Exercice 1 :

import java.util.\*;

public class DistinctElementsInTwoSets {

public static void findDistinctElements(int [] setOne, int [] setTwo){

System.out.println("Set 1: " + Arrays.toString(setOne) + ", Set 2: " + Arrays.toString(setTwo));

Map<Integer, Integer> map = new HashMap<>();

for (int i = 0; i <setOne.length ; i++) {

int element = setOne[i];

if(map.containsKey(element)) {

int count = map.get(element);

map.put(element, count+1);

}else

map.put(element, 1);

}

for (int i = 0; i <setTwo.length ; i++) {

int element = setTwo[i];

if(map.containsKey(element)) {

int count = map.get(element);

map.put(element, count+1);

}else

map.put(element, 1);

}

//get sum of distinct elements

int sum = 0;

Set<Integer> set = map.keySet();

Iterator<Integer> iterator = set.iterator();

while (iterator.hasNext()){

int key = iterator.next();

if(map.get(key)==1)

sum += key;

}

System.out.println("Distinct Elements Sum : " + sum);

}

public static void main(String[] args) {

int [] setOne = {3, 1, 7, 9};

int [] setTwo = {2, 4, 1, 9, 3};

findDistinctElements(setOne, setTwo);

}

}

Exercice2 :

import java.io.\*;

class GFG {

static int n = 3;

static int dotProduct(int vect\_A[], int vect\_B[])

{

int product = 0;

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

product = product + vect\_A[i] \* vect\_B[i];

return product;

}

static void crossProduct(int vect\_A[], int vect\_B[],

int cross\_P[])

{

cross\_P[0] = vect\_A[1] \* vect\_B[2]

- vect\_A[2] \* vect\_B[1];

cross\_P[1] = vect\_A[2] \* vect\_B[0]

- vect\_A[0] \* vect\_B[2];

cross\_P[2] = vect\_A[0] \* vect\_B[1]

- vect\_A[1] \* vect\_B[0];

}

public static void main(String[] args)

{

int vect\_A[] = { 3, -5, 4 };

int vect\_B[] = { 2, 6, 5 };

int cross\_P[] = new int[n];

System.out.print("Dot product:");

System.out.println(dotProduct(vect\_A, vect\_B));

System.out.print("Cross product:");

crossProduct(vect\_A, vect\_B, cross\_P);

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

System.out.print(cross\_P[i] + " ");

}

}