**Problem Statements for Practice**

1. **Problem 1: Time Zones and ZonedDateTime** Write a program that displays the current time in different time zones:

➢ GMT (Greenwich Mean Time)

➢ IST (Indian Standard Time)

➢ PST (Pacific Standard Time)

**Hint**: Use ZonedDateTime and ZoneId to work with different time zones.

import java.time.\*;

import java.time.format.\*;

class ZonesTime{

public static void main(String[] args) {

ZoneId gmtid = ZoneId.of("GMT");

ZoneId istid = ZoneId.of("Asia/Kolkata");

ZoneId pstid = ZoneId.of("America/Los\_Angeles");

ZonedDateTime gmt = ZonedDateTime.now(gmtid);

ZonedDateTime ist = ZonedDateTime.now(istid);

ZonedDateTime pst = ZonedDateTime.now(pstid);

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

System.out.println("GMT - " + gmt.format(formatter));

System.out.println("IST - " + ist.format(formatter));

System.out.println("PST - " + pst.format(formatter));

}

}

2. **Problem 2: Date Arithmetic** Create a program that:

➢ Takes a date input and adds 7 days, 1 month, and 2 years to it.

➢ Then subtracts 3 weeks from the result.

**Hint**: Use LocalDate.plusDays(), plusMonths(), plusYears(), and minusWeeks() methods.

import java.util.Scanner;

import java.time.\*;

import java.time.format.\*;

class DataArithmetic{

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

String indate = input.nextLine();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

LocalDate date = LocalDate.parse(indate, formatter);

LocalDate newdate = date.plusDays(7);

newdate = newdate.plusMonths(1);

newdate = newdate.plusYears(2);

newdate = newdate.minusWeeks(3);

System.out.println(newdate);

input.close();

}

}

3. **Problem 3: Date Formatting** Write a program that:

➢ Displays the current date in three different formats:

■ dd/MM/yyyy

■ yyyy-MM-dd

■ EEE, MMM dd, yyyy

**Hint**: Use DateTimeFormatter with custom patterns for date formatting.

import java.time.\*;

import java.time.format.\*;

class DateFormatting{

public static void main(String[] args) {

LocalDate date = LocalDate.now();

DateTimeFormatter format1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");

DateTimeFormatter format2 = DateTimeFormatter.ofPattern("yyyy-MM-dd");

DateTimeFormatter format3 = DateTimeFormatter.ofPattern("EEE, MMM dd, yyyy");

System.out.println(date.format(format1));

System.out.println(date.format(format2));

System.out.println(date.format(format3));

}

}

4. **Problem 4: Date Comparison** Write a program that:

➢ Takes two date inputs and compares them to check if the first date is before, after, or the same as the second date.

**Hint**: Use isBefore(), isAfter(), and isEqual() methods from the LocalDate

Class.

import java.util.Scanner;

import java.time.\*;

import java.time.format.\*;

class DataComparision{

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter a date : ");

String indate1 = input.nextLine();

System.out.print("Enter a date : ");

String indate2 = input.nextLine();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

LocalDate date1 = LocalDate.parse(indate1, formatter);

LocalDate date2 = LocalDate.parse(indate2, formatter);

if(date1.isAfter(date2)){

System.out.println("first date is after second date");

}

else if(date1.isBefore(date2)){

System.out.println("first date is before second date");

}

else System.out.println("first date and second date are same ");

input.close();

}

}