1. **Start Case**

Write a program to read a sentence in string variable and convert the first letter of each word to capital case. Print the final string.

Note: - Only the first letter in each word should be in capital case in final string.

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

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

Input and Output Format:

Input consists of a strings.

Output consists of a String (capitalized string).

Refer sample output for formatting specifications.

Sample Input:

Now is the time to act!

Sample Output:

Now Is The Time To Act!

Solution :

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s = sc.nextLine();
System.out.println(User.printCapitalized(s));
}
}
```

```
import java.util.StringTokenizer;

publicclass User {
publicstatic String printCapitalized(String s) {
StringTokenizer st= new StringTokenizer(s," ");
StringBuffer sb= new StringBuffer();
while(st.hasMoreTokens())
{
      String s1=st.nextToken();
      String s2=s1.substring(0, 1);
      String s3=s1.substring(1);
      sb.append(s2.toUpperCase());
      sb.append(s3);
      sb.append(" ");
```

```
}
return sb.toString();
}
}
```

```
import java.util.StringTokenizer;
publicclass UserMainCode {
        publicstatic String changeWord(String s)
        {
                StringTokenizer st=new StringTokenizer(s," ");
                StringBuffer sb=new StringBuffer();
                while(st.hasMoreTokens())
                {
                        String s1=st.nextToken();
                        sb.append(s1.substring(0,1).toUpperCase());
                        sb.append(s1.substring(1));
                        sb.append(" ");

                }
                return sb.toString();
        }
}
```

2. **Maximum Difference**

Write a program to read an integer array and find the index of larger number of the two adjacent numbers with largest difference. Print the index.

Include a class **UserMainCode** with a static method **findMaxDistance** which accepts an integer array and the number of elements in the array. The return type (Integer) should return index.

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

Input and Output Format:

Input consists of n+1 integers, where n corresponds the size of the array followed by n integers.

Output consists of an Integer (index).

Refer sample output for formatting specifications.

Sample Input :

6
4
8
6
1
9
4

Sample Output :

4

[In the sequence 4 8 6 1 9 4 the maximum distance is 8 (between 1 and 9). The function should return the index of the greatest of two. In this case it is 9 (which is at index 4). output = 4.]

Solution :

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int[] a= newint[n];
for(int i=0;i<n;i++)
 a[i]=sc.nextInt();

System.out.println(User.findMaxDistance(a));
}
}


publicclass User {
publicstaticint findMaxDistance(int[] a){

      int dif,max=0;
      int n=a.length;
      for(int i=0;i<n-1;i++)
      {
      dif=Math.abs(a[i]-a[i+1]);


      // if(max<dif)

if(dif>max)
      {
            if(a[i+1]>a[i])
            max=i+1;
            else
            max=i;
```

```
}

        }
        return max;
}
}
```

3. Palindrome - In Range

Write a program to input two integers, which corresponds to the lower limit and upper limit respectively, and find the sum of all palindrome numbers present in the range including the two numbers. Print the sum.

Include a class **UserMainCode** with a static method **addPalindromes** which accepts two integers. The return type (Integer) should return the sum if the palindromes are present, else return 0.

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

Note1 : A palindrome number is a number which remains same after reversing its digits.
Note2 : A single digit number is not considered as palindrome.

Input and Output Format:

Input consists of 2 integers, which corresponds to the lower limit and upper limit respectively.

Output consists of an Integer (sum of palindromes).

Refer sample output for formatting specifications.

Sample Input :
130
150
Sample Output :
272
(131+141 = 272)

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
intb = sc.nextInt();
```

```
System.out.println(User.addPalindromes(a,b));
}
}

publicclass User {
publicstaticint addPalindromes(int a, int b){

      int temp=0,sum=0,r,sum1=0;


      for(int i=a;i<=b;i++)
      {
          temp=i;
        sum=0;
while(temp>0){
             r=temp%10;
             sum=sum*10+r;
             temp=temp/10;
         }
if(i==sum)
          sum1=sum1+i;
   }
      return sum1;
}
}
```

4. **PAN Card**

Write a program to read a string and validate PAN no. against following rules:

1. There must be eight characters.
2. First three letters must be alphabets followed by four digit number and ends with alphabet
3. All alphabets should be in capital case.

Print “Valid” if the PAN no. is valid, else print “Invalid”.

Include a class **UserMainCode** with a static method **validatePAN** which accepts a string. The return type (Integer) should return 1 if the string is a valid PAN no. else return 2.

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

Input and Output Format:

Input consists of a string, which corresponds to the PAN number.

Output consists of a string - "Valid" or "Invalid"

Refer sample output for formatting specifications.

Sample Input 1:

ALD3245E

Sample Output 1:

Valid

Sample Input 2:

OLE124F

Sample Output 2:

Invalid

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s=sc.next();
int res=User.validatePAN(s);
if(res==1)
	System.out.println("Valid");
else
	System.out.println("Invalid");

}
}
```

```
publicclass User {
publicstaticint validatePAN(String s){

	int res=0;
	if(s.length()==8)
	{
	if(s.matches("[A-Z]{3}[0-9]{4}[A-Z]{1}"))
		res=1;
	else
		res=2;
	}
	return res;
}
}
```

