OOP documentation, 1st assignment 1. ***17th March, 2024***
Joseph Paul Koyi
Neptun Code: U9PQCO
u9pqco@inf.elte.hu
Group 7

TASK
Implement the set type which contains integers. Represent the set as a sequence of its elements. Implement as methods: inserting an element, removing an element, returning whether the set is empty, returning whether the set contains an element, returning a random element without removing it from the set, returning the number of even numbers in the set (suggestion: store the number of even numbers and update it when the set changes), printing the set. A set can store every element only once.

Integer Set Type

Set of Values

Set = {x: x ∈ ℤ}

Operations:

1. Inserting an element
Method: InsertElement(int x)
This method inserts an integer `x` into the set. If `x` is already in the set, it throws an `ElementAlreadyInSetException`.
If `x` is an even number, it increments the count of even numbers in the set.

Formally:
A = (s : set , x : ℤ)
Pre *= (s = s' ∧ x = x' ∧ x ∉ s)*
$Post = (Pre \wedge s := s \cup \{x\})$

2. Removing an element
Method: `RemoveElement(int x)` This method removes an integer `x` from the set. If `x` is not in the set, it throws an `ElementNotInSetException`. If `x` is an even number, it decrements the count of even numbers in the set.

Formally:
A = (s : set, x : ℤ)
Pre *= (s = s' ∧ x = x' ∧ x ∈ s)*
$Post = (Pre \wedge s := s \text{ remove } \{x\})$

3. Checking if the set is empty
Method: `IsEmpty()`
This method returns `true` if the set is empty, and `false` otherwise.

Formally:
A = (s : set, $\ell : \mathbb{L}$)
Pre *= (s = s')*
$Post = (Pre \wedge \ell)$

4. Checking if the set contains an element
Method: `IsElementInTheSet(int x)`
This method returns `true` if the integer `x` is in the set, and `false` otherwise.

Formally:
A = (s : set, x : $\mathbb{Z}$, $\ell : \mathbb{L}$)
Pre *= (s = s' $\wedge$ x = x')*
$Post = (Pre \wedge \ell)$

5. Returning a random element
Method: `GetRandomElement()`
This method returns a random element from the set without removing it. If the set is empty, it throws an `EmptySetException`.

Formally:
A = (s : set , random_x : $\mathbb{Z}$)
Pre *= (s = s')*
$Post = (Pre \wedge \text{random\_x:= x} \in \text{s})$

6. Returning the number of even numbers
Method: `GetEvenCount()`
This method returns the count of even numbers in the set.

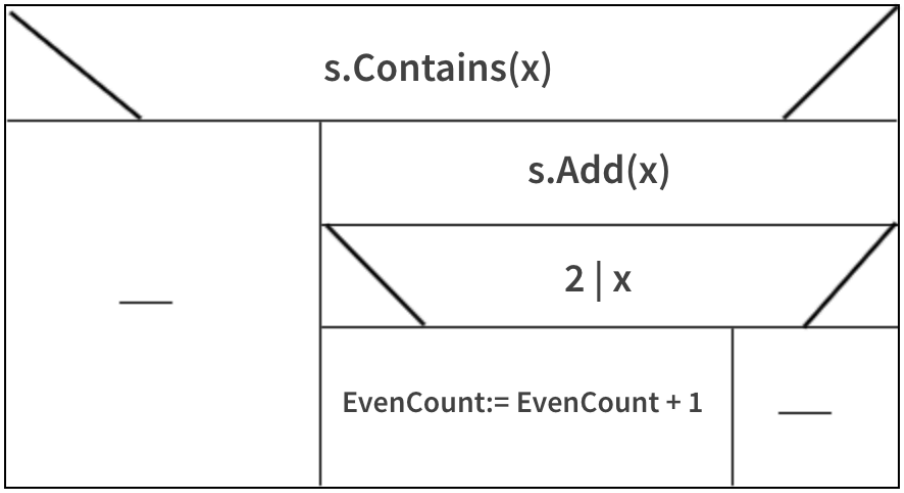
Formally:
A = (s : set)
Pre *= (s = s')*
$$Post = \left(Pre \wedge \text{c:=} \sum_{\substack{0 \\ 2 \mid s[i]}}^{|s|} 1 \right)$$

