**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;

public class Main {

public static void main(String[] args) {

String s1="03/01/1987";

getvalues(s1);

}

public static void getvalues(String s1) {

if(s1.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(s1);

System.out.println(1);

} catch (ParseException e) {

System.out.println(-1);

}

}

else

System.out.println(-1);

}}

**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.SimpleDateFormat;

import java.util.Date;

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main{

public static void main(String []args){

Scanner sc=new Scanner(System.in);

String str=sc.nextLine();

StringTokenizer st=new StringTokenizer(str,":");

if(st.countTokens()==3)

{

SimpleDateFormat sdf1 = new SimpleDateFormat("h:mm:ss a");

sdf1.setLenient(false);

try

{

Date d2=sdf1.parse(str);

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

}

catch(Exception e)

{

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

}

}

else

{

SimpleDateFormat sdf = new SimpleDateFormat("h:mm a");

sdf.setLenient(false);

try

{

Date d1=sdf.parse(str);

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

}

catch(Exception e){

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

}

}

}

**30.Find the difference between Dates in months**

Given a method with two date strings in yyyy-mm-dd format as input. Write code to find the difference between two dates in months.

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

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

Create a class **Main** which would get the input and call the static method **getMonthDifference** 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:**

2012-03-01

2012-04-16

**Sample Output 1:**

1

**Sample Input 2:**

2011-03-01

2012-04-16

**Sample Output 2:**

13

import java.text.\*;

import java.util.\*;

public class Main {

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

                String s1="2012-03-01";

                String s2="2012-03-16";

                System.out.println(monthsBetweenDates(s1,s2));

}

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

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

                                Date d1=sdf.parse(s1);

                                Date d2=sdf.parse(s2);

                Calendar cal=Calendar.getInstance();

                cal.setTime(d1);

                                int months1=cal.get(Calendar.MONTH);

                                int year1=cal.get(Calendar.YEAR);

                cal.setTime(d2);

                                int months2=cal.get(Calendar.MONTH);

                                int year2=cal.get(Calendar.YEAR);

                int n=((year2-year1)\*12)+(months2-months1);

                return n;

}

}

**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));

                }

}

**51.Finding the day of birth**

Given an input as date of birth of person, write a program to calculate on which day (MONDAY,TUESDAY....) he was born store and print the day in Upper Case letters.

Include a class **UserMainCode** with a static method **calculateBornDay** which accepts a string as input.

The return type of the output is a string which should be the day in which the person was born.

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

**Input and Output Format:**

NOTE: date format should be(dd-MM-yyyy)  
Input consists a date string.

Output is a string which the day in which the person was born.

Refer sample output for formatting specifications.

**Sample Input 1:**

29-07-2013

**Sample Output 1:**

MONDAY

**Sample Input 2:**

14-12-1992

**Sample Output 2:**

MONDAY

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Scanner;

public class Main {

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

                                Scanner sc=new Scanner(System.in);

                    String s1=sc.nextLine();

                    System.out.println(UserMainCode.calculateBornDay(s1));

                                }

                                }

import java.text.SimpleDateFormat;

import java.text.ParseException;

import java.util.Date;

public class UserMainCode {

                public static String calculateBornDay(String s1) throws ParseException

                {

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

                                SimpleDateFormat sdf1=new SimpleDateFormat("EEEEE");

                                Date d=sdf.parse(s1);

                                String s=sdf1.format(d);

                                return s.toUpperCase();

                }

}

**53.Experience Calculator**

Write a program to read Date of Joining and current date as Strings and Experience as integer and validate whether the given experience and calculated experience are the same. Print “true” if same, else “false”.

Include a class **UserMainCode** with a static method **calculateExperience**which accepts 2 strings and an integer. The return type is boolean.

Create a Class Main which would be used to accept 2 string (dates) and an integer and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of 2 strings and an integer, where the 2 strings corresponds to the date of joining and current date, and the integer is the experience.

Output is either “true” or “false”.

Refer sample output for formatting specifications.

