**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

**23.Removing Keys from HashMap**

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.Scanner;

**class**UserMainCode

{

**publicstaticint**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;

}

**publicstaticvoid** 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(UserMainCode.*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

**40.Average of Elements in Hashmap**

**import**java.text.DecimalFormat;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.Scanner;

**class**UserMainCode

{

**publicstaticfloat**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;

}

**publicstaticvoid** 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(UserMainCode.*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

**41.Calculate Average – Hash Map**

**import**java.text.DecimalFormat;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.Scanner;

**class**UserMainCode

{

**publicstaticdouble**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;

}

**publicstaticvoid** 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(UserMainCode.*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

**48.Sum of Lowest marks**

**import**java.util.Arrays;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.Scanner;

**class**UserMainCode

{

**publicstaticint**sizeOfResultandHashMap(HashMap<Integer ,Integer>hm)

{

**int** k=0;

**int** a[]=**newint**[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];

}

**publicstaticvoid** 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(UserMainCode.*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

**52.Removing elements from HashMap**

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.Scanner;

**class**UserMainCode

{

**publicstaticint**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;

}

**publicstaticvoid** 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(UserMainCode.*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

**11.Largest Key in HashMap**

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.Scanner;

**class**UserMainCode

{

**publicstatic** 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;

}

**publicstaticvoid** 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(UserMainCode.*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

**17.Vowels, Arrays &ArrayLists**

**import**java.util.ArrayList;

**import**java.util.Iterator;

**import**java.util.Scanner;

**publicclass**UserMainCode

{

**publicstatic**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=UserMainCode.*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

**18. Transfer from Hashmap to Arraylist**

**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** UserMainCode

{

**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=UserMainCode.*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

**28.Remove 3 Multiples**

**import**java.util.ArrayList;

**import**java.util.Iterator;

**import**java.util.Scanner;

**publicclass**UserMainCode

{

**publicstatic**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;

}

**publicstaticvoid** 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=UserMainCode.*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

**52. Employees & Designations**

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.HashMap;

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

**import** java.util.Scanner;

**public** **class** UserMainCode

{

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

{

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

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

{

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

{

l.add(map.getKey());

}

}

String s[]=**new** String[l.size()];

l.toArray(s);

Arrays.*sort*(s);

**return** s;

}

**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();

String []arr=UserMainCode.*getName*(hm,s);

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

{

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

}

}

}

**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

**53.Grade Calculator**

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.LinkedHashMap;

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

**import** java.util.Scanner;

**public** **class** UserMainCode

{

**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)

{

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=UserMainCode.*getName*(hm);

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

{

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

}

}

}

**56. ArrayList to String Array**

**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

**57.State ID generator**

**import** java.util.Iterator;

**import** java.util.LinkedHashMap;

**import** java.util.Scanner;

**class** UserMainCode

{

**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=UserMainCode.*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

**58. ArrayList to String Array**

**import** java.util.ArrayList;

**import** java.util.Scanner;

**class** UserMainCode

{

**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=UserMainCode.*calculateElectricityBill*(n,al,n1,al1);

**for**(**int** i=0;i<UserMainCode.*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

**59.Elements in ArrayList**

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Scanner;

**class** UserMainCode

{

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

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

{

**int** j=0,m=0;

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

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

{j=0;

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

{

**if**(!al.get(i).equals(al1.get(k)))

{

j++;

}

}

**if**(j==n1)

{

a[*l*]=al.get(i);

*l*++;

}

}

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

{m=0;

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

{

**if**(!al1.get(i).equals(al.get(k)))

{

m++;

}

}

**if**(m==n)

{

a[*l*]=al1.get(i);

*l*++;

}

}

Arrays.*sort*(a);

**return** b;

}

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

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

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

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

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

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

{

al.add(sc.nextInt());

}

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

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

{

al1.add(sc.nextInt());

}

**int** []ans=UserMainCode.*calculateElectricityBill*(n,al,n1,al1);

**for**(**int** i=0;i<UserMainCode.l;i++)

System.*out*.println(ans[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

**60.Price Calculator – II**

**import** java.text.DecimalFormat;

**import** java.util.HashMap;

**import** java.util.Scanner;

**class** UserMainCode

{

**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(UserMainCode.*calculateElectricityBill*(n,hm,n1,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

**65.Even Sum & Duplicate Elements**

**import** java.util.Iterator;

**import** java.util.LinkedHashSet;

**import** java.util.Scanner;

**class** UserMainCode

{

**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(UserMainCode.*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

**67.Integer Factorial**

**import** java.util.Iterator;

**import** java.util.LinkedHashMap;

**import** java.util.LinkedHashSet;

**import** java.util.Scanner;

**class** UserMainCode

{

**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=UserMainCode.*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**

**import** java.util.Iterator;

**import** java.util.LinkedHashSet;

**import** java.util.Scanner;

**class** UserMainCode

{

**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(UserMainCode.*display*(a));

}

}

1. **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

**Palindrome & Vowels**

**import** java.util.Scanner;

**class** UserMainCode

{

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

**public** **static** **int** display(String n)

{

**int** k=0,i;

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

String s1=sb.reverse().toString();

String a=n.toLowerCase();

**if**(s1.equals(n))

{

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

{

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

{

++k;

}

}

}

**if**(k>=2)

**return** 1;

**else**

**return** -1;

}

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

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

String s=sc.nextLine();

System.*out*.println(UserMainCode.*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

1. **ArrayList Manipulation**

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** UserMainCode

{

**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=UserMainCode.*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

1. **Duplicate Characters**

**import** java.util.Iterator;

**import** java.util.LinkedHashSet;

**import** java.util.Scanner;

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

**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();

**if**(o!=' ');

{

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(UserMainCode.*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

1. **Mastering Hashmap**

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** UserMainCode

{

**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(UserMainCode.*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.  
**SampleInput1:**  
2  
2  
programmer  
3000  
8  
manager  
50000  
**SampleOutput1:**  
8  
55000

1. **Managers & Hashmap**

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** UserMainCode

{

**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;

}

**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=UserMainCode.*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);

}

}

}