7. Printing the set
Method: `PrintElements()`
This method prints all the elements in the set.

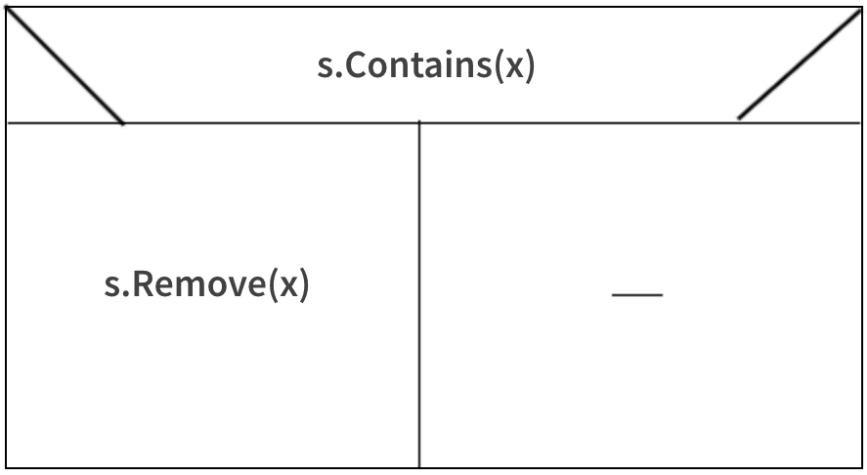
Representation:
The Set class, the set of integers is represented as a List<int>(sequence of integers). This allows for efficient insertion and removal of elements. The List<int> is a dynamic array, meaning it can grow and shrink in size as needed. Also a set is a collection of distinct objects and therefore no duplicates.
The Set class also keeps track of the count of even numbers in the set. This is done using an integer variable evenCount.
Set:
|--- List<int> elements // The list of elements in the set
|--- int evenCount // The count of even numbers in the set

Implementation:

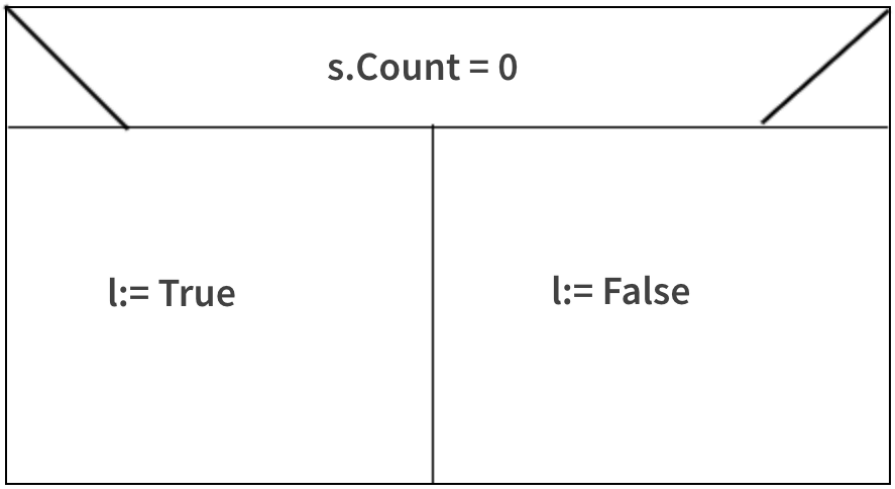
1. InsertElement(int x)
This method first checks if `x` is already in the set by calling `IsElementInTheSet(x)`. If `x` is in the set, it throws an `ElementAlreadyInSetException`. Otherwise, it adds `x` to the `elements` list. If `x` is even, it increments `evenCount`.



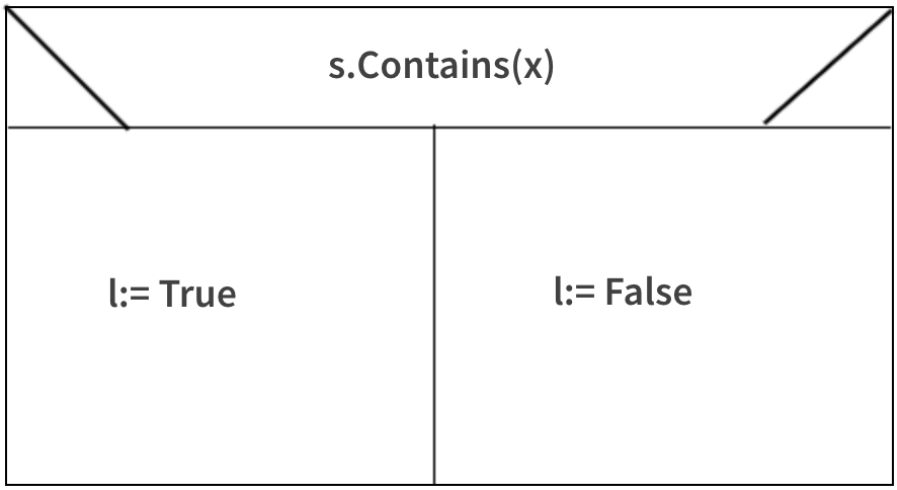
2. RemoveElement(int x)
This method first checks if `x` is in the set by calling `IsElementInTheSet(x)`. If `x` is not in the set, it throws an `ElementNotInSetException`. Otherwise, it removes `x` from the `elements` list. If `x` is even, it decrements `evenCount`.



