// COLLECTIONS

**23.Removing Keys from HashMap**

Given a method with a HashMap<Integer,string> as input. Write code to remove all the entries having keys multiple of 4 and return the size of the final hashmap.

Include a class **UserMainCode** with a static method **sizeOfResultandHashMap** which accepts hashmap as input.

The return type of the output is an integer which is the size of the resultant hashmap.

Create a class **Main** which would get the input and call the static method **sizeOfResultandHashMap** present in the UserMainCode.

**Input and Output Format:**

First input corresponds to the size of the hashmap.

Input consists of a hashmap<integer,string>.

Output is an integer which is the size of the hashmap.

Refer sample output for formatting specifications.

**Sample Input 1:**

3

2

hi

4

hello

12

hello world

**Sample Output 1:**

1

**Sample Input 2:**

3

2

hi

4

sdfsdf

3

asdf

**Sample Output 2:**

2

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** Main

{

**public** **static** **int** sizeOfResultandHashMap(HashMap<Integer ,String>hm)

{ **int** k,count=0;

Iterator<Integer>it=hm.keySet().iterator();

**while**(it.hasNext())

{

k=it.next();

**if**(k%4!=0)

count++;

}

**return** count;

}

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

HashMap<Integer,String>hm=**new** HashMap<Integer,String>();

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

{

hm.put(in.nextInt(),in.next());

}

System.***out***.println(Main.*sizeOfResultandHashMap*(hm));

}

}

**40.Average of Elements in Hashmap**

  Given a method with a HashMap<int, float> as input. Write code to find out avg of all values whose keys are even numbers. Round the average to two decimal places and return as output.  
  
[Hint : If the average is 5.901, the rounded average value is 5.9 . It the average is 6.333, the rounded average value is 6.33 . ]

Include a class **UserMainCode** with a static method **avgOfEven** which accepts a HashMap<int, float> as input.

The return type of the output is a floating point value which is the average of all values whose key elements are even numbers.

Create a class **Main** which would get the input and call the static method **avgOfEven** present in the UserMainCode.

**Input and Output Format:**

Input consists of the number of elements in the HashMap and the HashMap<int, float>.

Output is a floating point value that corresponds to the average.

Refer sample output for formatting specifications.

**Sample Input 1:**

3

1

2.3

2

4.1

6

6.2

**Sample Output 1:**

5.15

**Sample Input 2:**

3

9

3.1

4

6.3

1

2.6

**Sample Output 2:**

6.3

**import** java.text.DecimalFormat;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** Main

{

**public** **static** **float** sizeOfResultandHashMap(HashMap<Integer ,Float>hm)

{ **int** k,count=0;

**float** sum=0.0f;

DecimalFormat f=**new** DecimalFormat("#.00");

Iterator<Integer>i=hm.keySet().iterator();

**while**(i.hasNext())

{

k=i.next();

**if**(k%2==0)

{

sum=sum+hm.get(k);

count++;

}

}

**float** d=sum/count;

String str=f.format(d);

**float** d1=Float.*parseFloat*(str);

**return** d1;

}

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

HashMap<Integer,Float>hm=**new** HashMap<Integer,Float>();

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

{

hm.put(in.nextInt(),in.nextFloat());

}

System.***out***.println(Main.*sizeOfResultandHashMap*(hm));

}

}

**41.Calculate Average – Hash Map**

Write amethod that accepts the input data as a hash map and finds out the avg of all values whose keys are odd numbers.   
  
Include a class **UserMainCode** with a static method **calculateAverage** which accepts aHashMap<Integer,Double> and the size of the HashMap. The return type (Double) should return the calculated average. Round the average to two decimal places and return it.

Create a Class Main which would be used to accept Input values and store it as a hash map, and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of an integer n corresponds to number of hash map values, followed by 2n values. (index followed by value).

Output consists of a Double.

Refer sample input and output for formatting specifications.

**Sample Input :**

4

1

3.41

2

4.1

3

1.61

4

2.5

**Sample Output :**

2.51

**import** java.text.DecimalFormat;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** Main

{

**public** **static** **double** sizeOfResultandHashMap(HashMap<Integer ,Float>hm)

{ **int** k,count=0;

**float** sum=0.0f;

DecimalFormat f=**new** DecimalFormat("#.00");

Iterator<Integer>i=hm.keySet().iterator();

**while**(i.hasNext())

{

k=i.next();

**if**(k%2!=0)

{

sum=sum+hm.get(k);

count++;

}

}

**float** d=sum/count;

String str=f.format(d);

**double** d1=Double.*parseDouble*(str);

**return** d1;

}

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

HashMap<Integer,Float>hm=**new** HashMap<Integer,Float>();

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

{

hm.put(in.nextInt(),in.nextFloat());

}

System.***out***.println(Main.*sizeOfResultandHashMap*(hm));

}

}

**48.Sum of Lowest marks**

Given input as HashMap, value consists of marks and rollno as key.Find the sum of the lowest three subject marks from the HashMap.

Include a class **UserMainCode** with a static method **getLowest** which accepts a Hashmap with marks and rollno.

The return type of the output is the sum of lowest three subject marks.

Create a class **Main** which would get the input and call the static method **getLowest** present in the UserMainCode.

**Input and Output Format:**

First line of the input corresponds to the HashMap size.

Input consists a HashMap with marks and rollno.

Output is an integer which is the sum of lowest three subject marks.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

1

54

2

85

3

74

4

59

5

57

**Sample Output 1:**

170

**Sample Input 2:**

4

10

56

20

58

30

87

40

54

**Sample Output 2:**

168

**import** java.util.Arrays;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** Main

{

**public** **static** **int** sizeOfResultandHashMap(HashMap<Integer ,Integer>hm)

{

**int** k=0;

**int** a[]=**new** **int**[hm.size()];

Iterator<Integer> it=hm.values().iterator();

**while**(it.hasNext())

{

**int** l=it.next();

a[k]=l;

++k;

}

Arrays.*sort*(a);

**return** a[0]+a[1]+a[2];

}

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

HashMap<Integer,Integer>hm=**new** HashMap<Integer,Integer>();

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

{

hm.put(in.nextInt(),in.nextInt());

}

System.***out***.println(Main.*sizeOfResultandHashMap*(hm));

}

}

**52.Removing elements from HashMap**

Given a HashMap as input, write a program to perform the following operation :  If the keys are divisible by 3 then remove that key and its values and print the number of remaining keys in the hashmap.

Include a class **UserMainCode** with a static method **afterDelete** which accepts a HashMap as input.

The return type of the output is an integer which represents the count of remaining elements in the hashmap.

Create a class **Main** which would get the input and call the static method **afterDelete** present in the UserMainCode.

**Input and Output Format:**

First input corresponds to the size of hashmap

Input consists a HashMap

Output is an integer which is the count of remaining elements in the hashmap.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

339

RON

1010

JONS

3366

SMITH

2020

TIM

**Sample Output 1:**

**2**

**Sample Input 2:**

5

1010

C2WE

6252

XY4E

1212

M2ED

7070

S2M41ITH

8585

J410N

**Sample Output 2:**

3

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** Main

{

**public** **static** **int** sizeOfResultandHashMap(HashMap<Integer ,String>hm)

{ **int** k,count=0;

Iterator<Integer> it=hm.keySet().iterator();

**while**(it.hasNext())

{

k=it.next();

**if**(k%3!=0)

count++;

}

**return** count;

}

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

HashMap<Integer,String>hm=**new** HashMap<Integer,String>();

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

{

hm.put(in.nextInt(),in.next());

}

System.***out***.println(Main.*sizeOfResultandHashMap*(hm));

}

}

**11.Largest Key in HashMap**

Write a program that construts a hashmap and returns the value corresponding to the largest key.

Include a class UserMainCode with a static method **getMaxKeyValue** which accepts a string. The return type (String) should be the value corresponding to the largest key.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of 2n+1 values. The first value corresponds to size of the hashmap. The next n pair of numbers equals the integer key and value as string.

Output consists of a string which is the value of largest key.

Refer sample output for formatting specifications.