**Sample Input 1:**

11/01/2010

01/09/2014

4

**Sample Output 1:**

true

**Sample Input 2:**

11/06/2009

01/09/2014

4

**Sample Output 2:**

False

import java.util.Date;

import java.text.SimpleDateFormat;

public class Usermaincode

{public static boolean display(String s,String s1,int n)

{

boolean b=false;

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

try{

Date d=sdf.parse(s);

Date d1=sdf.parse(s1);

int y=d.getYear();

int y1=d1.getYear();

int m=d.getMonth();

int m1=d1.getMonth();

int day=d.getDay();

int day1=d1.getDay();

int age=y1-y;

if(m>m1)

age--;

else if(m==m1)

{if(day<day1)

age--;

}

if(age==n)

b=true;

else

b=false;

}

catch(Exception e)

{e.printStackTrace();

}

return b;

}

}

**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

import java.util.ArrayList;

import java.util.Collections;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Main {

public static void main(String[] args) {

String s="14#09/1991";

getvalues(s);

}

public static void getvalues(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);

System.out.println(1);

} catch (ParseException e) {

System.out.println(-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);

System.out.println(1);

} catch (ParseException e) {

System.out.println(-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);

System.out.println(1);

} catch (ParseException e) {

System.out.println(-1);

}

}

else

System.out.println(-1);

}

}

**63.Month Name**

Given a date as a string input in the format dd-mm-yy, write a program to extract the month and to print the month name in upper case.

Include a class **UserMainCode** with a static method “**getMonthName**” that accepts a String argument and returns a String that corresponds to the month name.

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

The month names are {JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER}

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input:**

01-06-82

**Sample Output:**

JUNE

import java.text.ParseException;

import java.util.Scanner;

public class Main {

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

                                Scanner sc=new Scanner(System.in);

                    String s1=sc.nextLine();

                    System.out.println(User.calculateBornDay(s1));

                                }

                                }

import java.text.SimpleDateFormat;

import java.text.ParseException;

import java.util.Date;

public class User {

                public static String calculateBornDay(String s1) throws ParseException

                {

                                SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yy");

                                SimpleDateFormat sdf1=new SimpleDateFormat("MMMM");

                                Date d=sdf.parse(s1);

                                String s=sdf1.format(d);

                                return s.toUpperCase();

                }

}

**67.Month : Number of Days**

Given two inputs year and month (Month is coded as: Jan=0, Feb=1 ,Mar=2 ...), write a program to find out total number of days in the given month for the given year.

Include a class **UserMainCode** with a static method “**getNumberOfDays**” that accepts 2 integers as arguments and returns an integer. The first argument corresponds to the year and the second argument corresponds to the month code. The method returns an integer corresponding to the number of days in the month.

Create a class **Main** which would get 2 integers as input and call the static method **getNumberOfDays** present in the UserMainCode.

**Input and Output Format:**

Input consists of 2 integers that correspond to the year and month code.

Output consists of an integer that correspond to the number of days in the month in the given year.

**Sample Input:**

2000

1

**Sample Output:**

29

 import java.util.\*;

public class MainDate{

     public static void main(String []args){

                Scanner sc=new Scanner(System.in);

                int year=sc.nextInt();

                int month=sc.nextInt();

                System.out.println(main13.display(year, month));

     }

}

import java.util.Calendar;

public class main13{

     public static int display(int year,int month)

     {

                 Calendar cal=Calendar.getInstance();

                 cal.set(Calendar.YEAR,year);

                 cal.set(Calendar.MONTH,month);

                 int DAY\_OF\_MONTH=cal.getActualMaximum(cal.DAY\_OF\_MONTH);

                return DAY\_OF\_MONTH;

     }}

**13.Day of the Week**

Write a program to read a date as string (MM-dd-yyyy) and return the day of week on that date.

Include a class UserMainCode with a static method **getDay** which accepts the string. The return type (string) should be the day of the week.

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:**

07-13-2012

**Sample Output 1:**

Friday

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Scanner;

public class Main {

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

                                Scanner sc=new Scanner(System.in);

                    String s1=sc.nextLine();

                    System.out.println(User.calculateBornDay(s1));

                                }

                                }

import java.text.SimpleDateFormat;

import java.text.ParseException;

import java.util.Date;

public class User {

                public static String calculateBornDay(String s1) throws ParseException

                {

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

                                SimpleDateFormat sdf1=new SimpleDateFormat("EEEEE");

                                Date d=sdf.parse(s1);

                                String s=sdf1.format(d);

                                return s;

                }

}

**21.Date Format Conversion**

Given a date string in the format dd/mm/yyyy, write a program to convert the given date to the format dd-mm-yy.

