Wesley Kwan

CS 2400.02

Project 1

26 February 2020

Section 1:

**Interface SetInterface<T>**

* All Superinterfaces:

java.lang.Iterable<T>

public interface SetInterface<T>

extends java.lang.Iterable<T>

* + - **Methods inherited from interface java.lang.Iterable**

forEach, iterator, spliterator

* + **Method Detail**
    - **contains**

boolean contains([T](file:///C:\Users\wkwan\OneDrive\Desktop\SetInterface.html) element)

Checks if the specified element is in this set.

Parameters:

element - the element whose presence in this set is to be tested

Returns:

true if this set contains the specified element

* + - **remove**

boolean remove([T](file:///C:\Users\wkwan\OneDrive\Desktop\SetInterface.html) element)

Removes the specified element from this set if it is present.

Parameters:

element - the element to be removed, if present

Returns:

true if this set contained the specified element

* + - **add**

boolean add([T](file:///C:\Users\wkwan\OneDrive\Desktop\SetInterface.html) element)

Adds the specified element to this set if it is not already present.

Parameters:

element - the element to be added to this set

Returns:

true if this set did not already contain the specified element

* + - **length**

int length()

Returns the number of elements in this set, its cardinality

Returns:

the cardinality of this set

* + - **subset**

boolean subset([SetInterface](file:///C:\\Users\\wkwan\\OneDrive\\Desktop\\SetInterface.html" \o "interface in <Unnamed>) set)

Compares the specified set with this set to determine whether this set is a subset of the specified set: every member of this set is contained in the specified set.

Parameters:

set - the set to be compared to with this set

Returns:

true if this set is a subset of the specified set

* + - **equals**

boolean equals([SetInterface](file:///C:\\Users\\wkwan\\OneDrive\\Desktop\\SetInterface.html" \o "interface in <Unnamed>) set)

Compares the specified set with this set for equality: the two sets have the same size, and every member of the specified set is contained in this set.

Parameters:

element - the element to be removed, if present

Returns:

true if this set contained the specified element

* + - **union**

[SetInterface](file:///C:\Users\wkwan\OneDrive\Desktop\SetInterface.html)<[T](file:///C:\Users\wkwan\OneDrive\Desktop\SetInterface.html)> union([SetInterface](file:///C:\\Users\\wkwan\\OneDrive\\Desktop\\SetInterface.html" \o "interface in <Unnamed>) set)

Performs the union operation on two sets, this set and the set specified.

Parameters:

set - the set to be operated with this set

Returns:

a set containing the elements in this set and the set specified

* + - **clear**

void clear()

Removes all the elements from this set. The set will be empty after this call.

Section 2:

All methods are called through their respective test method. Each test prints the precondition of the set the method was called on, the method argument, return value, and the postcondition of the set(s) after it executed. The information can be used to determine the logic correctness.

Every case is subject to each test (some more than once). The examples provided are specific examples that show proper logic.

**Case 1:** A and B are equal but distinct sets, for example, A = {1, 2, 3}, B = {2, 1, 3}

Examples of add and remove test output:

Test: add(T element)

Method call on: {2,3}

Method argument: 1

Return value: true

After method call: {1,2,3}

Test completed.

Test: remove(T element)

Method call on: {1,2,3}

Method argument: 2

Return value: true

After method call: {1,3}

Test completed.

Test: remove(T element)

Method call on: {1,3}

Method argument: 4

Return value: false

After method call: {1,3}

Test completed.

Examples of subset, equals, and union test output:

Test: subset(SetInterface set)

Method call on: {1,2,3}

Method argument: {2,1,3}

Return value: true

After method call (left): {1,2,3}

After method call (right): {2,1,3}

Test completed.

Test: equals(SetInterface set)

Method call on: {1,2,3}

Method argument: {2,1,3}

Return value: true

After method call (left): {1,2,3}

After method call (right): {2,1,3}

Test completed.

Test: union(SetInterface set)

Method call on: {1,2,3}

Method argument: {2,1,3}

Return value: {3,1,2}

After method call (left): {1,2,3}

After method call (right): {2,1,3}

Test completed.

The tests show that the methods do not change the sets, as they shouldn’t, and return the proper result.

Examples of clear test output:

Test: clear()

Method call on: {1,3}

Method argument: none

Return value: none

After method call: {}

Test completed.

Test: clear()

Method call on: {1,3}

Method argument: none

Return value: none

After method call: {}

Test completed

**Case 2:** A and B are such that they have different lengths but one is a subset of the other, for example, A = {1}, B = {1, 2}

Test: subset(SetInterface set)

Method call on: {1}

Method argument: {1,2}

Return value: true

After method call (left): {1}

After method call (right): {1,2}

Test completed.

Test: subset(SetInterface set)

Method call on: {1,2}

Method argument: {1}

Return value: false

After method call (left): {1,2}

After method call (right): {1}

Test completed.

The test correctly shoes A is a subset of B, but B is not a subset of A.

**Case 3:** A and B are non-empty and different in size but have common elements, for example, A = {1, 2, 3}, B = {2, 3, 4, 5}

Examples of equals and union test output:

Test: equals(SetInterface set)

Method call on: {1,2,3}

Method argument: {2,3,4,5}

Return value: false

After method call (left): {1,2,3}

After method call (right): {2,3,4,5}

Test completed.

Test: union(SetInterface set)

Method call on: {1,2,3}

Method argument: {2,3,4,5}

Return value: {5,4,3,2,1}

After method call (left): {1,2,3}

After method call (right): {2,3,4,5}

Test completes

**Case 4:** A and B are non-empty with nothing in common, for example, A = {1}, B = {2, 3}

Examples of subset test output:

Test: subset(SetInterface set)

Method call on: {1}

Method argument: {2,3}

Return value: false

After method call (left): {1}

After method call (right): {2,3}

Test completed.

Test: subset(SetInterface set)

Method call on: {2,3}

Method argument: {1}

Return value: false

After method call (left): {2,3}

After method call (right): {1}

Test completed.

A and B have nothing in common, therefore they can’t be subsets in either direction. Hence both return values are false.

**Case 5:** One is non-empty and the other empty, for example, A = {1, 2}, B = {}

Examples of subset test output:

Test: subset(SetInterface set)

Method call on: {1,2}

Method argument: {}

Return value: false

After method call (left): {1,2}

After method call (right): {}

Test completed.

Test: subset(SetInterface set)

Method call on: {}

Method argument: {1,2}

Return value: true

After method call (left): {}

After method call (right): {1,2}

Test completed.

B is an empty set, and thus must be a subset of A. This is correctly demonstrated by the test.