**Sample Input 1:**

3

12

amron

9

Exide

7

SF

**Sample Output 1:**

Amron

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** Main

{

**public** **static** String sizeOfResultandHashMap(HashMap<Integer ,String>hm)

{

**int** max=0;

String s2="";

Iterator<Integer> it=hm.keySet().iterator();

**while**(it.hasNext())

{

**int** a=it.next();

**if**(a>max)

{

max=a;

String s3=hm.get(a);

s2=s3;

}

}

**return** s2;

}

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

HashMap<Integer,String>hm=**new** HashMap<Integer,String>();

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

{

hm.put(in.nextInt(),in.next());

}

System.***out***.println(Main.*sizeOfResultandHashMap*(hm));

}

}

**17.Vowels, Arrays & ArrayLists**

Write a program to read an array of strings and return an arraylist which consists of words whose both first and last characters are vowels. Assume all inputs are in lowecase.

Include a class UserMainCode with a static method **matchCharacter** which accepts a string array. The return type shoud be an arraylist which should contain elements as mentioned above.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' string correspond to the elements in the array.

Output consists of strings which are elements of arraylist

Refer sample output for formatting specifications.

**Sample Input 1:**

4

abcde

pqrs

abci

orto

**Sample Output 1:**

abcde

abci

orto

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.Scanner;

**public** **class** Main

{

**public** **static** ArrayList<String>vowelCheck(String a[])

{

ArrayList<String> al=**new** ArrayList<String>();

**for**(**int** i=0;i<a.length;i++)

{

**if**(a[i].matches("[a||e||i||o||u]{1}.\*[a||e||i||o||u]{1}"))

{

al.add(a[i]);

}

}

**return** al;

}

**public** **static** **void** main(String[] args)

{

Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

String[] a=**new** String[n];

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

{

a[i]=s.next();

}

ArrayList<String>arr=Main.*vowelCheck*(a);

Iterator<String> it=arr.iterator();

**while**(it.hasNext()){

System.***out***.println(it.next());

}

}

}

**18.Transfer from Hashmap to Arraylist**

Write a program that constructs a hashmap with “employee id” as key and “name” as its value. Based on the rules below, on being satisfied, the name must be added to the arraylist.

i)First character should be small and the last character should be Capital.

ii)In name at least one digit should be there.

Include a class UserMainCode with a static method **getName** which accepts a hashmap. The return type is an arraylist as expected in the above statement

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of 2n+1 values. The first value corresponds to size of the hashmap. The next n pair of numbers contains the employee id and name.

Output consists of arraylist of strings as mentioned in the problem statement.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

1

ravi5raJ

2

sita8gitA

3

ram8sitA

4

rahul

**Sample Output 1:**

ravi5raJ

sita8gitA

ram8sitA

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**public** **class** Main

{

**public** **static** ArrayList<String>getName(HashMap<Integer,String>hm)

{

ArrayList<String> al=**new** ArrayList<String>();

Iterator<String> it=hm.values().iterator();

**while**(it.hasNext())

{

String s=it.next();

**if**(s.matches("^[a-z].\*") &&s.matches(".\*[0-9]{1}.\*") &&s.matches(".\*[A-Z]$"))

al.add(s);

}

**return** al;

}

**public** **static** **void** main(String[] args)

{

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

HashMap<Integer,String>hm=**new** HashMap<Integer,String>();

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

{

hm.put(in.nextInt(),in.next());

}

ArrayList<String>arr=Main.*getName*(hm);

Iterator<String> it=arr.iterator();

**while**(it.hasNext()){

System.***out***.println(it.next());

}

}

}

**28.Remove 3 Multiples**

Write a program that accepts an ArrayList of integers as input and removes every 3rd element and prints the final ArrayList.

Suppose the given arrayList contains 10 elements remove the 3rd, 6th and 9th elements.

Include a class **UserMainCode** with a static method “**removeMultiplesOfThree**” that accepts an ArrayList<Integer> as arguement and returns an ArrayList<Integer>.

Create a class **Main** which would get the required input and call the static method **removeMultiplesOfThree** present in the UserMainCode.

**Input and Output Format:**

The first line of the input consists of an integer n, that corresponds to the number of elements to be added in the ArrayList.

The next n lines consist of integers that correspond to the elements in the ArrayList.

Output consists of an ArrayList of integers.

**Sample Input:**

6

3

1

11

19

17

19

**Sample Output**

3

1

19

17

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.Scanner;

**public** **class** Main

{

**public** **static** ArrayList<Integer>getName(ArrayList<Integer> al)

{

ArrayList<Integer> array=**new** ArrayList<Integer>();

Iterator<Integer> it=al.iterator();

**while**(it.hasNext())

{

array.add(it.next());

array.add(it.next());

it.next();

}

**return** array;

}

**public** **static** **void** main(String[] args)

{

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

ArrayList<Integer> al=**new** ArrayList<Integer>();

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

{

al.add(in.nextInt());

}

ArrayList<Integer>arr=Main.*getName*(al);

Iterator<Integer> it=arr.iterator();

**while**(it.hasNext()){

System.***out***.println(it.next());

}

}

}

**52.Employees & Designations**

A Company wants to obtain employees of a particular designation. You have been assigned as the programmer to build this package. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:  
    Read Employee details from the User. The details would include name and designaton in the given order. The datatype for name and designation is string.  
    Build a hashmap which contains the name as key and designation as value.  
    You decide to write a function **obtainDesignation** which takes the hashmap and designation as input and returns a string List of employee names who belong to that designation as output. Include this function in class UserMainCode. Display employee name's in ascending order.  
Create a Class Main which would be used to read employee details in step 1 and build the hashmap. Call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of employee details. The first number indicates the size of the employees. The next two values indicate the employee name employee designation. The last string would be the designation to be searched.  
Output consists of a array values containing employee names.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
Manish  
MGR  
Babu  
CLK  
Rohit  
MGR  
Viru  
PGR  
MGR  
  
**Sample Output 1:**  
Manish  
Rohit

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Map;

**import** java.util.Map.Entry;

**import** java.util.Scanner;

**public** **class** Main

{

**public** **static** ArrayList<String> getName(HashMap<String,String>hm,String s1)

{

ArrayList<String> al=**new** ArrayList<String>();

**for**(Map.Entry<String,String> map:hm.entrySet())

{

**if**(s1.equals(map.getValue()))

{

al.add(map.getKey());

}

}

Collections.*sort*(al);

**return** al;

}

**public** **static** **void** main(String[] args)

{

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

HashMap<String,String>hm=**new** HashMap<String,String>();

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

{

hm.put(in.next(),in.next());

}

String s=in.next();

ArrayList<String> l1=**new** ArrayList<String>();

l1=Main.*getName*(hm,s);

Iterator<String> it=l1.iterator();

**while**(it.hasNext())

{

String s2=it.next();

System.***out***.println(s2);

}

}

}

**53.Grade Calculator**

A School wants to give assign grades to its students based on their marks. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:  
    Read student details from the User. The details would include name, mark in the given order. The datatype for name is string, mark is float.  
    You decide to build a hashmap. The hashmap contains name as key and mark as value.  
  
BUSINESS RULE:  
1. If Mark is less than 60, then grade is FAIL.  
2. If Mark is greater than or equal to 60, then grade is PASS.  
Note: FAIL/PASS should be in uppercase.  
Store the result in a new Hashmap with name as Key and grade as value.  
4. You decide to write a function **calculateGrade** which takes the above hashmap as input and returns the hashmap as output. Include this function in class UserMainCode.  
Create a Class Main which would be used to read student details in step 1 and build the hashmap. Call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of student details. The first number indicates the size of the students. The next two values indicate the name, mark.  
  
Output consists of a name and corresponding grade for each student.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
3  
Avi  
76.36  
Sunil  
68.42  
Raja  
36.25  
  
**Sample Output 1:**  
Avi  
PASS  
Sunil  
PASS  
Raja  
FAIL

**import** java.io.IOException;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.LinkedHashMap;

**import** java.util.Map;

**import** java.util.Map.Entry;

**import** java.util.Scanner;

**public** **class** Main

{

**public** **static** LinkedHashMap<String,String>getName(LinkedHashMap<String,Float>hm)

{

LinkedHashMap<String,String> res=**new** LinkedHashMap<String,String>();

**for**(Map.Entry<String,Float>map:hm.entrySet())

{

**if**(map.getValue()>=60)

{

res.put(map.getKey(),"PASS");

}

**else**

res.put(map.getKey(),"FAIL");

}

**return** res;

}

**public** **static** **void** main(String[] args) **throws** IOException

{

Scanner in=**new** Scanner(System.***in***);

**int** n=in.nextInt();

LinkedHashMap<String,Float>hm=**new** LinkedHashMap<String,Float>();

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

{

hm.put(in.next(),in.nextFloat());

}

LinkedHashMap<String,String>arr=Main.*getName*(hm);

**for**(Map.Entry<String,String>map:arr.entrySet())

{

System.***out***.println(map.getKey()+"\n"+map.getValue());

}

}

}

}

**56.ArrayList to String Array**

Write a program that performs the following actions:  
    Read n strings as input.  
    Create an arraylist to store the above n strings in this arraylist.  
    Write a function convertToStringArray which accepts the arraylist as input.  
    The function should sort the elements (strings) present in the arraylist and convert them into a string array.  
    Return the array.  
Include a class UserMainCode with the static method **convertToStringArray** which accepts an arraylist and returns an array.  
  
Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of n+1 integers. The first integer denotes the size of the arraylist, the next n strings are values to the arraylist.  
Output consists of an arrayas per step 4.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
a  
d  
c  
b  
**Sample Output 1:**  
a  
b  
c  
d

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Iterator;

**import** java.util.Scanner;

**public** **class** Main

{

**public** **static** String[] convertToStringArray (ArrayList<String> l1,**int** n)

{

String a[]=**new** String[n];

**int** i=0;

Iterator<String> it=l1.iterator();

**while**(it.hasNext())

{

a[i]=it.next();

++i;

}

**return** a;

}

**public** **static** **void** main(String[] args)

{

Scanner sc=**new** Scanner(System.*in*);

**int** k1=Integer.*parseInt*(sc.nextLine());

ArrayList<String> l1=**new** ArrayList<String>();

**for**(**int** i=0;i<k1;i++)

{

l1.add(sc.next());

}

Collections.*sort*(l1);

String []s=Main.*convertToStringArray*(l1,k1);

**for**(String s1:s)

{

System.*out*.println(s1);

}

}

}

**57.State ID generator**

Write a program to generate the state ID.  
     1)Read n Strings as input(as State Name).  
     2)Create a String Array to Store the above Input.  
     3)Write a function **getStateId** which accepts String Array as input.  
     4)Create a HashMap<String,String> which stores state name as key and state Id as Value.  
     5)The function getStateId returns the HashMap to the Main Class.  
  
Include UserMainCode Class With static method **getStateId** which accepts String array and return a hashmap.  
  
Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.  
  
  
**Input and Output Format:**  
Input Consists of an integer n denotes the size of the string array.  
Output consists of an HashMap displayed in the string array order.  
  
**Sample Input 1:**  
3  
Kerala  
Gujarat  
Goa  
  
**Sample Output 1:**  
KER:Kerala  
GUJ:Gujarat  
GOA:Goa

**import** java.util.Iterator;

**import** java.util.LinkedHashMap;

**import** java.util.Scanner;

**class** Main

{

**public** **static** LinkedHashMap<String, String>calculateElectricityBill(String a[])

{

String s1="";

LinkedHashMap<String,String>hm=**new** LinkedHashMap<String,String>();

**for**(**int** i=0;i<a.length;i++)

{

s1=a[i].toUpperCase();

hm.put(s1.substring(0,3),a[i]);

}

**return** hm;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** n=sc.nextInt();

String a[]=**new** String[n];

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

{

a[i]=sc.next();

}

LinkedHashMap<String, String>ans=Main.*calculateElectricityBill*(a);

Iterator<String> it=ans.keySet().iterator();

**while**(it.hasNext())

{

String s2=it.next();

String s3=ans.get(s2);

System.***out***.println(s2+":"+s3);

}

}

}

**58.ArrayList to String Array**

Write a program that performs the following actions:  
  
1.Read m strings as input (fruit names).  
2.Create an arraylist to store the above m strings in this arraylist.  
3.Read n strings as input (fruit names).  
4.Create an arraylist to store the above n strings in this arraylist.  
5.Write a function fruitSelector which accepts the arraylists as input.  
6.Remove all fruits whose name ends with 'a' or 'e' from first arrayList and remove all fruits whose name begins  with 'm' or 'a' from second arrayList then combine the two lists and return the final output as a String array.  
7.If the array is empty the program will print as “No fruit found”  
Include a class UserMainCode with the static method **fruitSelector** which accepts the two arraylists and returns an array.  
  
Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.  
  
Output consists of an array as per step 6. Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
3  
Apple  
Cherry  
Grapes  
4  
Orange  
Mango  
Melon  
Apple  
**Sample Output 1:**  
Cherry  
Grapes  
Orange

**import** java.util.ArrayList;

**import** java.util.Scanner;

**class** Main

{

**static** **int** *j*=0;

**public** **static** String[] calculateElectricityBill(**int** n,ArrayList<String>al,**int** n1,ArrayList<String>al1)

{

String a[]=**new** String[n+n1];

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

{

String s1=al.get(i);

**if**(s1.charAt(s1.length()-1)!='a'&& s1.charAt(s1.length()-1)!='e'

&&s1.charAt(s1.length()-1)!='A'&& s1.charAt(s1.length()-1)!='E')

{

a[*j*]=s1;

*j*++;

}

}

**for**(**int** i=0;i<n1;i++)

{

String s1=al1.get(i);

**if**(s1.charAt(0)!='A'&& s1.charAt(0)!='M'&& s1.charAt(0)!='a'&& s1.charAt(0)!='m')

{

a[*j*]=s1;

*j*++;

}

}

**return** a;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** n=sc.nextInt();

ArrayList<String> al=**new** ArrayList<String>();

ArrayList<String> al1=**new** ArrayList<String>();

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

{

al.add(sc.next());

}

**int** n1=sc.nextInt();

**for**(**int** i=0;i<n1;i++)

{

al1.add(sc.next());

}

String[] ans=Main.*calculateElectricityBill*(n,al,n1,al1);

**for**(**int** i=0;i<Main.*j*;i++)

System.***out***.println(ans[i]);

}

}

**59.Elements in ArrayList**

Use Collection Methods.  
Write a program that takes two ArrayLists as input and  finds out all elements present either in A or B, but not in both.

Include a class UserMainCode with the static method arrayListSubtractor which accepts the two arraylists and returns an array.  
  
Create a Class Main which would be used to read the inputs and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.  
  
Output consists of an array. The elements in the output array need to be printed in sorted order.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
1  
8  
3  
5  
2  
3  
5  
**Sample Output 1:**  
1  
8  
  
**Sample Input 2:**  
4  
9  
1  
3  
5  
4  
1  
3  
5  
6  
**Sample Output 2:**  
6  
9

import java.io.\*;

import java.util.\*;

class Main2

{

       public static int[] arrayListSubtractor(List<Integer>l1,List<Integer>l2)

       {

             Iterator<Integer> it=l1.iterator();

             List<Integer> l3=new ArrayList<Integer>();

             l3.addAll(l1);

             while(it.hasNext())

             {

                    Integer a=it.next();

                    if(l2.contains(a))

                    {

                           it.remove();

                    }

             }

             it=l2.iterator();

             while(it.hasNext())

             {

                    Integer a=it.next();

                    if(l3.contains(a))

                    {

                           it.remove();

                    }

             }

             l1.addAll(l2);

             int arr[]=new int[l1.size()];

             int i=0;

             for(Integer a:l1)

             {

                    arr[i++]=a;

             }

             return arr;

       }

       public static void main(String[] args) throws IOException {

             BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

             List<Integer> l1=new ArrayList<Integer>();

             List<Integer> l2=new ArrayList<Integer>();

             int n=Integer.parseInt(br.readLine());

             for(int i=0;i<n;i++)

             {

                    l1.add(Integer.parseInt(br.readLine()));

             }

             n=Integer.parseInt(br.readLine());

             for(int i=0;i<n;i++)

             {

                    l2.add(Integer.parseInt(br.readLine()));

             }

             int arr[]= Main2.arrayListSubtractor(l1, l2);

             Arrays.sort(arr);

             for(int i:arr)

             {

                    System.out.println(i);

             }

       }

}

**60.Price Calculator - II**

Write a small price calculator application with the below mentioned flow:  
  
1. Read a value n indicating the total count of devices. This would be followed by the name and price of the device. The datatype for name would be String and price would be float.  
  
2. Build a hashmap containing the peripheral devices with name as key and price as value.  
  
3. Read a value m indicating the number of devices for which the price has to be calculated. This would be followed by device names.  
  