Include a class **UserMainCode** with a static method “**convertDateFormat**” that accepts a String and returns a String.

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

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input:**

12/11/1998

**Sample Output:**

12-11-98

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Main {

public static void main(String[] args) {

String s1="12/12/1998";

getvalues(s1);

}

public static void getvalues(String s1) {

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

sdf.setLenient(false);

try {

Date d1=sdf.parse(s1);

SimpleDateFormat sdf1=new SimpleDateFormat("dd-MM-yy");

String s2=sdf1.format(d1);

System.out.println(s2);

} catch (ParseException e) {

e.printStackTrace();

}

}

}

**25.Next Year day**

Given a date string in dd/mm/yyyy format, write a program to calculate the day which falls on the same date next year. Print the output in small case.

The days are sunday, monday, tuesday, wednesday, thursday, friday and saturday.

Include a class **UserMainCode** with a static method “**nextYearDay**” that accepts a String and returns a String.

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

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input:**

13/07/2012

**Sample Output:**

saturday

 import java.util.ArrayList;import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

public class Main {

public static void main(String[] args) {

String s1="29/02/2000";

getvalues(s1);

}

public static void getvalues(String s1) {

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

sdf.setLenient(false);

try {

Date d1=sdf.parse(s1);

Calendar cal=Calendar.getInstance();

cal.setTime(d1);

cal.add(Calendar.YEAR, 1);

Date d2=cal.getTime();

SimpleDateFormat sdf1=new SimpleDateFormat("EEEEE");

String s=sdf1.format(d2);

System.out.println(s);

}

catch (ParseException e)

{

                e.printStackTrace();

}

}

}

**54.DOB - Validation**

Write a program to validate the Date of Birth given as input in String format (MM/dd/yyyy) as per the validation rules given below. Return true for valid dates else return false.  
1. Value should not be null  
2. month should be between 1-12, date should be between 1-31 and year should be a four digit number.  
Include a class UserMainCode with a static method **ValidateDOB** which accepts the string. The return type is TRUE / FALSE.  
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 string.  
Output consists of TRUE / FALSE.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
12/23/1985  
  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
31/12/1985  
  
**Sample Output 2:**  
FALSE

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Scanner;

public class UserMainCode {

public static void main(String[] args)

{

String str=new String();

Scanner sc=new Scanner(System.in);

str=sc.nextLine();

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

sdf.setLenient(false);

try

{

Date d1=sdf.parse(str);

System.out.println("TRUE");

}

catch(Exception e)

{

System.out.println("FALSE");

}

}

}

**61.String Processing - ZigZag**

Write a program to read a string containing date in DD-MM-YYYY format. find the number of days in the given month.  
  
Note - In leap year February has got 29 days.  
  
Include a class UserMainCode with a static method **getLastDayOfMonth** which accepts the string. The return type is the integer having number of days.  
  
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 string.  
Output consists of integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
12-06-2012  
**Sample Output 1:**  
30  
  
**Sample Input 2:**  
10-02-2012  
**Sample Output 2:**  
29

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class User {

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

                                // TODO Auto-generated method stub

                String s1="10-02-2012";

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

                Calendar cal=Calendar.getInstance();

                Date d1=sdf.parse(s1);

                cal.setTime(d1);

                int n=cal.getActualMaximum(Calendar.DAY\_OF\_MONTH);

                System.out.println(n);

                }

}

**62.Leap Year**

Write a program to read a string containing date in DD/MM/YYYY format and check if its a leap year. If so, return true else return false.  
  
Include a class UserMainCode with a static method **isLeapYear** which accepts the string. The return type is the boolean indicating TRUE / FALSE.  
  
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 string.  
  