5. **Fibonacci Sum**

Write a program to read an integer n, generate fibonacci series and calculate the sum of first n numbers in the series. Print the sum.

Include a class **UserMainCode** with a static method **getSumOfNfibos** which accepts an integer n. The return type (Integer) should return the sum of n fibonacci numbers.

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

Note: First two numbers in a Fibonacci series are 0, 1 and all other subsequent numbers are sum of its previous two numbers. Example - 0, 1, 1, 2, 3, 5...

Input and Output Format:

Input consists of an integer, which corresponds to n.

Output consists of an Integer (sum of fibonacci numbers).

Refer sample output for formatting specifications.

Sample Input :

5

Sample Output :

7

[0 + 1 + 1 + 2 + 3 = 7]

Solutions:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
int s=sc.nextInt();

      System.out.println(User.getSumOfNfibos(s));


}
}

publicclass User {
publicstaticint getSumOfNfibos(int s){


      int a=0,b=1,c=0,d=1;
      for(int i=3;i<=s;i++)
      {
            c=a+b;
            a=b;
            b=c;
            d=d+c;

      }
      returnd;
}
}
```

6. **Test Vowels**

Write a program to read a string and check if given string contains exactly five vowels in any order. Print “Yes” if the condition satisfies, else print “No”.
Assume there is no repetition of any vowel in the given string and all characters are lowercase.
Include a class **UserMainCode** with a static method **testVowels** which accepts a string. The return type (Integer) should return 1 if all vowels are present, else return 2.

Create a Class Main which would be used to accept a 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:
acbisouzze
Sample Output 1:
Yes

Sample Input 2:
cbisouzze
Sample Output 2:
No

Solutions:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s=sc.next();
int res=User.testVowels (s);
if(res==1)
      System.out.println("Yes");
else
      System.out.println("No");
}
}

publicclass User {
publicstaticint testVowels (String s){

      int res,count=0;
      String s1="aeiou";
```

```
        String s2=s.toLowerCase();
        for(int i=0;i<s2.length();i++)
        {
                for(int j=0;j<s1.length();j++)
                {
        if(s2.charAt(i)==s1.charAt(j))
        {
                count++;

        }
        }
        }

        if(count==s1.length())
                res=1;
        else
                res=2;
        return res;
}
}
```

```
publicclass User {
publicstaticint testOrderVowels(String s1) {

StringBuffer sb = newStringBuffer();
int res = 0;
for (int i = 0; i < s1.length(); i++) {
if (s1.charAt(i) == 'a' || s1.charAt(i) == 'A'
|| s1.charAt(i) == 'e' || s1.charAt(i) == 'E'
|| s1.charAt(i) == 'i' || s1.charAt(i) == 'I'
|| s1.charAt(i) == 'o' || s1.charAt(i) == 'O'
|| s1.charAt(i) == 'u' || s1.charAt(i) == 'U') {
sb.append(s1.charAt(i));
}
}
if (sb.toString().equals("aeiou"))
res = 1;
else
res = 0;
return res;
}
}
```

7 . Dash Check

Write a program to read two strings and check whether or not they have dashes in the same places. Print “Yes” if the condition satisfies, else print “No”.
Include a class **UserMainCode** with a static method **compareDashes** which accepts two strings. The return type (Integer) should return 1 if all dashes are placed correctly, else return 2.

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

Note: The strings must have exactly the same number of dashes in exactly the same positions. The strings might be of different length.
Input and Output Format:

Input consists of two strings.

Output consists of a string (“Yes” or “No”).

Refer sample output for formatting specifications.