4. For each devices mentioned in the array calcuate the total price.  
  
5. You decide to write a function costEstimator which takes the above hashmap and array as input and returns the total price (float) as output with two decimal points. Include this function in class UserMainCode.  
  
Create a Class Main which would be used to read details in step 1 and build the hashmap. Call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of device details. The first number indicates the size of the devices. The next two values indicate the name,price.  
  
This would be followed by m indicating the size of the device array. The next m values would be the device names.  
Output consists of the total price in float.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
3  
Monitor  
1200.36  
Mouse  
100.42  
Speakers  
500.25  
2  
Speakers  
Mouse  
**Sample Output 1:**  
600.67

**import** java.util.HashMap;

**import** java.text.DecimalFormat;

**import** java.util.Scanner;

**class** Main

{

**static** **int** *l*=0;

**public** **static** **float** calculateElectricityBill(**int** n,HashMap<String,Float>hm,**int** n1,String a[])

{

**float** f=0;

DecimalFormat df=**new** DecimalFormat("#.00");

**for**(**int** i=0;i<n1;i++)

{

**if**(hm.containsKey(a[i]))

{

f=f+hm.get(a[i]);

}

}

f=Float.*parseFloat*(df.format(f));

**return** f;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** n=sc.nextInt();

HashMap<String,Float>hm=**new** HashMap<String,Float>();

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

{

hm.put(sc.next(),sc.nextFloat());

}

**int** n1=sc.nextInt();

String s1[]=**new** String[n1];

**for**(**int** i=0;i<n1;i++)

{

s1[i]=sc.next();

}

System.***out***.println(Main.*calculateElectricityBill*(n,hm,n1,s1));

}

}

**55.Experience Validator**

Write a program to valiadate the experience of an employee.  
    An employee who has recently joined the organization provides his year of passing and total number of years of experience in String format. Write code to validate his experience against the current date.  
  
1) Input consists of two String first represent the year of passed out and the second string reperesent the year of experience.  
2) create a function with  name **validateExp**which accepts two string as input and boolean as output.  
3) The difference between current year and year of pass should be more than or equal to Experience  
Return true if all condition are true.  
  
Note:Conside 2015 as the current year.  
  
Include a class UserMainCode with the static function validateExp  
  
Create a Class Main which would be used to accept the boolean and call the static method present in UserMainCode.  
  
**Input and Output Formate:**  
Input consists of two Strings.  
output will display true if the given data are correct.  
  
**Sample Input:**  
2001  
5  
  
**Sample Output:**  
TRUE

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** **boolean** getvalues(String s,String s1)

{

**int** y1=Integer.*parseInt*(s);

**int** y2=2015;

**int** y=Math.*abs*(y1-y2);

**int** e=Integer.*parseInt*(s1);

**if**(y>=e)

**return** **true**;

**else**

**return** **false**;

}

**public** **static** **void** main(String args[]){

Scanner sc = **new** Scanner(System.*in*);

String s=sc.nextLine();

String s1=sc.nextLine();

System.*out*.print(UserMainCode.*getvalues*(s,s1));

}

}

**65.Even Sum & Duplicate Elements**

Write a program to read a integer array, Remove the duplicate elements and display sum of even numbers in the output. If input array contain only odd number then return -1.  
Include a class UserMainCode with a static method **sumElements** which accepts the integer array. The return type is integer.  
  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of an integer n which is the number of elements followed by n integer values.  
Output consists of integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
7  
2  
3  
54  
1  
6  
7  
7  
**Sample Output 1:**  
62  
  
**Sample Input 2:**  
6  
3  
7  
9  
13  
17  
21  
**Sample Output 2:**  
-1

**import** java.util.Iterator;

**import** java.util.LinkedHashSet;

**import** java.util.Scanner;

**class** Main

{

**static** **int** *l*=0;

**public** **static** **int** display(**int** n[])

{

LinkedHashSet<Integer>h1=**new** LinkedHashSet<Integer>();

**int** s=0;

**for**(**int** i=0;i<n.length;i++)

{

h1.add(n[i]);

}

Iterator<Integer> it=h1.iterator();

**while**(it.hasNext())

{

**int** k=it.next();

**if**(k%2==0)

{

s=s+k;

}

}

**if**(s>0)

**return** s;

**else**

**return** -1;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** n=sc.nextInt();

**int** a[]=**new** **int**[n];

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

{

a[i]=sc.nextInt();

}

System.***out***.println(Main.*display*(a));

}

}

**67.Integer Factorial**

Give an array of integer as input, store the numbers and their factorials in an hashmap and print the same.  
Include a class UserMainCode with a static method **getFactorial** which accepts the integer array. The return type is the hashmap which is printed key:value.  
  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of a number denoting the size of the array and followed by the elements.  
Output consists of a hashmap printed in the output format .  
Refer sample output for formatting specifications.  
  
**Sample Input1:**  
4  
2  
3  
5  
4  
**Sample Output1:**  
2:2  
3:6  
5:120  
4:24

**import** java.util.Iterator;

**import** java.util.LinkedHashMap;

**import** java.util.LinkedHashSet;

**import** java.util.Scanner;

**class** Main

{

**static** **int** *l*=0;

**public** **static** LinkedHashMap<Integer,Integer> display(**int** n[])

{

LinkedHashMap<Integer,Integer>hm=**new** LinkedHashMap<Integer,Integer>();

**for**(**int** i=0;i<n.length;i++)

{

**int** fact=1;

**for**(**int** j=1;j<=n[i];j++)

{fact=fact\*j;}

hm.put(n[i],fact);

}

**return** hm;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** s=Integer.*parseInt*(sc.nextLine());

**int** []a=**new** **int**[s];

**for**(**int** i=0;i<s;i++)

{

a[i]=sc.nextInt();

}

LinkedHashMap<Integer,Integer>hm2=**new** LinkedHashMap<Integer,Integer>();

hm2=Main.*display*(a);

Iterator<Integer> it=hm2.keySet().iterator();

**while**(it.hasNext())

{

**int** n=it.next();

**int** fac=hm2.get(n);

System.***out***.println(n+":"+fac);

}

}

}

1. **Unique Even Sum**

Write a program to read an array, eliminate duplicate elements and calculate the sum of even numbers (values) present in the array.

Include a class UserMainCode with a static method **addUniqueEven** which accepts a single integer array. The return type (integer) should be the sum of the even numbers. In case there is no even number it should return -1.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

In case there is no even integer in the input array, print**no even numbers** as output. Else print the sum.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

4

2

5

1

4

**Sample Output 1:**

6

**Sample Input 2:**

3

1

1

1

**Sample Output 2:**

no even numbers

**import** java.util.Iterator;

**import** java.util.LinkedHashSet;

**import** java.util.Scanner;

**class** Main

{

**static** **int** *l*=0;

**public** **static** **int** display(**int** n[])

{

LinkedHashSet<Integer>h1=**new** LinkedHashSet<Integer>();

**int** s=0;

**for**(**int** i=0;i<n.length;i++)

{

h1.add(n[i]);

}

Iterator<Integer> it=h1.iterator();

**while**(it.hasNext())

{

**int** k=it.next();

**if**(k%2==0)

{

s=s+k;

}

}

**if**(s>0)

**return** s;

**else**

**return** -1;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** n=sc.nextInt();

**int** a[]=**new** **int**[n];

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

{

a[i]=sc.nextInt();

}

System.***out***.println(Main.*display*(a));

}

}

**Palindrome & Vowels**

Write a program to check if a given string is palindrome and contains at least two different vowels.

Include a class UserMainCode with a static method **checkPalindrome** which accepts a string. The return type (integer) should be 1 if the above condition is satisfied, otherwise return -1.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

Note – Case Insensitive while considering vowel, i.e a & A are same vowel, But Case sensitive while considering palindrome i.e abc CbA are not palindromes.

**Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single Integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

abceecba

**Sample Output 1:**

valid

**Sample Input 2:**

abcd

**Sample Output 2:**

invalid

**import** java.util.Iterator;

**import** java.util.LinkedHashSet;

**import** java.util.Scanner;

**class** Main

{

**static** **int** *l*=0;

**public** **static** **void** display(String s)

{

StringBuffer sb=**new** StringBuffer(s);

**int** k=0;

LinkedHashSet<Character>l1=**new** LinkedHashSet<Character>();

String s2=sb.reverse().toString();

**if**(s2.equals(s))

{

String s3=s2.toLowerCase();

**for**(**int** i=0;i<s3.length();i++)

{

l1.add(s3.charAt(i));

}

Iterator<Character> it=l1.iterator();

**while**(it.hasNext())

{

**char** a=it.next();

**if**(a=='a'||a=='e'||a=='i'||a=='o'||a=='u')

k++;

}

}

**if**(k>=2)

System.***out***.println("Valid");

**else**

System.***out***.println("Invalid");

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

String s=sc.nextLine();

Main.*display*(s);

}

}

1. **ArrayList Manipulation**

Write a program that performs the following actions:

1. Read 2n integers as input.
2. Create two arraylists to store n elements in each arraylist.
3. Write a function **generateOddEvenList**which accepts these two arraylist as input.
4. The function fetch the odd index elements from first array list and even index elements from second array list and add them to a new array list according to their index.
5. Return the arraylist.

Include a class UserMainCode with the static method **generateOddEvenList** which accepts two arraylist and returns an arraylist.

Create a Class Main which would be used to read 2n integers and call the static method present in UserMainCode.

Note:

- The index of first element is 0.

- Consider 0 as an even number.

- Maintain order in the output array list

**Input and Output Format:**

Input consists of 2n+1 integers. The first integer denotes the size of the arraylist, the next n integers are values to the first arraylist, and the last n integers are values to the second arraylist.

Output consists of a modified arraylist as per step 4.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

12

13

14

15

16

2

3

4

5

6

**Sample Output 1:**

2

13

4

15

6

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** Main

{

**static** **int** *l*=0;

**public** **static** ArrayList<Integer> display(ArrayList<Integer> al1,ArrayList<Integer> al2)

{

ArrayList<Integer>al3=**new** ArrayList<Integer>();

**for**(**int** i=0;i<al1.size();i++)

{

**if**(i%2==0)

al3.add(al2.get(i));

**else**

al3.add(al1.get(i));

}

**return** al3;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** s=Integer.*parseInt*(sc.nextLine());

ArrayList<Integer>al1=**new** ArrayList<Integer>();

ArrayList<Integer>al2=**new** ArrayList<Integer>();

**for**(**int** i=0;i<s;i++)

al1.add(sc.nextInt());

**for**(**int** i=0;i<s;i++)

al2.add(sc.nextInt());

ArrayList<Integer>al3=**new** ArrayList<Integer>();

al3=Main.*display*(al1,al2);

Iterator<Integer> it=al3.iterator();

**while**(it.hasNext())

{

**int** n=it.next();

System.***out***.println(n);

}

}

}

1. **Duplicate Characters**

Write a Program which removes duplicate characters from the string. Your program should read a sentence (string) as input from user and return a string removing duplicate characters. Retain the first occurance of the duplicate character. Assume the characters are case – sensitive.

Include a class UserMainCode with a static method **removeDuplicates** which accepts a string. The return type is the modified sentence of type string.

Create a Class Main which would be used to accept the input string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

hi this is sample test

**Sample Output 1:**

hi tsample

**Sample Input 2:**

ABC DEF

**Sample Output 2:**

ABC DEF

**import** java.util.Iterator;

**import** java.util.LinkedHashSet;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** String nameFormatter(String s1)

{ String temp;

StringBuffer sb=**new** StringBuffer();

LinkedHashSet<Character>hs=**new** LinkedHashSet<Character>();

**for**(**int** i=0;i<s1.length();i++)

{

hs.add(s1.charAt(i));

}

Iterator<Character>itr=hs.iterator();

**while**(itr.hasNext())

{

**char** o=itr.next();

sb.append(o);

}

**return** sb.toString();

}

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.***in***);

String s1=in.nextLine();

System.***out***.println(Main.*nameFormatter*(s1));

in.close();

}

}

1. **Mastering Hashmap**

You have recently learnt about hashmaps and in order to master it, you try and use it in all of your programs.

Your trainer / teacher has given you the following exercise:

1. Read 2n numbers as input where the first number represents a key and second one as value. Both the numbers are of type integers.
2. Write a function **getAverageOfOdd**to find out average of all values whose keys are represented by odd numbers. Assume the average is an int and never a decimal number. Return the average as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read 2n numbers and build the hashmap. Call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a 2n+ 1 integers. The first integer specifies the value of n (essentially the hashmap size). The next pair of n numbers denote the key and value.

Output consists of an integer representing the average.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

2

34

1

4

5

12

4

22

**Sample Output 1:**

8

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** Main

{

**public** **static** **int** display(HashMap<Integer,Integer> h1)

{

**int** av=0,c=0,s=0;

Iterator<Integer> it=h1.keySet().iterator();

**while**(it.hasNext())

{

**int** a=it.next();

**if**(a%2!=0)

{

**int** b=h1.get(a);

s=s+b;

c++;

}

}

av=s/c;

**return** av;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** n=sc.nextInt();

HashMap<Integer,Integer> h1=**new** HashMap<Integer,Integer>();

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

{

h1.put(sc.nextInt(),sc.nextInt());

}

System.***out***.println(Main.*display*(h1));

}

}

1. **Managers & Hashmaps**

A Company wants to automate its payroll process. You have been assigned as the programmer to build this package. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1.    Read Employee details from the User. The details would include id, designation and salary in the given order. The datatype for id is integer, designation is string and salary is integer.

2.    You decide to build two hashmaps. The first hashmap contains employee id as key and designation as value, and the second hashmap contains same employee ids as key and salary as value.

3.    The company decides to hike the salary of managers by 5000. You decide to write a function **increaseSalaries**which takes the above hashmaps as input and returns a hashmap with only managers id and their increased salary as output. Include this function in class UserMainCode.  
Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps.

Call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee designation and employee salary.  
Output consists of a single string. Refer sample output for formatting specifications.  
**Sample**

**Input 1:**  
2  
2  
programmer  
3000  
8  
manager  
50000  
**Sample Output 1:**  
8  
55000

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** UserMainCode2

{

**public** **static** HashMap<Integer,Integer>

display(HashMap<Integer,String>hm,HashMap<Integer,Integer>hm1)

{

Iterator<Integer> it=hm.keySet().iterator();

Iterator<Integer> it1=hm1.values().iterator();

HashMap<Integer,Integer>hm3=**new** HashMap<Integer,Integer>();

**while**(it.hasNext())

{

**int** a=it.next();

String s=hm.get(a);

**int** b=it1.next();

**if**(s.equals("manager"))

{

hm3.put(a,b+5000);

}

}

**return** hm3;

}

}

**class** Main

{

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.*in*);

**int** n=Integer.*parseInt*(in.nextLine());

HashMap<Integer,String>hm=**new** HashMap<Integer,String>();

HashMap<Integer,Integer>hm1=**new** HashMap<Integer,Integer>();

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

{

**int** a=in.nextInt();

String s=in.next();

**int** sal=in.nextInt();

hm.put(a,s);

hm1.put(a,sal);

}

HashMap<Integer,Integer>hm3=UserMainCode2.*display*(hm,hm1);

Iterator<Integer>it=hm3.keySet().iterator();

**for**(**int** i=0;i<hm3.size();i++)

{

**int** k=it.next();

**int** fac=hm3.get(k);

System.*out*.println(k+"\n"+fac);

}

}

1. }**Check first and last word**

Write a program to check if the first word and the last word in the input string match.

Include a class **UserMainCode** with a static method **“check”** that accepts a string argument and returns an int. If the first word and the last word in the string match, the method returns the number of characters in the word. Else the method returns the sum of the number of characters in the first word and last word.

Create a class **Main** which would get the input as a String and call the static method **check** present in the UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output is an integer.

**Sample Input 1:**

how are you you are how

**Sample Output 1:**

3

**Sample Input 2:**

how is your child

**Sample Output 2:**

8

**import** java.util.Scanner;

**publicclass** UserMainCode {

**staticint** validateNumber(String s1)

{

**int** a=s1.indexOf(" ");

**int** b=s1.lastIndexOf(" ");

String s=s1.substring(0,a);

String s2=s1.substring(b+1);

**if**(s.equals(s2))

**return** s.length();

**return** (s.length()+s2.length());

}

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner in=**new** Scanner(System.*in*);

String n1=in.nextLine();

System.*out*.println(UserMainCode.*validateNumber*(n1));

}

}

**14.Retirement**

Given an input as HashMap which contains key as the ID and dob as value of employees, write a program to find out employees eligible for retirement. A person is eligible for retirement if his age is greater than or equal to 60.

