OOP. Shokhrukh Nigmatillaev. Documentation of 1st Assignment. Task 10.

Shokhrukh Nigmatillaev **1. Assignment/10 Task.** 17 March 2024
APVAVZ
apvavz@inf.elte.hu
Group 6

Task

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 largest element in the bag (suggestion: store the largest element and update it when the bag changes), printing the bag. Lecture code cannot be submitted.

Bag type.

Set of values

Bag() = {nums ∈ $N^{0..border}$}

Array with integers.

Operations

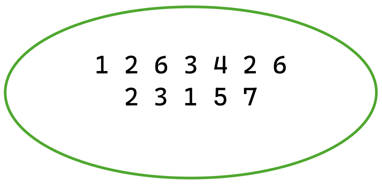
1. **Printing the bag.** The operation will loop through the array. If the value of nums[i] > 0, it will print the value of **i** that is the number in the bag.
   Formally:
   A = (num: $N^{0..border}$)
   Post: (∀i[0 .. border] cond(nums[i] > 0))
2. **Inserting an element.** Inserting an element in the bag that's in interval [0..border].
   Formally:
   A = (nums: $N^{0..border}$, num : N, largestValue : N)
   Pre = (num = num' ^ nums = nums' ^ largestValue = largestValue')
   Post = (Pre ^ num $\underline{\cup}$ {num} cond(num > 0 ^ num ≤ border) ^ cond(num > largestValue, largestValue = num))
   The operation will check for condition and if true will add num into the bag then check if it is bigger than largestValue, if yes it will update it.
3. **Removing an element.** Removing element from the bag that's in interval [0..border].
   Formally:
   A = (nums: $N^{0..border}$, num : N, largestValue : N)
   Pre = (nums = nums' ^ num = num' ^ largestValue = largestValue' ^ |nums| > 0)
   Post = (Pre ^ nums[num] – 1, cond(num > 0 ^ num ≤ border) ^ nums.UpdateLargestValue())

The operation will check for condition for emptyBag, if false it will check for interval condition, it true it will reduce the frequency of given element by 1 and update largest value.

4. **Getting the frequency of given element.**
   Formally:
   A = (nums: $N^{0..border}$, num : N, elem:N)
   Pre = (nums = nums' ^ num = num' ^ |nums| > 0)
   Post = (Pre ^ elem:=frequent(num) cond(|nums|>0))
   The operation will return the frequency of the given number, if it is in the Bag, if Bag is not empty.
5. **Getting the largest element of the bag.**
   Formally:
   A = (nums: $N^{0..border}$, largestValue : N)
   Pre = (nums = nums' ^ largestValue = largestValue' ^ |nums| > 0)
   Post = (Pre ^ largestValue = MAX{|nums|})
   The operation will return the largest value in the bag.

Representation.

Bag:



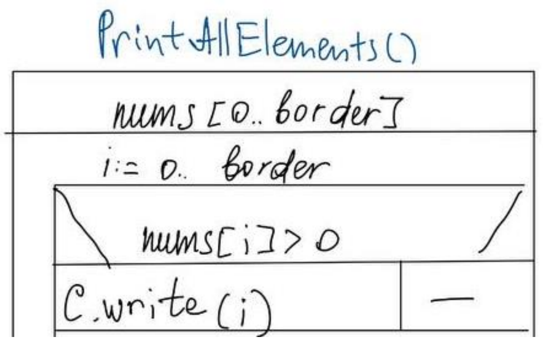
nums:

0	1	2	3	4	5	6	7
0	2	3	2	1	1	2	1

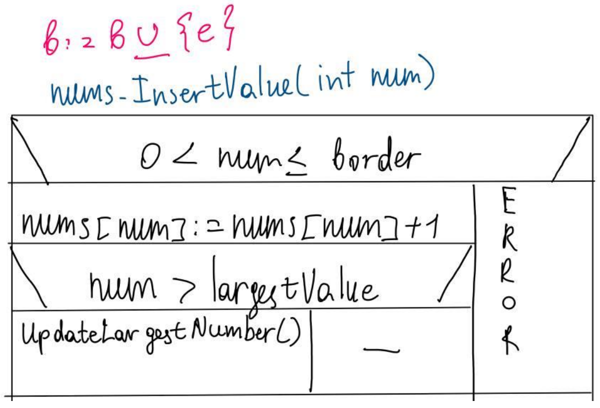
max: 7

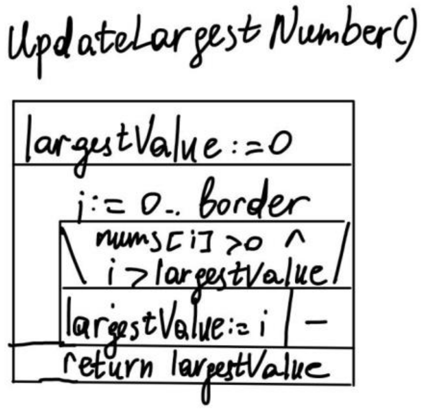
Implementation

1. **PrintAllElements.** Algorithm of the method.

