1.Check Sum of Odd Digits

Write a program to read a number , calculate the sum of odd digits (values) present in the given number.

Include a class **UserMainCode** with a static method **checkSum** which accepts a positive integer . The return type should be 1 if the sum is odd . In case the sum is even return -1 as output.

Create a class **Main** which would get the input as a positive integer and call the static method **checkSum** present in the UserMainCode.

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

Sample Input 1:

56895

Sample Output 1:

Sum of odd digits is odd.

Sample Input 2:

84228

Sample Output 2:

Sum of odd digits is even.

```
public class UserMainCode {
	public static int SumOfOddsAndEvens(int n){
		int n1,n2=0,n3;
		while(n!=0)
		{
			n1=n%10;
			if((n1%2)!=0)
				n2+=n1;
			n/=10;
		}
		if(n2%2==0)
			n3=-1;
```

```
                else
                        n3=1;


                return n3;
        }
        public static void main(String[] args) {
                int n=84882;
                System.out.println(SumOfOddsAndEvens(n));
        }
}
```

2.Number Validation

Write a program to read a string of 10 digit number , check whether the string contains a 10 digit number in the format XXX-XXX-XXXX where 'X' is a digit.

Include a class **UserMainCode** with a static method **validateNumber** which accepts a string as input .
The return type of the output should be 1 if the string meets the above specified format . In case the number does not meet the specified format then return -1 as output.

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

Input and Output Format:

Input consists of a string.

Output is a string specifying the given string is valid or not .

Refer sample output for formatting specifications.

Sample Input 1:

123-456-7895

Sample Output 1:

Valid number format

Sample Input 2:

-123-12344322

Sample Output 2:

Invalid number format

```
import java.util.*;
public class Main {
public static void main(String[] args) {
	Scanner s=new Scanner(System.in);
	String pan=s.next();
	int b=panNumberValidation(pan);
	if(b==1)
		System.out.println("valid Pancard Number");
	else
		System.out.println("not a valid credential");
}
public static int panNumberValidation(String input) {
	int b=0;
	if(input.matches("[0-9]{3}[-]{1}[0-9]{3}[-]{1}[0-9]{4}"))
	{b=1;}
	else
		b=0;
	return b;
}
}
```

3.Sum of Squares of Even Digits

Write a program to read a number , calculate the sum of squares of even digits (values) present in the given number.

Include a class **UserMainCode** with a static method **sumOfSquaresOfEvenDigits** which accepts a positive integer . The return type (integer) should be the sum of squares of the even digits.

Create a class **Main** which would get the input as a positive integer and call the static method sumOfSquaresOfEvenDigits present in the UserMainCode.

Input and Output Format:

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

Sample Input 1:

56895

Sample Output 1:

100

```
public class UserMainCode

{



        public static int display(int number){

            int n1=0,n2=0;

            while(number!=0)

            {

            n1=number%10;

            if((n1%2)==0)
```

```
            n2+=n1*n1;
            number/=10;
            }
            return n2;
        }
    }
```

4.Fetching Middle Characters from String

Write a program to read a string of even length and to fetch two middle most characters from the input string and return it as string output.

Include a class **UserMainCode** with a static method **getMiddleChars** which accepts a string of even length as input . The return type is a string which should be the middle characters of the string.

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

Input and Output Format:

Input consists of a string of even length.

Output is a string .

Refer sample output for formatting specifications.

Sample Input 1:

this

Sample Output 1:

hi

Sample Input 1:

Hell

Sample Output 1:

el

```
import java.util.Scanner;


public class Middle {


	public static void main(String[] args) {


		Scanner sc=new Scanner(System.in);

		String s=sc.nextLine();
		StringBuffer sb=new StringBuffer();
		if(s.length()%2==0)
		{
		sb.append(s.substring(s.length()/2-1,s.length()/2+1));
		//System.out.println(sb.toString());

		}
		System.out.println(sb.toString());
	}

}
```

5.Check Characters in a String

Write a program to read a string and to test whether first and last character are same. The string is said to be be valid if the 1st and last character are the same. Else the string is said to be invalid.

Include a class **UserMainCode** with a static method **checkCharacters** which accepts a string as input .
The return type of this method is an int. Output should be 1 if the first character and last character are same . If they are different then return -1 as output.

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

Input and Output Format:

Input consists of a string.

Output is a string saying characters are same or not .

Refer sample output for formatting specifications.

