Name of the Course : Complete Java SE8 Developer Bootcamp

Level : Difficult

Tool Stack : Java8 and Junit5

Problem Statement : Provide a code solution to perform the set operations namely union, intersection and difference on arrays.

Description : Create class SetOperation with static methods 1. Public static Boolean performSetOperations(Arraylist<Integer> a1, Arraylist<Integer> b1), which accepts two integer array and returns array list.The function would perform the following set operations ,’+’ for set union ,’\*’ for set-intersection and ‘-‘ for set difference.

2. pubic static void main method, to perform

1. Read 2 n integers as input and a set operator of type char.Create two array list to store n elements in each array list.
2. Input consists of 2n+2 integers.The first integer denotes the size of the arraylist n , the next n integers are for the first arraylist.

**Sample Input 1:**

3

1

2

3

3

5

7

+

**Sample Output 1:**

1

2

3

5

7

**Sample Input 2:**

4

10

9

8

7

2

4

6

8

**\***

**Sample Output 2:**

8

**Sample Input 3:**

4

5

10

15

20

0

10

12

20

-

**Sample Output 3:**

5

15

Code:

import java.util.ArrayList;

import java.util.Iterator;

import java.util.LinkedHashSet;

import java.util.Scanner;

public class SetOperations{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

ArrayList<Integer> aa=new ArrayList<Integer>();

for(int i=0;i<n;i++)

{

aa.add(sc.nextInt());

}

ArrayList<Integer> aa2=new ArrayList<Integer>();

for(int i=0;i<n;i++)

{

aa2.add(sc.nextInt());

}

char c=sc.next().charAt(0);

ArrayList<Integer> op=new ArrayList<Integer>();

op=performSetOperations( aa, aa2,c);

Iterator<Integer> itr=op.iterator();

while(itr.hasNext())

{

int a=(Integer)itr.next();

System.out.println(a);

}

}

public static ArrayList<Integer> performSetOperations

(ArrayList<Integer>aa,ArrayList<Integer>aa2,char c)

{

ArrayList<Integer> aa3= new ArrayList<Integer>();

if(c=='+')

{

aa.removeAll(aa2);

aa.addAll(aa2);

aa3=aa;

}

if(c=='\*')

{

aa.retainAll(aa2);

aa3=aa;

}

if(c=='-')

{

aa.removeAll(aa2);

aa3=aa;

}

return aa3;

}

}

Junit Testing

**import** **static** org.junit.Assert.*assertTrue*;

**import** java.text.ParseException;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.stream.Collectors;

**import** org.junit.Test;

**import** handson.SetOperations;

**public** **class** TestSetOperations {

@Test

**public** **void** testSetOperations\_Success() **throws** ParseException {

**int**[] a = { 1, 2, 3 };

**int**[] b = { 3, 5, 7 };

**int**[] c = {1,2,3,5,7};

ArrayList<Integer> list1 = (ArrayList<Integer>) Arrays.*stream*(a) // IntStream

.boxed() // Stream<Integer>

.collect(Collectors.*toList*());

ArrayList<Integer> list2 = (ArrayList<Integer>) Arrays.*stream*(b) // IntStream .boxed() // Stream<Integer>

.collect(Collectors.*toList*());

ArrayList<Integer> list3 = (ArrayList<Integer>) Arrays.*stream*(c) // IntStream

.boxed() // Stream<Integer>

.collect(Collectors.*toList*());

*assertTrue*(list3.equals(SetOperations.*performSetOperations*(list1,list2,'+') ));

// test case 2

**int**[] a1 = { 10, 9, 8,7 };

**int**[] b1 = { 2, 4, 6,8 };

**int**[] c1 = {8};

ArrayList<Integer> list11 = (ArrayList<Integer>) Arrays.*stream*(a1) // IntStream .boxed() // Stream<Integer>

.collect(Collectors.*toList*());

ArrayList<Integer> list22 = (ArrayList<Integer>) Arrays.*stream*(b1) // IntStream .boxed() // Stream<Integer>

. collect(Collectors.*toList*());

ArrayList<Integer> list33 = (ArrayList<Integer>) Arrays.*stream*(c1) // IntStream

.boxed() // Stream<Integer>

.collect(Collectors.*toList*());

*assertTrue*(list33.equals(SetOperations.*performSetOperations*(list11,list22,'\*') ));

// test case 3

**int**[] a2 = { 5,10, 15, 20 };

**int**[] b2 = { 0, 10, 12,20 };

**int**[] c2 = {5,15};

ArrayList<Integer> list111 = (ArrayList<Integer>) Arrays.*stream*(a2) // IntStream

.boxed() // Stream<Integer>

.collect(Collectors.*toList*());

ArrayList<Integer> list222 = (ArrayList<Integer>) Arrays.*stream*(b2) // IntStream

.boxed() // Stream<Integer> .collect(Collectors.*toList*());

ArrayList<Integer> list333 = (ArrayList<Integer>) Arrays.*stream*(c2) // IntStream

.boxed() // Stream<Integer> .collect(Collectors.*toList*());

*assertTrue*(list333.equals(SetOperations.*performSetOperations*(list111,list222,'-') ));

}

}

Learning outcome: Participant could able to learn how to use the Generics , Collections API.