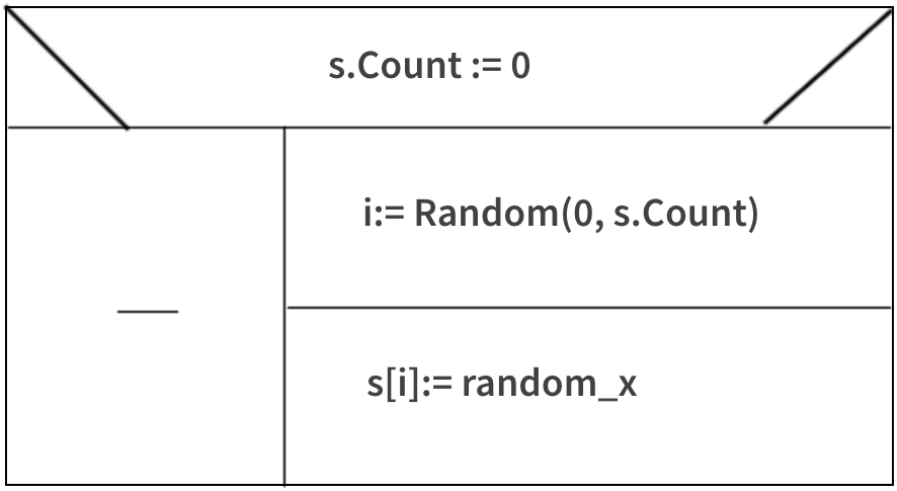
3. IsEmpty()
This method simply returns `true` if the `elements` list is empty, and `false` otherwise.



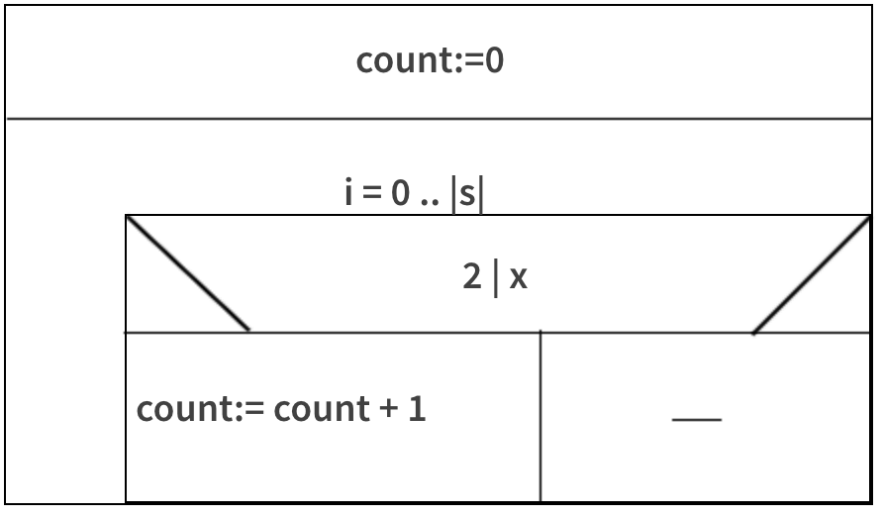
4. IsElementInTheSet(int x)
This method uses the `Contains` method of the `List<int>` class to check if `x` is in the `elements` list. `Contains` performs a linear search on the list.



5. GetRandomElement()
This method first checks if the set is empty by calling `IsEmpty()`. If the set is empty, it throws an `EmptySetException`. Otherwise, it generates a random index into the `elements` list and returns the element at that index.



6. GetEvenCount()
This method simply returns the value of `evenCount`, which is updated whenever an even number is inserted or removed from the set.



7. PrintElements()
This method uses a `foreach` loop to iterate over the `elements` list and print each element. It uses `Console.WriteLine` to print the elements.

Testing

Testing the operations (black box testing)
1) Creating a Set
a) TestSetConstructor: This test checks if a new Set object is not null after it's been instantiated.

2) Inserting an element into the Set
a) TestInsertElement: This test checks if an element can be successfully inserted into the Set.

b) TestInsertElementTwice: This test checks if the program throws an `ElementAlreadyInSetException` when trying to insert an element that is already in the Set.

3) Checking if an element is in the Set
a) TestIsElementInTheSet: This test checks if the `IsElementInTheSet` method correctly identifies whether an element is in the Set.

Testing based on the code (white box testing)
1) Inserting an element into the Set
a) TestInsertElement: This test checks the `InsertElement` method by inserting an element and then using `IsElementInTheSet` to verify that the element was inserted.

b) TestInsertElementTwice: This test checks the `InsertElement` method by inserting an element twice and verifying that an `ElementAlreadyInSetException` is thrown the second time.

2) Checking if an element is in the Set

a) TestIsElementInTheSet: This test checks the `IsElementInTheSet` method by inserting an element and then using `IsElementInTheSet` to verify that the element is in the Set.