##### 45. Name Shrinking

Write a program that accepts a string as input and converts the first two names into dotseparated

initials and printa the output.

Input string format is 'fn mn ln'. Output string format is 'ln [mn's 1st character].[fn's 1st character]'

Include a class **UserMainCode** with a static method **getFormatedString** which accepts a string. The return type (String) should return the shrinked name.

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 a string. Output consists of a String.

Refer sample output for formatting specifications. **Sample Input:**

Sachin Ramesh Tendulkar **Sample Output:** Tendulkar R.S

**Main**

**import** java.text.\*;

**import** java.util.\*;

**public** **class** Main

{

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

{

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

String s1=sc.nextLine();

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

}

}

**UserMainCode**

**import** java.util.StringTokenizer;

**class** UserMainCode

{

**public** **static** String getFormatedString(String s1)

{

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

StringTokenizer st=**new** StringTokenizer(s1," ");

String s2=st.nextToken();

String s3=st.nextToken();

String s4=st.nextToken();

sb.append(s4).append(" ");

sb.append(s3.substring(0,1));

sb.append(".");

sb.append(s2.substring(0,1));

**return** sb.toString();

}

}

**60. Date Validation**

Write a program to read a string representing a date. The date can be in any of the three formats 1:dd-MM-yyyy 2: dd/MM/yyyy 3: dd.MM.yyyy If the date is valid, print valid else print invalid. Include a class UserMainCode with a static method **getValidDate** which accepts a string. The return type (integer) should be based on the validity of the date. 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 a string. Output consists of a string. Refer sample output for formatting specifications.

Sample Input 1:

03.12.2013

Sample Output 1: valid Sample Input 2:

03$12$2013

Sample Output 3: Invalid

**Main**

**import** java.util.\*;

**public** **class** Main

{

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

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

String s= sc.next();

**int** b = UserMainCode.*getValidDate*(s);

**if**(b==1)

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

**else**

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

}

}

**UserMainCode**

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

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

**public** **static** **int** getValidDate(String s) {

**if**(s.matches("[0-9]{2}[.]{1}[0-9]{2}[.]{1}[0-9]{4}"))

{

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

sdf.setLenient(**false**);

**try**

{

Date d1=sdf.parse(s);

**return** 1;

}

**catch** (ParseException e)

{

**return** -1;

}

}

**else** **if**(s.matches("[0-9]{2}[/]{1}[0-9]{2}[/][0-9]{4}"))

{

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

sdf.setLenient(**false**);

**try**

{

Date d1=sdf.parse(s);

**return** 1;

}

**catch** (ParseException e)

{

**return** -1;

}

}

**else** **if**(s.matches("[0-9]{2}[-]{1}[0-9]{2}[-][0-9]{4}"))

{

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

sdf.setLenient(**false**);

**try**

{

Date d1=sdf.parse(s);

**return** 1;

}

**catch** (ParseException e)

{

**return** -1;

}

}

**else** **return** -1;

}

}

**65.String Finder**

Given three strings say Searchstring, Str1 and Str2 as input, write a program to find out if Str2 comes after Str1 in the Searchstring.

Include a class **UserMainCode** with a static method “**stringFinder**” that accepts 3 String arguments and returns an integer. The 3 arguments correspond to SearchString, Str1 and Str2. The function returns 1 if Str2 appears after Str1 in the Searchtring. Else it returns 2.

Create a class **Main** which would get 3 Strings as input and call the static method **stringFinder** present in the UserMainCode.

**Input and Output Format:**

Input consists of 3 strings.

The first input corresponds to the SearchString. The second input corresponds to Str1.

The third input corresponds to Str2.