Assume that the current date is 01/01/2014.

Include a class **UserMainCode** with a static method “retirementEmployeeList” that accepts a HashMap<String,String> as input and returns a ArrayList<String>. In this method, add the Employee IDs of all the retirement eligible persons to list and return the sorted list.

(Assume date is in dd/MM/yyyy format).

Create a class **Main** which would get the HashMap as input and call the static method **retirementEmployeeList**present in the UserMainCode.

**Input and Output Format:**

The first line of the input consists of an integer n, that corresponds to the number of employees.

The next 2 lines of the input consists of strings that correspond to the id and dob of employee 1.

The next 2 lines of the input consists of strings that correspond to the id and dob of employee 2.

and so on...

Output consists of the list of employee ids eligible for retirement in sorted order.

**Sample Input :**  
4  
C1010

02/11/1987

C2020

15/02/1980

C3030

14/12/1952

T4040

20/02/1950

**Sample Output :**

[C3030, T4040]

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Calendar;

**import** java.util.Collections;

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.LinkedHashMap;

**import** java.util.Map;

**import** java.util.Scanner;

**public** **class** UserMainCode

{

**public** **static** ArrayList<String> retirementEmployeeList (HashMap<String,String> hm)

{

String s1="01/01/2014";

**int** year;

ArrayList <String> al=**new** ArrayList <String>();

**for**(Map.Entry<String,String> map:hm.entrySet())

{

String s=map.getValue();

**if**(s.matches("[0-9]{2}/[0-9]{2}/[0-9]{4}"))

{

SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");

sdf.setLenient(**false**);

**try**

{

Calendar c=Calendar.*getInstance*();

Date d1=sdf.parse(s1);

Date d2=sdf.parse(s);

c.setTime(d1);

**int** y1=c.get(Calendar.*YEAR*);

**int** m1=c.get(Calendar.*MONTH*);

**int** d11=c.get(Calendar.*DATE*);

c.setTime(d2);

**int** y2=c.get(Calendar.*YEAR*);

**int** m2=c.get(Calendar.*MONTH*);

**int** d22=c.get(Calendar.*DATE*);

year=Math.*abs*(y2-y1);

**if**(m1>m2)

year--;

**else** **if**(m2==m1 && d11>d22)

year--;

**if**(year>=60)

{

String k=map.getKey();

al.add(k);

}

}

**catch**(ParseException e)

{

**return** al;

}

}

}

Collections.*sort*(al);

**return** al;

}

**public** **static** **void** main(String args[]){

Scanner sc=**new** Scanner(System.*in*);

**int** n=Integer.*parseInt*(sc.nextLine());

LinkedHashMap<String,String>a1=**new** LinkedHashMap<String,String>();

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

{

a1.put(sc.nextLine(),sc.nextLine());

}

ArrayList<String> l1=**new** ArrayList<String>();

l1=UserMainCode.*retirementEmployeeList*(a1);

System.*out*.println(l1);

}

}

**17.Unique Characters**

Given a String as input , write a program to count and print the number of unique characters in it.

Include a class **UserMainCode** with a static method “**checkUnique**” that accepts a String as input and returns the number of unique characters in it. If there are no unique characters in the string, the method returns -1.

Create a class **Main**which would get a String as input and call the static method **checkUnique**present in the UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output consists of an integer.

**Sample Input 1:**

HOWAREYOU

**Sample Output 1:**

7

(Hint :Unique characters are : H,W,A,R,E,Y,U and other characters are repeating)

**Sample Input 2:**

MAMA

**Sample Output2:**

-1

**import** java.util.Scanner;

**public** **class** Main

{

**static** **int** uniqueCounter(String s)

{

**int** i,l=0;

String temp;

**for**(i=0;i<s.length();i++)

{

**if**(i!=0)

temp=s.substring(0,i).concat(s.substring(i+1));

**else**

temp=s.substring(i+1);

String c=s.charAt(i)+"";

**if**(!temp.contains(c))

{

++l;

}

//else

//return -1;

}

**return** l;

}

**public** **static** **void** main(String args[])

{

Scanner in=**new** Scanner(System.*in*);

String str=in.nextLine();

**int** ans=Main.*uniqueCounter*(str);

**if**(ans==0)

System.*out*.println("-1");

**else**

System.*out*.println(ans);

in.close();

}

}

**19. ArrayList and Set Operations**

Write a program that performs the following actions:

1.    Read 2n integers as input & a set operator (of type char).

2.    Create two arraylists to store n elements in each arraylist.

3.    Write a function **performSetOperations**which accepts these two arraylist and the set operator as input.

4.    The function would perform the following set operations:.  
  
'+' for SET-UNION  
  
'\*' for SET-INTERSECTION  
  
'-' for SET-DIFFERENCE  
  
Refer to sample inputs for more details.

5.    Return the resultant arraylist.  
  
Include a class UserMainCode with the static method **performSetOperations** which accepts two arraylist and returns an arraylist.  
  
Create a Class Main which would be used to read 2n+1 integers and call the static method present in UserMainCode.  
  
Note:  
  
- The index of first element is 0.  
  
  
   
  
**Input and Output Format:**  
  
Input consists of 2n+2 integers. The first integer denotes the size of the arraylist, the next n integers are values to the first arraylist, and the next n integers are values to the second arraylist and the last input corresponds to that set operation type.  
  
Output consists of a modified arraylist as per step 4.  
  
Refer sample output for formatting specifications.  
  
**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

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.HashSet;

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** ArrayList<Integer> getvalues(ArrayList<Integer>a1,ArrayList<Integer>a2,**char** c)

{

ArrayList<Integer>op1=**new** ArrayList<Integer>();

HashSet<Integer> set1 = **new** HashSet<Integer>(a1);

HashSet<Integer> set2 = **new** HashSet<Integer>(a2);

HashSet<Integer> set3 = **new** HashSet<Integer>();

**switch**(c)

{

**case** '+':

set3.addAll(set1);

set3.addAll(set2);

op1.addAll(set3);

**break**;

**case** '\*':

set1.retainAll(set2);

op1.addAll(set1);

**break**;

**case** '-':

set1.removeAll(set2);

op1.addAll(set1);

**break**;

}

Collections.*sort*(op1);

**return** op1;

}

**public** **static** **void** main(String args[]){

Scanner sc = **new** Scanner(System.*in*);

**int** n=Integer.*parseInt*(sc.nextLine());

ArrayList<Integer>a1=**new** ArrayList<Integer>();

ArrayList<Integer>a2=**new** ArrayList<Integer>();

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

a1.add(Integer.*parseInt*(sc.nextLine()));

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

a2.add(Integer.*parseInt*(sc.nextLine()));

**char** c=sc.nextLine().charAt(0);

System.*out*.println(UserMainCode.*getvalues*(a1,a2,c));

}

}

23. **All Vowels**

Write a Program to check if given word contains exactly five vowels and the vowels are in alphabetical order. Return 1 if the condition is satisfied else return -1. Assume there is no repetition of any vowel in the given string and all letters are in lower case.

Include a class UserMainCode with a static method **testOrderVowels** which accepts a string The return type is integer based on the condition stated above.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

acebisouzz

**Sample Output 1:**

valid

**Sample Input 2:**

alphabet

**Sample Output 2:**

Invalid

**import**java.util.Scanner;

**publicclass**UserMainCode {

**publicstatic**StringgetVowels(String s1)

{ String s=”yes”;

**if**(s1.contains(“a”) && s1.contains(“e”) &&s1.contains(“i”) &&s1.contains(“o”) && s1.contains(“u”))

s=yes;

else

s=”no”;

**return** s;

}

**publicstaticvoid** main(String[] args) {

Scanner in=**new**Scanner(System.*in*);

String s1=in.nextLine();

System.*out*.println(UserMainCode.*getVowels*(s1));

in.close();

}

}

**27.Employee Bonus**

A Company wants to give away bonus to its employees. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read Employee details from the User. The details would include id, DOB (date of birth) and salary in the given order. The datatype for id is integer, DOB is string and salary is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and salary as value.
3. If the age of the employee in the range of 25 to 30 years (inclusive), the employee should get bonus of 20% of his salary and in the range of 31 to 60 years (inclusive) should get 30% of his salary. store the result in TreeMap in which Employee ID as key and revised salary as value. Assume the age is caculated based on the date 01-09-2014. (Typecast the bonus to integer).
4. Other Rules:

a. If Salary is less than 5000 store -100.

b. If the age is less than 25 or greater than 60 store -200.

c. a takes more priority than b i.e both if a and b are true then store -100.

