## 31.Sum of cubes and squares of elements in an array

Write a program to get an int array as input and identify even and odd numbers. If number is odd get cube of it, if number is even get square of it. Finally add all cubes and squares together and return it as output.

Include a class **UserMainCode** with a static method **addEvenOdd** which accepts integer array as input.

The return type of the output is an integer which is the sum of cubes and squares of elements in the array.

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

**Input and Output Format:**

Input consists of integer array.

Output is an integer sum.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

2

6

3

4

5

**Sample Output 1:**

208

```
public class Main {

public static void main(String[] args) {

int a[]={2,4,3,5,6};

System.out.println(summationPattern(a));
}

public static int summationPattern(int[] a) {

int n1=0,n2=0;

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

if(a[i]%2==0)

n1+=(a[i]*a[i]);

else

n2+=(a[i]*a[i]*a[i]);

return n1+n2;

}

}
```

## 32.IP Validator

Write a program to read a string and validate the IP address. Print “Valid” if the IP address is valid, else print “Invalid”.

Include a class **UserMainCode** with a static method **ipValidator** which accepts a string. The return type (integer) should return 1 if it is a valid IP address else return 2.

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 that corresponds to an IP.

Output consists of a string(“Valid” or “Invalid”).

Refer sample output for formatting specifications.

**Note**: An IP address has the format a.b.c.d where a,b,c,d are numbers between 0-255.

**Sample Input 1:**
132.145.184.210
**Sample Output 1:**
Valid

**Sample Input 2:**
132.145.184.290
**Sample Output 2:**
Invalid

```
import java.util.*;

public class Main {

        public static void main(String[] args) {

                String ipAddress="10.230.110.160";

           boolean b=validateIpAddress(ipAddress);

           if(b==true)

                System.out.println("valid ipAddress");

           else

                System.out.println("not a valid ipAddress");

        }

        public static boolean validateIpAddress(String ipAddress) {

                boolean b1=false;

                StringTokenizer t=new StringTokenizer(ipAddress,".");

                                String s=t.nextToken();

                                int a=Integer.parseInt(s);

                                                                int
b=Integer.parseInt(t.nextToken());

                int c=Integer.parseInt(t.nextToken());
```

```
                    int d=Integer.parseInt(t.nextToken());

                    if((a>=0 && a<=255)&&(b>=0 && b<=255)&&(c>=0 && c<=255)&&(d>=0 
&& d<=255))

                              b1=true;

                    return b1;

          }

}
```

## 33.Difference between two dates in days

Get two date strings as input and write code to find difference between two dates in days.

Include a class **UserMainCode** with a static method **getDateDifference** which accepts two date strings as input.

The return type of the output is an integer which returns the diffenece between two dates in days.

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

**Input and Output Format:**

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

2012-03-12

2012-03-14

**Sample Output 1:**

2

**Sample Input 2:**

```
2012-04-25
2012-04-28
```

**Sample Output 2:**

```
3
```

```
import java.text.*;

import java.util.*;

public class Main {

public static int dateDifference(String s1,String s2) throws ParseException{

        SimpleDateFormat sd=new SimpleDateFormat("yyyy-MM-dd");

        Date d=sd.parse(s1);

        Calendar c=Calendar.getInstance();

        c.setTime(d);

        long d1=c.getTimeInMillis();

        d=sd.parse(s2);

        c.setTime(d);

        long d2=c.getTimeInMillis();

        int n=Math.abs((int) ((d1-d2)/(1000*3600*24)));

        return n;

        }

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

        String s1="2012-03-12";

        String s2="2012-03-14";

        System.out.println(dateDifference(s1,s2));
        }

}
```

## 34.File Extension

Write a program to read a file name as a string and find out the file extension and return it as output. For example, the file sun.gif has the extension gif.

Include a class **UserMainCode** with a static method **fileIdentifier** which accepts a string. The return type (string) should return the extension of the input string (filename).

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 that corresponds to a file name.

Output consists of a string(extension of the input string (filename)).

Refer sample output for formatting specifications.

**Sample Input 1:**

sun.gif

**Sample Output 1:**

Gif

```
import java.util.*;

public class Main {

        public static String extensionString(String s1){

                StringTokenizer t=new StringTokenizer(s1,".");

                String ss=t.nextToken();

                String s2=t.nextToken();

                return s2;

        }

        public static void main(String[] args) {
                String s1="sun.gif";

                System.out.println(extensionString(s1));

        }

}
```

## 35.Find common characters and unique characters in string

Given a method with two strings as input. Write code to count the common and unique letters in the two strings.

Note:
- Space should not be counted as a letter.
- Consider letters to be case sensitive. ie, "a" is not equal to "A".

Include a class **UserMainCode** with a static method **commonChars** which accepts two strings as input.

The return type of the output is the count of all common and unique characters in the two strings.

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

**Input and Output Format:**

Input consists of two strings.

Output is an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

a black cow

battle ship

**Sample Output 1:**

2

[**Explanation** : b, l and a are the common letters between the 2 input strings. But 'a' appears more than once in the 1st string. So 'a' should not be considered while computing the count value.]

