elem \geqq 0)$$

$$Post = (Pre \land b : Bag \land b \underline{\cup} \{elem\})$$

2. Remove an element

This operation removes one instance of a given element from the bag. If the element occurs multiple times, only one instance will be removed.

Formality:

$$A = a : Bag(n) \times m : \mathbb{Z} \times x : \mathbb{Z}^* \times elem : \mathbb{Z}$$

$$Pre = (m = m' \land x = x' \land |x| \geqq 1 \land \forall i[1..|x|] : x[i] \in [0..m] \land elem \in Bag(m))$$

$$Post = (Pre \land b : Bag \land Remove\{elem\})$$

3. Return the frequency of an element

This operation returns the number of times a specific element appears in the bag.

Formality:

$$A = a : Bag(n) \times m : \mathbb{Z} \times x : \mathbb{Z}^* \times elem : \mathbb{Z}$$

$$Pre = (m = m' \land x = x' \land |x| \geqq 1 \land \forall i[1..|x|] : x[i] \in [0..m])$$

$$Post = (Pre \land b : Bag \land b \overset{|x|}{\underset{i=1}{\underline{\cup}}} \{x[i]\} \land elem = frequent(b))$$

4. Return the number of elements that occur only once in the bag

This operation counts the number of elements that occur exactly once in the bag. This requires iterating through all the elements in the bag and counting those whose frequency count is equal to one.

Formality:

$$A = a : Bag(n) \times m : \mathbb{Z} \times x : \mathbb{Z}^* \times elem : \mathbb{Z}$$

$$Pre = (m = m' \wedge x = x' \wedge |x| \geqq 1 \wedge \forall i[1..|x|] : x[i] \in [0..m])$$

$$Post = (Pre \wedge b : Bag \wedge b \overset{|x|}{\underset{i=1}{=}} \underline{\cup}\{x[i]\} \wedge elem = singleElement(b))$$

5. Printing the bag *

This operation displays the contents of the bag, typically in a formatted way that shows the elements and their respective frequency counts.

Formality:

$$A = a : Bag(n) \times m : \mathbb{Z} \times x : \mathbb{Z}^* \times elem : \mathbb{Z}$$

$$Pre = (m = m' \wedge x = x' \wedge |x| \geqq 1 \wedge \forall i[1..|x|] : x[i] \in [0..m])$$

$$Post = (Pre \wedge b : Bag \wedge b \overset{|x|}{\underset{i=1}{=}} \underline{\cup}\{x[i]\})$$

Representation

- The bag will be represented as a list of tuples, where each tuple contains an element and its frequency in the bag.

$$Bag = vec : Item^*$$

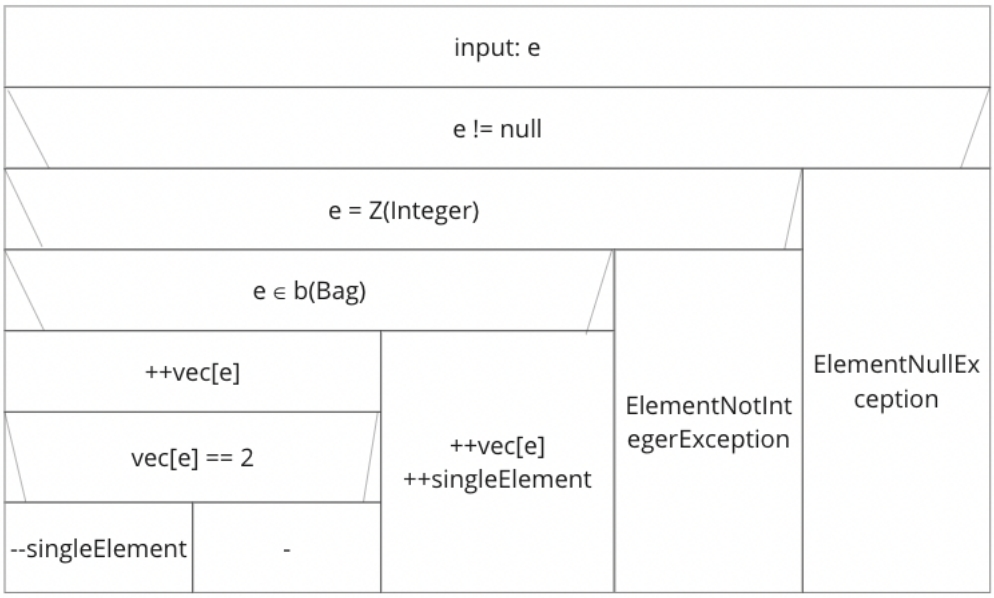
$$Item^* = < (d1, f1), (d2, f2), (d3, f3)... >$$

- The input for the data must be restricted to integer numbers, while the frequency input must only accept positive values. Additionally, the collection mechanism employed prohibits the inclusion of multiple instances of the same element within the data structure.