1. You decide to write a function **calculateRevisedSalary** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee DOB and employee salary. The Employee DOB format is “dd-mm-yyyy”

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

2

1010

20-12-1987

10000

2020

01-01-1985

14400

**Sample Output 1:**

1010

12000

2020

17280  
**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**import** java.util.TreeMap;

**public** **class** UserMainCode

{

**public** **static** TreeMap<Integer,Integer> display(HashMap<Integer,String>hm,HashMap<Integer,Integer>hm1)

{

TreeMap<Integer,Integer>hm2=**new** TreeMap<Integer,Integer>();

Iterator<Integer> it=hm.keySet().iterator();

**while**(it.hasNext())

{

**int** y=it.next();

String dob=hm.get(y);

SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MM-yyyy");

sdf.setLenient(**false**);

String now="01-09-2014";

**try**

{

Calendar c=Calendar.*getInstance*();

Date d1=sdf.parse(now);

Date d2=sdf.parse(dob);

c.setTime(d1);

**int** y1=c.get(Calendar.*YEAR*);

**int** m1=c.get(Calendar.*MONTH*);

**int** day1=c.get(Calendar.*DATE*);

c.setTime(d2);

**int** y2=c.get(Calendar.*YEAR*);

**int** m2=c.get(Calendar.*MONTH*);

**int** day2=c.get(Calendar.*DATE*);

**int** age=y1-y2;

**if**(m1<m2)

age--;

**else** **if**(m1==m2 && day1<day2)

age--;

**if**(hm1.get(y)<5000)

{

hm2.put(y,-200);

}

**else** **if**((age<25 || age>60) && hm1.get(y)<5000 )

hm2.put(y,-100);

**else** **if**(age>=25 && age<=30)

{**float** bonus=(**float**)0.2\*hm1.get(y)+hm1.get(y);

hm2.put(y,(**int**)bonus );

}

**else** **if**(age>30 && age<=60)

{ **float** bonus=(**float**) (0.3\*hm1.get(y))+hm1.get(y);

hm2.put(y,(**int**)bonus );

}

**else** **if**(age<25 || age>60)

hm2.put(y,-200);

}

**catch**(Exception e)

{e.printStackTrace();

}}

**return** hm2;

}

**public** **static** **void** main(String args[])

{

HashMap<Integer,String>hm=**new** HashMap<Integer,String>();

HashMap<Integer,Integer>hm1=**new** HashMap<Integer,Integer>();

TreeMap<Integer,Integer>hm2=**new** TreeMap<Integer,Integer>();

Scanner sc=**new** Scanner(System.*in*);

**int** s=Integer.*parseInt*(sc.nextLine());

**for**(**int** i=0;i<s;i++)

{

**int** id=Integer.*parseInt*(sc.nextLine());

String ss=sc.nextLine();

**int** salary=Integer.*parseInt*(sc.nextLine());

hm.put(id,ss);

hm1.put(id,salary);

}

hm2=UserMainCode.*display*(hm,hm1);

Iterator<Integer> it=hm2.keySet().iterator();

**for**(**int** i=0;i<s;i++)

{

**int** id=it.next();

**int** bonus=hm2.get(id);

System.*out*.println(id);

System.*out*.println(bonus);

}

}

}

**28.** **Grade Calculator**

A School wants to assign grades to its students based on their marks. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read student details from the User. The details would include roll no, mark in the given order. The datatype for id is integer, mark is integer.
2. You decide to build a hashmap. The hashmap contains roll no as key and mark as value.
3. BUSINESS RULE:

1. If Mark is greater than or equal to 80 store medal as ""GOLD"".

2. If Mark is less then to 80 and greater than or equal to 60 store medal as ""SILVER"".

3 .If Mark is less then to 60 and greater than or equal to 45 store medal as ""BRONZE"" else store ""FAIL"".

Store the result in TreeMap in which Roll No as Key and grade as value.

4. You decide to write a function **calculateGrade** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of employee details. The first number indicates the size of the students. The next two values indicate the roll id, mark.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

2

1010

80

100

40

**Sample Output 1:**

100

FAIL

1010

GOLD

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Map;

**import** java.util.Scanner;

**import** java.util.TreeMap;

**public** **class** UserMainCode

{

**public** **static** TreeMap<Integer,String> calculateGrade (HashMap<Integer,Integer> h1)

{

TreeMap<Integer,String> hm=**new** TreeMap<Integer,String>();

**for**(Map.Entry<Integer,Integer> map:h1.entrySet())

{

**if**(map.getValue()>=80)

hm.put(map.getKey(),"GOLD");

**else** **if**(map.getValue()>=60)

hm.put(map.getKey(),"SILVER");

**else** **if**(map.getValue()>=45)

hm.put(map.getKey(),"BRONZE");

**else**

hm.put(map.getKey(),"FAIL");

}

**return** hm;

}

**public** **static** **void** main(String []args){

Scanner sc=**new** Scanner(System.*in*);

**int** s=sc.nextInt();

HashMap<Integer,Integer>hm=**new** HashMap<Integer,Integer>();

**for**(**int** i=0;i<s;i++)

{

hm.put(sc.nextInt(),sc.nextInt());

}

TreeMap<Integer,String>hm1=**new** TreeMap<Integer,String>();

hm1=UserMainCode.*calculateGrade*(hm);

Iterator<Integer> it=hm1.keySet().iterator();

**for**(**int** i=0;i<s;i++)

{

**int** n=it.next();

String fac=hm1.get(n);

System.*out*.println(n);

System.*out*.println(fac);

}

}

}**37.Symmetric Difference**

Write a program to read two integer array and calculate the symmetric difference of the two arrays. Finally Sort the array.  
Symmetric difference is the difference of A Union B and A Intersection B ie. [ (A U B) - (A ^ B)]  
Union operation merges the two arrays and makes sure that common elements appear only once. Intersection operation includes common elements from both the arrays.  
Ex - A={12,24,7,36,14} and B={11,26,7,14}.  
A U B ={ 7,11,12,14,24,26,36} and  
A ^ B = {7,14}  
Symmetric difference of A and B after sorting= [A U B] - [ A ^ B] = {11,12,24,26,36}.  
Include a class UserMainCode with a static method **getSymmetricDifference** which accepts the integer array. The return type is an integer array.  
Create a Class Main which would be used to accept the two integer arrays and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of an integer n which is the number of elements followed by n integer values. The same sequnce is followed for the next array.  
Output consists of sorted symmetric difference array.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
5  
11  
5  
14  
26  
3  
3  
5  
3  
1  
**Sample Output 1:**  
1  
11  
14  
26

**import** java.util.Arrays;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.util.Scanner;

**public** **class** UserMainCode

{

**public** **static** **int**[] getSymmetricDifference (**int**[] a1,**int**[] a2)

{

HashSet<Integer> set1 = **new** HashSet<Integer>();

HashSet<Integer> set2 = **new** HashSet<Integer>();

**for**(**int** i=0;i<a1.length;i++)

set1.add(a1[i]);

**for**(**int** i=0;i<a2.length;i++)

set2.add(a2[i]);

HashSet<Integer> set3 = **new** HashSet<Integer>();

set3.addAll(set1);

set3.addAll(set2);

set1.retainAll(set2);

set3.removeAll(set1);

**int** k[]=**new** **int**[set3.size()];

**int** i=0;

Iterator<Integer> itr=set3.iterator();

**while**(itr.hasNext())

{

k[i]=itr.next();

++i;

}

Arrays.*sort*(k);

**return** k;

}

**public** **static** **void** main(String[] args)

{

**int** n,m;

Scanner sin = **new** Scanner(System.*in*);

n = sin.nextInt();

**int**[] a1 = **new** **int**[n];

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

{

a1[i] = sin.nextInt();

}

m = sin.nextInt();

**int**[] a2 = **new** **int**[m];

**for**(**int** i=0;i<m;i++)

{

a2[i] = sin.nextInt();

}

**int**[] result = UserMainCode.*getSymmetricDifference* (a1,a2);

**for**(**int** i=0;i<result.length;i++)

System.*out*.println(result[i]);

}

}

**42.Interest Calculation**

Write a  program to calculate amount of the acccount holders based on the below mentioned prototype:  
1. Read account details from the User. The details would include id, DOB (date of birth) and amount in the given order. The datatype for id is string, DOB is string and amount is integer.  
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and amount as value.  
3. Rate of interest as on 01/01/2015:  
    a. If the age greater than or equal to 60 then interest rate is 10% of Amount.  
    b.If the age less then to 60 and greater than or equal to 30 then interest rate is 7% of Amount.  
    v. If the age less then to 30 interest rate is 4% of Amount.  
4. Revised Amount= principle Amount + interest rate.  
5.  You decide to write a function **calculateInterestRate** which takes the above hashmaps as input and returns the treemap  as output. Include this function in class UserMainCode.  
  
Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of account details. The first number indicates the size of the acoount. The next three values indicate the user id, DOB and amount. The Employee DOB format is “dd-mm-yyyy”  
Output consists of the user id and the amount for each user one in a line.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
SBI-1010  
20-01-1987  
10000  
SBI-1011  
03-08-1980  
15000  
SBI-1012  
05-11-1975  
20000  
SBI-1013  
02-12-1950  
30000  
**Sample Output 1:**  
SBI-1010:10400  
SBI-1011:16050  
SBI-1012:21400  
SBI-1013:33000

**import** java.util.Calendar;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.TreeMap;

**import** java.util.Scanner;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** UserMainCode

{

**public** **static** TreeMap<String,Integer> display (HashMap<String,String>hm,HashMap<String,Integer>hm1)

{

**int** year=0,amount=0;

**double** dis=0;

String now="01/01/2015";

TreeMap<String,Integer>tm=**new** TreeMap<String,Integer>();

Iterator<String> it=hm.keySet().iterator();

**while**(it.hasNext())

{

String id=it.next();

String dor=hm.get(id);

amount=hm1.get(id);

SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MM-yyyy");

SimpleDateFormat sdf1=**new** SimpleDateFormat("dd/MM/yyyy");

**try**

{

Calendar c=Calendar.*getInstance*();

Date d1=sdf.parse(dor);

Date d2=sdf1.parse(now);

c.setTime(d1);

**int** y1=c.get(Calendar.*YEAR*);

**int** m1=c.get(Calendar.*MONTH*);

**int** d11=c.get(Calendar.*DATE*);

c.setTime(d2);

**int** y2=c.get(Calendar.*YEAR*);

**int** m2=c.get(Calendar.*MONTH*);

**int** d22=c.get(Calendar.*DATE*);

year=Math.*abs*(y2-y1);

**if**(m1>m2)

year--;

**else** **if**(m2==m1 && d11>d22)

year--;

**if**(year>=60)

dis=0.1\*amount+amount;

**else** **if**(year<60 && year>=30 )

dis=0.07\*amount+amount;

**else**

dis=0.04\*amount+amount;

tm.put(id,(**int**)dis);

}

**catch**(Exception e)

{

e.printStackTrace();

}

}

**return** tm;

}

**public** **static** **void** main(String []args){

Scanner sc=**new** Scanner(System.*in*);

**int** s=Integer.*parseInt*(sc.nextLine());

HashMap<String,String>hm=**new** HashMap<String,String>();

HashMap<String,Integer>hm1=**new** HashMap<String,Integer>();

**for**(**int** i=0;i<s;i++)

{

String id=sc.nextLine();

hm.put(id, sc.nextLine());

hm1.put(id,Integer.*parseInt*(sc.nextLine()));

}

TreeMap<String,Integer>tm=**new** TreeMap<String,Integer>();

tm=UserMainCode.*display*(hm,hm1);

Iterator<String> it=tm.keySet().iterator();

**while**(it.hasNext())

{

String n=it.next();

**int** fac=tm.get(n);

System.*out*.println(n+":"+fac);

}

}

}

**43.Discount Rate Calculation**

Write a  program to calculate discount of the acccount holders based on the transaction amount and registration date using below mentioned prototype:  
1. Read account details from the User. The details would include id, DOR (date of registration) and transaction amount in the given order. The datatype for id is string, DOR is string and transaction amount is integer.  
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOR as value, and the second hashmap contains same employee ids as key and amount as value.  
3. Discount Amount as on 01/01/2015:  
    a. If the transaction amount greater than or equal to 20000 and registration greater than or equal to 5     year then discount rate is 20% of transaction amount.  
    b. If the transaction amount greater than or equal to 20000 and registration less then to 5 year then     discount rate is 10% of transaction amount.  
    c. If the transaction amount less than to 20000 and registration greater than or equal to 5 year then     discount rate is 15% of transaction amount.  
    d. If the transaction amount less than to 20000 and registration less then to 5 year then discount rate     is 5% of transaction amount.  
4. You decide to write a function **calculateDiscount** which takes the above hashmaps as input and returns the treemap  as output. Include this function in class UserMainCode.  
  
Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.  
**Input and Output Format:**  
Input consists of transaction details. The first number indicates the size of the employees. The next three values indicate the user id, user DOR and transaction amount. The DOR (Date of Registration) format is “dd-mm-yyyy”  
Output consists of a string which has the user id and discount amount one in a line for each user.  
Refer sample output for formatting specifications.  
**Sample Input 1:**  
4  
A-1010  
20-11-2007  
25000  
B-1011  
04-12-2010  
30000  
C-1012  
11-11-2005  
15000  
D-1013  
02-12-2012  
10000  
**Sample Output 1:**  
A-1010:5000  
B-1011:3000  
C-1012:2250  
D-1013:500

**import** java.util.Calendar;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.TreeMap;

**import** java.util.Scanner;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** UserMainCode

{

**public** **static** TreeMap<String,Integer> display (HashMap<String,String>hm,HashMap<String,Integer>hm1)

{

**int** year=0,amount=0;

**double** dis=0;

String now="01/01/2015";

TreeMap<String,Integer>tm=**new** TreeMap<String,Integer>();

Iterator<String> it=hm.keySet().iterator();

**while**(it.hasNext())

{

String id=it.next();

String dor=hm.get(id);

amount=hm1.get(id);

SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MM-yyyy");

SimpleDateFormat sdf1=**new** SimpleDateFormat("dd/MM/yyyy");

**try**

{

Calendar c=Calendar.*getInstance*();

Date d1=sdf.parse(dor);

Date d2=sdf1.parse(now);

c.setTime(d1);

**int** y1=c.get(Calendar.*YEAR*);

**int** m1=c.get(Calendar.*MONTH*);

**int** d11=c.get(Calendar.*DATE*);

c.setTime(d2);

**int** y2=c.get(Calendar.*YEAR*);

**int** m2=c.get(Calendar.*MONTH*);

**int** d22=c.get(Calendar.*DATE*);

year=Math.*abs*(y2-y1);

**if**(m1>m2)

year--;

**else** **if**(m2==m1 && d11>d22)

year--;

**if**(year>=5 && amount>=20000)

dis=0.2\*amount;

**else** **if**(year<5 && amount>=20000)

dis=0.1\*amount;

**else** **if**(year>=5 && amount<20000)

dis=0.15\*amount;

**else**

dis=0.05\*amount;

tm.put(id,(**int**)dis);

}

**catch**(Exception e)

{

e.printStackTrace();

}

}

**return** tm;

}

**public** **static** **void** main(String []args){

Scanner sc=**new** Scanner(System.*in*);

**int** s=Integer.*parseInt*(sc.nextLine());

HashMap<String,String>hm=**new** HashMap<String,String>();

HashMap<String,Integer>hm1=**new** HashMap<String,Integer>();

**for**(**int** i=0;i<s;i++)

{

String id=sc.nextLine();

hm.put(id, sc.nextLine());

hm1.put(id,Integer.*parseInt*(sc.nextLine()));

}

TreeMap<String,Integer>tm=**new** TreeMap<String,Integer>();

tm=UserMainCode.*display*(hm,hm1);

Iterator<String> it=tm.keySet().iterator();

**while**(it.hasNext())

{

String n=it.next();

**int** fac=tm.get(n);

System.*out*.println(n+":"+fac);

}

}

}