Output consists of a string that is either “yes” or “no”

**Sample Input 1: Sample Output 1:**

geniousRajKumarDev Yes

Raj

Dev

***Sample Input 2: Sample Output 2:***

geniousRajKumarDev No Dev

Raj

**Main**

**import** java.util.\*;

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

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

{

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

String s1=s.next();

String s2=s.next();

String s3=s.next();

**int** b=UserMainCode.*stringFinder*(s1, s2, s3);

**if**(b==1)

{

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

}

**else**

System.***out***.println("No");

s.close();

}

}

**UserMainCode**

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

**public** **static** **int** stringFinder(String s1,String s2,String s3)

{

String a1=s1.toLowerCase();

String a2=s2.toLowerCase();

String a3=s3.toLowerCase();

**if**(a1.contains(a2)&&a1.contains(a3))

{

**if**(a1.indexOf(a2)<a1.indexOf(a3))

{

**return** 1;

}

**else**

**return** 2;

}

**return** 0;

}

}

##### 14. Max Substring

Write a program to accept two string inputs. The first being a source string and second one

a delimiter. The source string contains the delimiter at various locations. Your job is to return the substring with maximum number of characters. If two or more substrings have

maximim number of characters return the substring which appears first. The size of the delimiter is 1.

Include a class UserMainCode with a static method **extractMax** which accepts the string.

The return type (string) should be the max substring.

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 a source string and delimiter. Output consists of a string.

Refer sample output for formatting specifications. **Sample Input 1:**

delhi-pune-patna

-

**Sample Output 1:** Delhi\

**Main**

**import** java.util.\*;

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

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

{

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

String input1=sc.next();

String input2=sc.next();

System.***out***.println(UserMainCode.*extractMax*(input1,input2));

}

}

**UserMainCode**

**import** java.util.\*;

**import** java.util.StringTokenizer;

**public** **class** UserMainCode

{

**public** **static** String extractMax(String input1,String input2)

{

**int** max = 0;

String s3= **null**;

StringTokenizer st=**new** StringTokenizer(input1,"-");

**while**( st.hasMoreTokens())

{

String s2=st.nextToken();

**int** n=s2.length();

**if**(n>max)

{

max=n;

s3=s2;

}

}

**return** s3;

}

}

**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

**Main**

**import** java.util.\*;

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

**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++)

{

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

}

LinkedHashMap<String,String>arr=UserMainCode.*calculateGrade*(hm);

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

{

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

}

}

}

**UserMainCode**

**import** java.util.LinkedHashMap;

**import** java.util.Map;

**class** UserMainCode

{

**public** **static** LinkedHashMap<String,String> calculateGrade(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;

}

}

**28. Grade Calculator REFER 53 FROM LEVEL2**

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:
4. If Mark is greater than or equal to 80 store medal as ""GOLD"".
5. 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**Main**

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.HashMap;

**import** java.util.TreeMap;

**import** java.util.Scanner;

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

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

tm=UserMainCode.*calculateGrade*(hm);

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

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

{

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

String fac=tm.get(n);

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

}

}

}

**UserMainCode**

**import** java.util.Iterator;

**import** java.util.HashMap;

**import** java.util.TreeMap;

**public** **class** UserMainCode

{

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

{

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

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

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

{

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

**int** mark=hm.get(id);

**if**(mark>=80)

tm.put(id,"GOLD");

**else** **if**(mark<80 && mark>=60) tm.put(id,"SILVER");

**else** **if**(mark<60 && mark>=45) tm.put(id,"BRONZE");

**else** tm.put(id,"FAIL");

}

**return** tm;

}

}

**29.Digits - II**

Write a program to read a non-negative integer n, compute the sum of its digits. If sum is greater than 9 repeat the process and calculate the sum once again until the final sum comes to single digit.Return the single digit. Include a class UserMainCode with a static method getDigitSum which accepts the integer value. The return type is integer. Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format: Input consists of a integer. Output consists of integer. Refer sample output for formatting specifications.