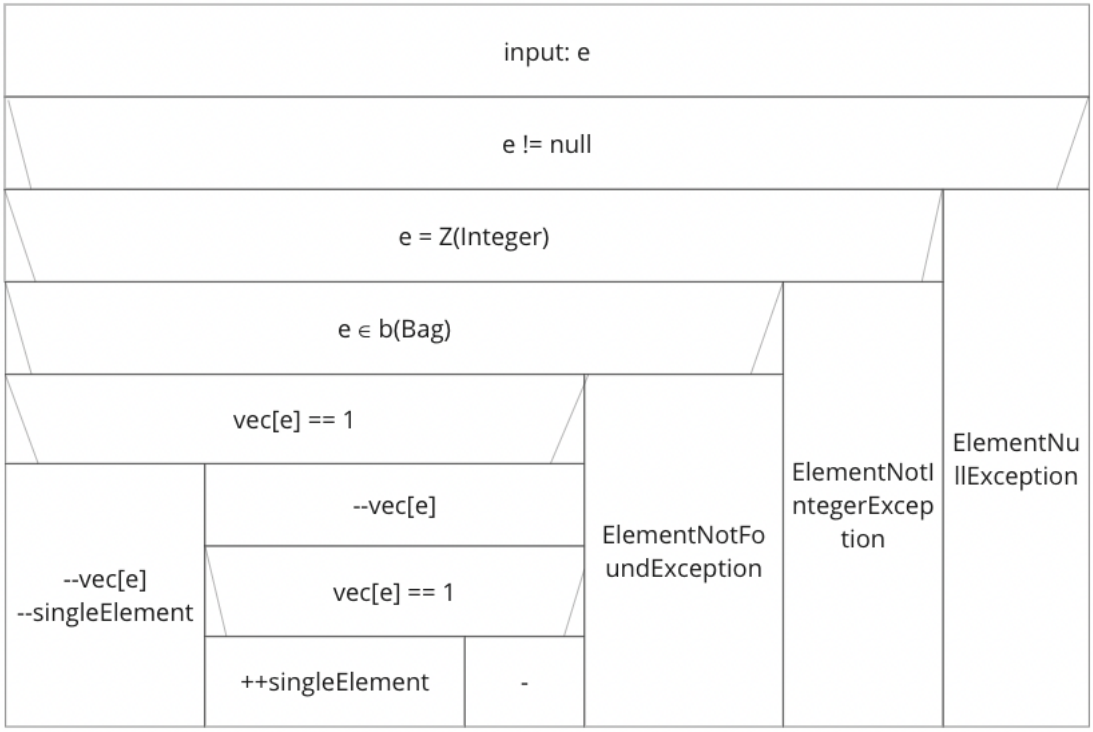
$$Item : < d : data : \mathbb{Z},\ f : frequency : \mathbb{N} > \rightarrow\ \forall i, j[1..|Item.length|] : Item[i].d \in \mathbb{Z} \wedge Item[j].d \in \mathbb{Z} \wedge Item[i]$$

Implementation

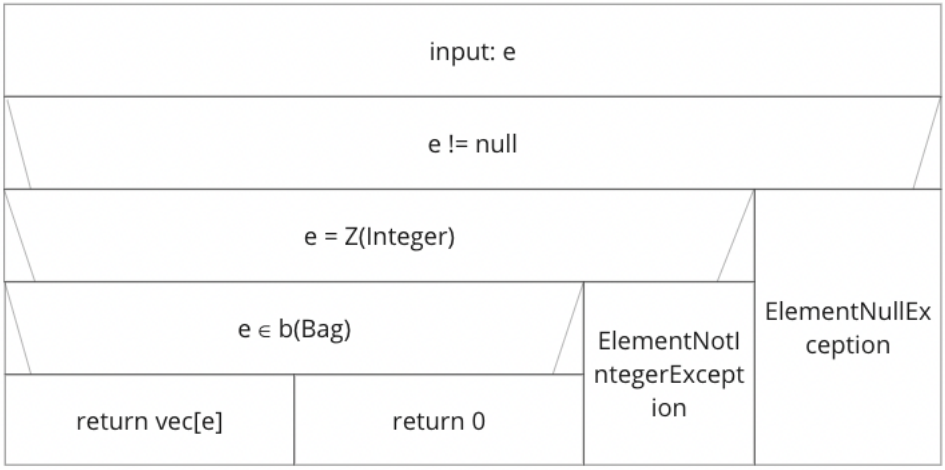
1. Insert an element
    a. Check if the input element is not null.
    b. Convert the input element to an integer using `Int32.TryParse()` method.
    c. If the conversion is successful, search for the index of the element in a private list called `_elements`.
    d. If the element is found in the list, update its count by incrementing by 1 and check if its count is equal to 2. If it is, decrement `_singleElementCount`.
    e. If the element is not found in the list, add it to the `_elements` list with a count of 1 and increment `_singleElementCount`.
    f. If the input element is null, throw an exception.