Sample Input 1:
hi—there-you.
12--(134)-7539
Sample Output 1:
Yes

Sample Input 2:
-15-389
-xyw-zzy
Sample Output 2:
No

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s1=sc.next();
String s2=sc.next();
int res=User.compareDashes (s1,s2);
if(res==1)
      System.out.println("Yes");
else
      System.out.println("No");
}
}

import java.util.ArrayList;
```

```
publicclass User {
publicstaticint compareDashes (String s1,String s2){

int res=0;
      ArrayList<Integer> a1=new ArrayList<Integer>();
      ArrayList<Integer> a2=new ArrayList<Integer>();
      for(int i=0;i<s1.length();i++)
      {
            if(s1.charAt(i)=='-')
                  a1.add(i);
      }
      for(int i=0;i<s2.length();i++)
      {
            if(s2.charAt(i)=='-')
                  a2.add(i);
      }
      if(a1.equals(a2))
            res=1;
      else
            res=2;
      return res;

}
}
```

8. **Reverse Split**

Write a program to read a string and a character, and reverse the string and convert it in a format such that each character is separated by the given character. Print the final string.
Include a class **UserMainCode** with a static method **reshape** which accepts a string and a character. The return type (String) should return the final string.

Create a Class Main which would be used to accept a string and a character, 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 final string).

Refer sample output for formatting specifications.

Sample Input:
Rabbit
-
Sample Output:

t-i-b-b-a-R

```
importjava.util.HashMap;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s1=sc.next();
String s2=sc.next();
System.out.println(User.extractMax(s1,s2));

}
}
```

```
publicclass UserMain {
      publicstaticString extractMax(String s1,String s2){
            StringBuffer sb=new StringBuffer();
            for(int i=0;i<s1.length()-1;i++)
            {
                  sb.append(s1.charAt(i));
                  sb.append(s2);

            }
            sb.append(s1.charAt(s1.length()-1));
            return sb.reverse().toString();
            }
}
```

9. Remove 10's

Write a program to read an integer array and remove all 10s from the array, shift the other elements towards left and fill the trailing empty positions by 0 so that the modified array is of the same length of the given array.

Include a class **UserMainCode** with a static method **removeTens** which accepts the number of elements and an integer array. The return type (Integer array) should return the final array. Create a Class Main which would be used to read the number of elements and the input array, and call the static method present in UserMainCode.

Input and Output Format:

Input consists of n+1 integers, where n corresponds to size of the array followed by n elements of the array.

Output consists of an integer array (the final array).

Refer sample output for formatting specifications.

Sample Input :

5
1
10
20
10
2

Sample Output :

1
20
2
o
o

Solution :

```
 import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int[] a= newint[n];
for(int i=0;i<n;i++)
      a[i]=sc.nextInt();
User.removeTens(a);

}
}


importjava.util.ArrayList;


publicclass User {
publicstaticint[] removeTens(int[] a){
      int[] out=newint[a.length];
      int k=0;
for(int i=0;i<a.length;i++)
{
      if(a[i]!=10)
      {
            out[k]=a[i];
            k++;
}}
```

```
for(int i=0;i<a.length;i++)
      System.out.println(out[i]);
return out;
}
}
```

10. Last Letters

Write a program to read a sentence as a string and store only the last letter of each word of the sentence in capital letters separated by $. Print the final string.

Include a class **UserMainCode** with a static method **getLastLetter** which accepts a string. The return type (string) should return the final string.
Create a Class Main which would be used to read a string, and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.

Output consists of a string (the final string).

Refer sample output for formatting specifications.

Smaple Input :
This is a cat
Sample Output :
SSA$T

Solution :

```
importjava.util.ArrayList;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s= sc.nextLine();
System.out.println(User.getLastLetter(s));

}
}

import java.util.StringTokenizer;


publicclass User {
```

```
publicstatic String getLastLetter(String s){

	StringTokenizer st= new StringTokenizer(s," ");
	String s2=st.nextToken();
	StringBuffer sb= new StringBuffer();
	String s3=Character.toUpperCase(s2.charAt(s2.length()-1))+"";
	while(st.hasMoreTokens())
	{
	 s2=st.nextToken();
	      s3=s3+"$"+Character.toUpperCase(s2.charAt(s2.length()-1));

	}
	returns3;
}
}
```

****
****

```
publicclass UserMain {
	publicstatic String getLastLetter(String s)
	{
	StringTokenizer st= new StringTokenizer(s," ");

	StringBuffer sb= new StringBuffer();

	String b=st.nextToken();
	sb.append(b.charAt(b.length()-1));

	while(st.hasMoreTokens())
	{
	String a=st.nextToken();
	sb.append("$");
	sb.append(a.charAt(a.length()-1));

	}
	return sb.toString().toUpperCase();

}
}
```

****
****