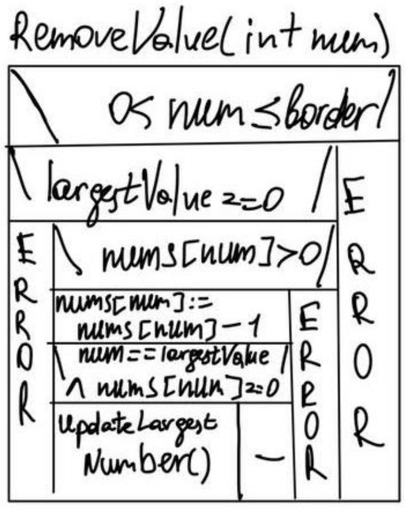


2. **InsertValue.** Implementation of the method:

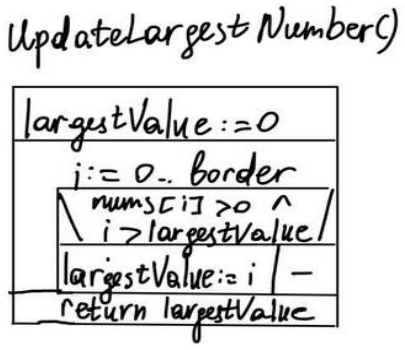




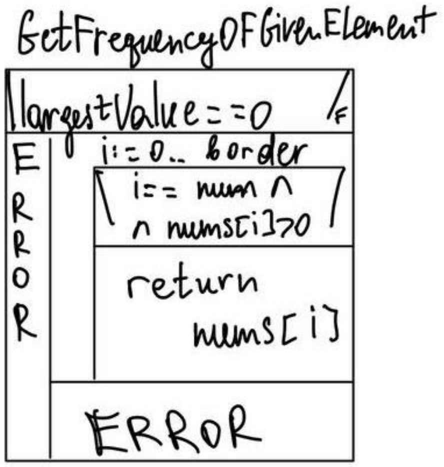
3. **RemoveValue.** Algorithm of the method.



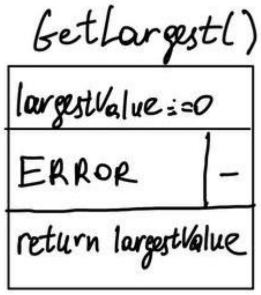
when using RemoveValue method we will also need to update largestValue in case we will need to remove and change it. Therefore we have additional helper method that is implemented:



4. **GetFrequencyOfGivenElement.** Algorithm of the method.



5. **GetLargest.** Algorithm of the method.



Testing.

Testing the operations (black box testing)

1. Testing if inserted value is in the Bag.
   We insert **num** = **4** and check if it is inside the bag.
   1.1 We try to insert invalid number (out of our made range) into the bag. That will cause InvalidEntryException().
   1.2 After insertion we check the frequency of the inserted element. Frequency should increase by 1 per insertion.
   1.3 We insert 2 different nums and check if the larger one becomes the largest in our bag.
2. Testing if the removed num is still in the bag.
   We insert and then remove the same num and with the help of helper function we check whether num is still in or was removed.
   2.1 We try to remove invalid number (out of our made range) from the bag. That will cause and handle InvalidEntryException().
   2.2 After removing we check the frequency of the chosen num and it should be decreased by 1 per deletion.
   2.3 Testing of the update of largestValue after removal of the num. We use helper function to check whether we are having correct largestValue after removal.

3. Testing getting frequency of the given element in the bag. After 2 insertions the answer should be 2 as well.
   3.1 Testing the frequency of the element in the empty bag. Program should cause and handle EmptyBagException.
4. Testing getting the largestValue from the bag. We return the largest value with the help of method and check with the real largestValue.
   4.1 Getting largest number from the empty bag, that should cause and handle EmptyBagException.