2. Remove an element
    a. Check if the input element is not null.
    b. If it is not null, try to parse the element to an integer. If parsing fails, throw an ElementNotIntegerException.
    c. Find the index of the element in the _elements list.
    d. If the element is not found, throw an ElementNotFoundException.
    e. If the element count is 1, remove the element from the list and decrement _singleElementCount.
    f. Otherwise, decrement the count of the element in the list and increment _singleElementCount if the count becomes 1.
    g. Return true to indicate that the removal was successful.
    h. If the input element is null, throw an ElementNullException.

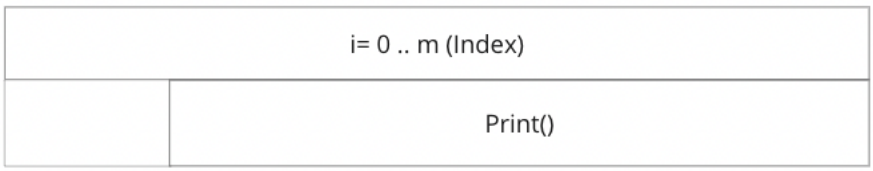


3. Return the frequency of an element
    a. The method takes an optional integer value as its input.
    b. If the input value is not null, the method tries to parse it into an integer.
    c. If the parsing is successful, the method searches for the index of the input value in a private list called `_elements`.
    d. If the input value is found in the list, the method returns the frequency count of the value (the second item in the tuple).
    e. If the input value is not found in the list, the method returns 0.
    f. If the input value is null, the method throws an `ElementNullException`.



4. Return the number of elements that occur only once in the bag
    a. Define a method called "SingleElementCount".
    b. Inside the method, return the private field "_singleElementCount".
    c. End the method.

5. Printing the bag
    a. The `foreach` loop iterates over each element in the `_elements` list.
    b. For each element, it uses string interpolation to print out the element's first and second items (i.e., `Item1` and `Item2`) to the console, separated by a colon and space.
    c. The `Console.WriteLine()` method is used to print out the resulting string to the console.



Testing

Testing the operations (black box testing)

1) Insert the element into the bag

    a) Inserting a non-null element into an empty bag

b) Inserting a non-null element into a non-empty bag

c) Inserting an empty element into an empty bag

d) Inserting an empty element into a non-empty bag

e) Inserting a non-integer element into a non-empty bag

2) Remove the element from the bag

a) Removing an element that exists in the bag

b) Removing an element that does not exist in the bag

c) Removing the last occurrence of an element

d) Removing the only occurrence of an element

e) Removing an empty element

3) Get the frequency of the element in the bag

a) Getting the frequency of an element that exists in the bag

b) Getting the frequency of an element that does not exist in the bag

c) Getting the frequency of the last occurrence of an element

d) Getting the frequency of the only occurrence of an element

e) Getting the frequency of an empty element

4) Get a count of single occurrence elements in the bag

a) Getting the count of single occurrence elements in an empty bag

b) Getting the count of single-occurrence elements in a bag with no single-occurrence elements

c) Getting the count of single-occurrence elements in a bag with multiple single-occurrence elements

d) Getting the count of single occurrence elements in a bag with only one element that occurs more than once

5) Print elements and their frequencies in the bag

a) Printing an empty bag

b) Printing a bag with one element

c) Printing a bag with multiple elements

d) Printing a bag with only single-occurrence elements

e) Printing a bag with no single occurrence elements

Testing based on the code (white box testing)

1. Generating and catching exceptions.

1. Inserting Elements into the Bag:
    - When inserting an element into the bag, the input value must not be null.
    - The input value must be an integer, otherwise, it will not be accepted and will be rejected by the data structure.
2. Removing Elements from the Bag:
    - When removing an element from the bag, the input value must not be null.
    - If the input value is not present in the bag, the data structure will return an error message indicating that the element does not exist in the bag.
3. Returning the Frequency of an Element:
    - When returning the frequency of an element in the bag, the input value must not be null.
    - The input value must be an integer, otherwise, it will not be accepted and will be rejected by the data structure.