Sample Input 1: 9999 Sample Output 1: 9

Sample Input 2: 698 Sample Output 2: 5

**Main**

**import** java.util.Scanner; **public** **class** Main {

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

{

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

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

sum=UserMainCode.*getDigitSum*(a); System.***out***.println(sum);

}

}

**UserMainCode**

**public** **class** UserMainCode

{

**public** **static** **int** getDigitSum(**int** n) {

**int** sum = 0;

**int** n1=n;

**while**(n>10)

{

**int** a = 0 ;

sum = 0;

**while**(n!=0)

{

a = n%10;

sum+= a;

n=n/10;

}

n=sum;

}

**return** sum;

}

}

##### 45. Name Shrinking

Main;

**import** java.util.Scanner;

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

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

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

String s=sc.nextLine();

System.***out***.println(UserMainCode.*getFormatedString*(s));

}

##### }

##### UserMainCOde;

**import** java.util.StringTokenizer;

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

**public** **static** String getFormatedString(String s)

{

StringTokenizer st=**new** StringTokenizer(s," ");

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

**while**(st.hasMoreTokens())

{

String a=st.nextToken();

String b=st.nextToken();

String c=st.nextToken();

sb.append(c.substring(0));

sb.append(" ");

sb.append(b.substring(0,1));

sb.append(".");

sb.append(a.substring(0,1));

//String ss=sb.toString();

}

**return** sb.toString();

}

##### }

**53. Grade Calculator**

Main;

**import** java.util.LinkedHashMap;

**import** java.util.Map;

**import** java.util.Scanner;

**public** **class** Main

{

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

{

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

LinkedHashMap<String,String>hm1=**new** LinkedHashMap<String,String>();

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

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

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

{

String s=sc.next();

**double** d=sc.nextDouble();

hm.put(s,d);

}

LinkedHashMap<String,String>hm2=UserMainCode.*dis*(hm);

**for**(Map.Entry<String,String>entry:hm2.entrySet())

{

System.***out***.println(entry.getKey());

System.***out***.println(entry.getValue());

}

}

}

UserMainCode;

**import** java.util.LinkedHashMap;

**import** java.util.Map;

**class** UserMainCode

{

**public** **static** LinkedHashMap<String,String>dis(LinkedHashMap<String,Double>h1)

{

LinkedHashMap<String,String>h2=**new** LinkedHashMap<String,String>();

**for**(Map.Entry m:h1.entrySet())

{

**double** d=(Double)m.getValue();

**if**(d>60)

{

String s=(String)m.getKey();

h2.put(s,"PASS");

}

**else**

{

String s=(String)m.getKey();

h2.put(s,"FAIL");

}

}

**return** h2;

}

}

**53. Grade Calculator**

Main;

**import** java.util.\*;

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

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

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

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

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

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

{

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

}

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

op=UserMainCode.*noOfDigits*(ip);

Iterator<String> itr= op.keySet().iterator();

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

{

String key=itr.next();

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

String value=op.get(key);

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

}

}

}

UserMainCode;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.LinkedHashMap;

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

**public** **static** LinkedHashMap<String,String> noOfDigits(HashMap<String,Float>hm) {

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

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

String res=**null**;

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

{

String key=itr.next();

**float** value=hm.get(key);

**if**(value>=60)

res="PASS";

**else**

res="FAIL";

op.put(key,res);

}

**return** op;

}

}

**28. Grade Calculator REFER 53 FROM LEVEL2**

Main;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**import** java.util.TreeMap;

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

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

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

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

**int** i;

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

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

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

}

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

t.putAll(UserMainCode.*display*(n, hm));

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

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

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

String g = t.get(r);

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

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

}

}

}

UserMainCode;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.TreeMap;

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

**public** **static** TreeMap<Integer, String> display(**int** n,HashMap<Integer, Integer> h) {

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

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

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

**int** r = i.next();

**int** m = h.get(r);

**if** (m >= 80)

t.put(r, "GOLD");

**else** **if** (m < 80 && m >= 60)

t.put(r, "SILVER");

**else** **if** (m < 60 && m >= 45)

t.put(r, "BRONZE");

**else**

t.put(r, "FAIL");

}

**return** t;

}

}