Sample Input 1:

the picture was great

Sample Output 1:

Valid

Sample Input 1:

this

Sample Output 1:

Invalid

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


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
int res=UserMainCode.checkCharacter(s);
if(res==1)
{
	System.out.println("Valid");

}
else
	System.out.println("Invalid");
	}


	}




public class UserMainCode {
	public static int checkCharacter(String s)

	{
		int res=-1;
		if(s.charAt(0)==s.charAt(s.length()-1))
		{
		res=1;
```

```
            }
                return res;
        }
}
```

6.Forming New Word from a String

Write a program to read a string and a positive integer n as input and construct a string with first n and last n characters in the given string.

Include a class **UserMainCode** with a static method **formNewWord** which accepts a string and positive integer .

The return type of the output should be a string (value) of first n character and last n character.

Create a class **Main** which would get the input as a string and integer n and call the static method **formNewWord**present in the UserMainCode.

Input and Output Format:

Input consists of a string of even length.

Output is a string .

Note: The given string length must be >=2n.

Refer sample output for formatting specifications.

Sample Input 1:

California
3

Sample Output 1:

Calnia

Sample Input 2:

this
1

Sample Output 2:

ts

```
import java.util.Scanner;
```

```
public class Main {


        public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
int n=sc.nextInt();
System.out.println(UserMainCode.stringChange(s,n));



}
}

public class UserMainCode {
        public static String stringChange(String s,int n)
        {


                StringBuffer sb=new StringBuffer();
           sb.append(s.substring(0,n));
           sb.append(s.substring(s.length()-n));
           return sb.toString();
        }
}
```

7.Reversing a Number

Write a program to read a positive number as input and to get the reverse of the given number and return it as output.

Include a class **UserMainCode** with a static method **reverseNumber** which accepts a positive integer .

The return type is an integer value which is the reverse of the given number.

Create a **Main** class which gets the input as a integer and call the static method **reverseNumber** present in the**UserMainCode**

Input and Output Format:

Input consists of a positive integer.

Output is an integer .

Refer sample output for formatting specifications.

Sample Input 1:

543

Sample Output 1:

345

Sample Input 1:

1111

Sample Output 1:

1111

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

public static void main(String[] args)
{

Scanner sc=new Scanner(System.in);
int a=sc.nextInt();
System.out.println(UserMainCode.reverse(a));

}
}
```

```
public class UserMainCode
```

```
{
        public static int reverse(int a)
        {

        String s=String.valueOf(a);
        StringBuffer sb=new StringBuffer(s);
    sb.reverse();   //reverse return type is void
    int res=Integer.parseInt(sb.toString());
    return res;
        }
}
```

8.Array List Sorting and Merging

Write a code to read two int array lists of size 5 each as input and to merge the two arrayLists, sort the merged arraylist in ascending order and fetch the elements at 2nd, 6th and 8th index into a new arrayList and return the final ArrayList.

Include a class **UserMainCode** with a static method **sortMergedArrayList** which accepts 2 ArrayLists.

The return type is an ArrayList with elements from 2,6 and 8th index position .Array index starts from position 0.

Create a **Main** class which gets two array list of size 5 as input and call the static method **sortMergedArrayList**present in the **UserMainCode.**

Input and Output Format:

Input consists of two array lists of size 5.

Output is an array list .

Note - The first element is at index 0.

Refer sample output for formatting specifications.

Sample Input 1:

3
1
17
11
19
5
2
7
6
20

Sample Output 1:

3

11

19

Sample Input 2:

1

2

3

4

5

6

7

8

9

10

Sample Output 2:

3

7

9

9.Validating Date Format

Obtain a date string in the format dd/mm/yyyy. Write code to validate the given date against the given format.

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

The return type of the validateDate method is 1 if the given date format matches the specified format , If the validation fails return the output as -1.

Create a **Main** class which gets date string as an input and call the static method **validateDate** present in the**UserMainCode.**

Input and Output Format:

Input is a string .

Refer sample output for formatting specifications

Sample Input 1:

12/06/1987
Sample Output 1:
Valid date format

Sample Input 2:

03/1/1987
Sample Output 2:
Invalid date format

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Qus8Main {

	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
```

```
sdf.setLenient(false);
int res=0;
if(s.matches("[0-9]{2}(/)[0-9]{2}(/)[0-9]{4}"))
{

	try {
		Date d=sdf.parse(s);
		res=1;
	} catch (ParseException e) {
		res=-1;
	}

System.out.println(res);

	}
}
}
```

10.Validate Time

Obtain a time string as input in the following format 'hh:mm am' or 'hh:mm pm'. Write code to validate it using the following rules:

- It should be a valid time in 12 hrs format
- It should have case insensitive AM or PM

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

If the given time is as per the given rules then return 1 else return -1.If the value returned is 1 then print as valid time else print as Invalid time.

Create a **Main** class which gets time(string value) as an input and call the static method **validateTime** present in the**UserMainCode.**

Input and Output Format:

Input is a string .

Output is a string .

Sample Input 1:

09:59 pm

Sample Output 1:

Valid time

Sample Input 2:

10:70 AM

Sample Output 2:

Invalid time

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Qus8Main {


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
SimpleDateFormat sdf=new SimpleDateFormat("hh:mm a");
sdf.setLenient(false);
int res=0;

	try {
		Date d=sdf.parse(s);
		res=1;
	} catch (ParseException e) {
		res=-1;
	}

System.out.println(res);


}
}
```