Output consists of TRUE / FALSE.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
  
23/02/2012  
  
**Sample Output 1:**  
  
TRUE  
  
**Sample Input 2:**  
  
12/12/2011  
  
**Sample Output 2:**  
  
FALSE

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class UserMainCode {

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

                                String s1="23/02/2012";

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

                                GregorianCalendar g=new GregorianCalendar();

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

                                String s2=t.nextToken();

                                String s3=t.nextToken();

                                String s4=t.nextToken();

                                int n1=Integer.parseInt(s4);

                                Date d1=sdf.parse(s1);

                                boolean b=g.isLeapYear(n1);

                                System.out.println(b);

                                }

}

**38.Day of Week**

Write a program to read a string  containing date in DD/MM/YYYY format and prints the day of the week that date falls on.  
Return the day in lowercase letter (Ex: monday)  
  
Include a class UserMainCode with a static method **getDayOfWeek** which accepts the string. The return type is the string.  
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 string.  
Output consists of a string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
02/04/1985  
**Sample Output 1:**  
Tuesday

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Scanner;

public class Main {

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

                                Scanner sc=new Scanner(System.in);

                    String s1=sc.nextLine();

                    System.out.println(User.calculateBornDay(s1));

                                }

                                }

import java.text.SimpleDateFormat;

import java.text.ParseException;

import java.util.Date;

public class User {

                public static String calculateBornDay(String s1) throws ParseException

                {

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

                                SimpleDateFormat sdf1=new SimpleDateFormat("EEEEE");

                                Date d=sdf.parse(s1);

                                String s=sdf1.format(d);

                                return s.toLowerCase();

                }

}

**39.Add Time**

Write a program to read  two String variables containing time intervals in hh:mm:ss format. Add the two time intervals and return a string in days:hours:minutes:seconds format where DD is number of days.  
Hint: Maximum value for hh:mm:ss is 23:59:59  
  
Include a class UserMainCode with a static method **addTime** which accepts the string values. The return type is the string.  
Create a Class Main which would be used to accept the two string values and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of two string.  
Output consists of a string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
12:45:30  
13:50:45  
**Sample Output 1:**  
1:2:36:15  
  
**Sample Input 2:**  
23:59:59  
23:59:59  
**Sample Output 2:**  
1:23:59:58  
public class Main {

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

                String s1="23:59:59";

                String s2="23:59:59";

                SimpleDateFormat sdf=new SimpleDateFormat("HH:mm:ss");

                sdf.setTimeZone(TimeZone.getTimeZone("UTC"));

                sdf.setTimeZone(TimeZone.getTimeZone("s1"));

                sdf.setTimeZone(TimeZone.getTimeZone("s2"));

                Date d1=sdf.parse(s1);

                Date d2=sdf.parse(s2);

                long add=d1.getTime()+d2.getTime();

                String s=sdf.format(add);

                Calendar cal=Calendar.getInstance();

                cal.setTime(sdf.parse(s));

                int FindDay=cal.get(Calendar.DAY\_OF\_MONTH);

                if(FindDay>1)

                FindDay=FindDay-1;

                String op=FindDay+":"+s;

                System.out.println(op);

                }

}

**41.Date Format**

Write a program to read  two String variables in DD-MM-YYYY.Compare the two dates and return the older date in 'MM/DD/YYYY' format.  
  
Include a class UserMainCode with a static method **findOldDate** which accepts the string values. The return type is the string.  
Create a Class Main which would be used to accept the two string values and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of two string.  
Output consists of a string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
05-12-1987  
8-11-2010  
**Sample Output 1:**  
12/05/1987

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class Main {

                /\*\*

                 \* @param args

                 \* @throws ParseException

                 \*/

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

                                // TODO Auto-generated method stub

                                String s1="12/07/1994";

                                String s2="12/07/1995";

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

                                SimpleDateFormat sdf1=new SimpleDateFormat("MM-dd-yyyy");

                                Date d1=sdf.parse(s1);

                                Date d2=sdf.parse(s2);

                                Calendar cal=Calendar.getInstance();

                                cal.setTime(d1);

                                long y=cal.getTimeInMillis();

                                cal.setTime(d2);

                                long y1=cal.getTimeInMillis();

                                String s3=sdf1.format(d1);

                                String s4=sdf1.format(d2);

                                if(y<y1)

                                                System.out.println(s3);

                                else

                                                System.out.println(s4);

                }

}