**Sample Input 2:**

australia
sri lanka

**Sample Output 2:**

4

```
import java.util.Arrays;

import java.util.StringTokenizer;

public class PO

{

public static int display(String s,String s1)

{

int c=0,m=0;String t=null;

char a[]=s.toCharArray();

char b[]=s1.toCharArray();

Arrays.sort(a);

Arrays.sort(b);

s=new String(a);

s1=new String(b);

StringTokenizer st=new StringTokenizer(s);

StringTokenizer st1=new StringTokenizer(s1);

s=st.nextToken();

s1=st1.nextToken();

if(s.length()>s1.length())

{t=s1;
s1=s;

s=t;

}

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

{

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

{

if(s.charAt(i)==s1.charAt(j))

{

if((s.indexOf(s.charAt(i))==s.lastIndexOf(s.charAt(i)))&&(s1.indexOf(s1.charAt(j))==s1.lastInd
exOf(s1.charAt(j))))

{

c++;

}

}}}

return c;



}

}
```

**36.)Initial Format**

Write a program to input a person's name in the format "FirstName LastName" and return the person name in the following format - "LastName, InitialOfFirstName".

Include a class **UserMainCode** with a static method **nameFormatter** which accepts a string. The return type (string) should return the expected format.

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 that corresponds to a Person's name.

Output consists of a string(person's name in expected format).

Refer sample output for formatting specifications.

**Sample Input :**
Jessica Miller
**Sample Output:**
Miller, J

```
import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

String s1="vishal jadiya";

getvalues(s1);

}

public static void getvalues(String s1) {

StringBuffer sb=new StringBuffer();

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

String s2=st.nextToken();

String s3=st.nextToken();

    sb.append(s3);

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

System.out.println(sb);

}

}
```

**37) Character cleaning**

Write a program to input a String and a character, and remove that character from the given String. Print the final string.
Include a class **UserMainCode** with a static method **removeCharacter** which accepts a string and a character. The return type (string) should return the character cleaned string.

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 and a character.

Output consists of a string(the character cleaned string).

Refer sample output for formatting specifications.

**Sample Input :**
elephant
e
**Sample Output:**
lphant

```
import java.util.Scanner;
public class Qus8Main {



          public static void main(String[]args){
          Scanner sc=new Scanner(System.in);
          String name=sc.nextLine();
               char ch=sc.nextLine().charAt(0);
          StringBuffer sb=new StringBuffer(name);
               for(int i=0;i<sb.length();i++)
               {if(ch==sb.charAt(i))
          {
          sb.deleteCharAt(i);
          i--;
          }
          }
          System.out.print(sb.toString());
          }}
```

**38) Vowel Check**

Write a program to read a String and check if that String contains all the vowels. Print “yes” if the string contains all vowels else print “no”.
Include a class **UserMainCode** with a static method **getVowels** which accepts a string. The return type (integer) should return 1 if the String contains all vowels else return -1.

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(“yes” or “no”).

Refer sample output for formatting specifications.

**Sample Input 1:**
abceiduosp
**Sample Output 1:**
yes

**Sample Input 2:**
bceiduosp
**Sample Output 2:**
No

```
import java.util.Scanner;

public class Qus8Main {


        public static void main(String[] name)
        {
                Scanner sc=new Scanner(System.in);
```

```
                        String s=sc.nextLine();
                        System.out.println(Qus8.display(s));
                }
}


public class Qus8 {
        public static int display(String name){

                String s1=name;
        int n1=0,n2=0,n3=0,n4=0,n5=0;
        for(int i=0;i<s1.length();i++){
        char c=s1.charAt(i);
        if(c=='a' || c=='A')
        n1++;
        if(c=='e' || c=='E')
        n2++;
        if(c=='i' || c=='I')
        n3++;
        if(c=='o' || c=='O')
        n4++;
        if(c=='u' || c=='U')
        n5++;}
        if(n1==1 && n2==1 && n3==1 && n4==1 && n5==1)
        return 1;
        else
        return 0 ;
        }

        }
```

## 39) Swap Characters

Write a program to input a String and swap the every 2 characters in the string. If size is an odd number then keep the last letter as it is. Print the final swapped string.
Include a class **UserMainCode** with a static method **swapCharacter** which accepts a string. The return type (String) should return the character swapped string.

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:**
TRAINER
**Sample Output 1:**

RTIAENR

**Sample Input 2:**
TOM ANDJERRY
**Sample output 2:**
OT MNAJDREYR

```
import java.util.Scanner;

public class Qus8Main {


      public static void main(String[] args) {
      String s1="TRAINER";
      getvalues(s1);
      }
      public static void getvalues(String s1)
      {
      StringBuffer sb=new StringBuffer();
      int l=s1.length();
      if(l%2==0)
      {
      for(int i=0;i<s1.length()-1;i=i+2)
      {
      char a=s1.charAt(i);
      char b=s1.charAt(i+1);
      sb.append(b)
         sb.append(a);
      }
      System.out.println(sb);
      }
      else
      {
      for(int i = 0;i<s1.length()-1;i=i+2)
      {
      char a=s1.charAt(i);
      char b=s1.charAt(i+1);
      sb.append(b).append(a);
      }
      sb.append(s1.charAt(l-1));
      System.out.println(sb);
      }
      }
      }
```

## 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
```