자바에서 날짜 다루기

1. 오늘의 날짜 구하기

자바를 이용하여 오늘의 날짜를 알아내는 방법은 여러가지가 있다.

```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Calendar;
import java.util.Date;

public class Ex01_getCurrentDate {
    // 현재의 날짜 알아내기
    @SuppressWarnings("deprecation")
    public static void main(String[] args) {
        // Java 7 이전
        Date date = new Date();
        System.out.println("현재 : " + date);
        System.out.println(String.format("현재 : %4d-%02d-%02d",
                date.getYear()+1900, date.getMonth()+1, date.getDate()));
        Calendar calendar = Calendar.getInstance();
        System.out.println(String.format("현재 : %4d-%02d-%02d",
                                         calendar.get(Calendar.YEAR),
                                         calendar.get(Calendar.MONTH)+1,
                                         calendar.get(Calendar.DAY_OF_MONTH)));

        // 정해진 표시형식으로
        DateFormat dateFormat1 = DateFormat.getDateInstance(DateFormat.LONG);
        System.out.println("현재 : " + dateFormat1.format(date));
        DateFormat dateFormat2 = DateFormat.getDateInstance(DateFormat.FULL);
        System.out.println("현재 : " + dateFormat2.format(calendar.getTime()));
        DateFormat dateFormat3 = DateFormat.getTimeInstance(DateFormat.LONG);
        System.out.println("현재 : " + dateFormat3.format(date));
        DateFormat dateFormat4 = DateFormat.getTimeInstance(DateFormat.FULL);
        System.out.println("현재 : " + dateFormat4.format(calendar.getTime()));
        DateFormat dateFormat5 = DateFormat.getDateTimeInstance(DateFormat.LONG,
DateFormat.MEDIUM);
        System.out.println("현재 : " + dateFormat5.format(calendar.getTime()));

        // 사용자 정의 표시형식으로
        SimpleDateFormat simpleDateFormat1 = new SimpleDateFormat();
        System.out.println("현재 : " + simpleDateFormat1.format(date));
        SimpleDateFormat simpleDateFormat2 = new SimpleDateFormat("y-MM-dd(E) a
hh:mm:ss");
        System.out.println("현재 : " +
simpleDateFormat2.format(calendar.getTime()));
        System.out.println();

        // Java 8 이후
```

```
        LocalDate localDate = LocalDate.now();
        LocalTime localTime = LocalTime.now();
        LocalDateTime localDateTime = LocalDateTime.now();
        System.out.println("현재 : " + localDate);
        System.out.println("현재 : " + localTime);
        System.out.println("현재 : " + localDateTime);

        // 정해진 표시형식으로
        System.out.println("현재 : " +
DateTimeFormatter.ISO_DATE.format(localDateTime));
        System.out.println("현재 : " +
localDateTime.format(DateTimeFormatter.ISO_DATE));
        System.out.println("현재 : " +
DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL).format(localDateTime));
        System.out.println("현재 : " +
DateTimeFormatter.ofLocalizedDate(FormatStyle.LONG).format(localDateTime));

        // 사용자 정의 표시형식으로
        String pattern1 = "yyyy'년' MM'월' dd'일('EEEE')'";
        DateTimeFormatter formatter1 = DateTimeFormatter.ofPattern(pattern1);
        System.out.println("현재 : " + formatter1.format(localDate));
        String pattern2 = "yyyy'년' MM'월' dd'일('E')' a hh:mm:ss";
        DateTimeFormatter formatter2 = DateTimeFormatter.ofPattern(pattern2);
        System.out.println("현재 : " + formatter2.format(localDateTime));
    }
}
```

결과

```
현재 : Fri Oct 08 15:51:20 KST 2021
현재 : 2021-10-08
현재 : 2021-10-08
현재 : 2021년 10월 8일
현재 : 2021년 10월 8일 금요일
현재 : 오후 3시 51분 20초 KST
현재 : 오후 3시 51분 21초 대한민국 표준시
현재 : 2021년 10월 8일 오후 3:51:21
현재 : 21. 10. 8. 오후 3:51
현재 : 2021-10-08(금) 오후 03:51:21

현재 : 2021-10-08
현재 : 15:51:21.244923900
현재 : 2021-10-08T15:51:21.244923900
현재 : 2021-10-08
현재 : 2021-10-08
현재 : 2021년 10월 8일 금요일
현재 : 2021년 10월 8일
현재 : 2021년 10월 08일(금요일)
현재 : 2021년 10월 08일(금) 오후 03:51:21
```

2. 어제 오늘 그리고 내일 구하기

```
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Period;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;

public class Ex02_YesterdayTodayTomorrow {
    public static void main(String[] args) {
        // Java 7 이전
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd(E)");
        Calendar calendar = Calendar.getInstance();

        String today = sdf.format(calendar.getTime());
        calendar.add(Calendar.DAY_OF_MONTH, -1);
        String yesterday = sdf.format(calendar.getTime());
        calendar.add(Calendar.DAY_OF_MONTH, +2);
        String tomorrow = sdf.format(calendar.getTime());
        view(today, yesterday, tomorrow);

        // Java 8 이후
        LocalDate localDate = LocalDate.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd(E)");
        localDate = localDate.minusDays(1);
        yesterday = formatter.format(localDate);
        localDate = localDate.plusDays(2);
        tomorrow = formatter.format(localDate);
        view(today, yesterday, tomorrow);

        localDate = LocalDate.now();
        today = formatter.format(localDate);
        yesterday = formatter.format(localDate.minus(Period.ofDays(1)));
        tomorrow = formatter.format(localDate.plus(Period.ofDays(1)));
        view(today, yesterday, tomorrow);
    }

    private static void view(String today, String yesterday, String tomorrow) {
        System.out.println("어제 : " + yesterday);
        System.out.println("오늘 : " + today);
        System.out.println("내일 : " + tomorrow);
        System.out.println();
    }
}
```

결과

```
어제 : 2021-10-07(목)
오늘 : 2021-10-08(금)
내일 : 2021-10-09(토)

어제 : 2021-10-07(목)
오늘 : 2021-10-08(금)
내일 : 2021-10-09(토)

어제 : 2021-10-07(목)
오늘 : 2021-10-08(금)
내일 : 2021-10-09(토)
```

3. 1일과 마지막일 구하기

```
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.YearMonth;
import java.time.format.DateTimeFormatter;
import java.time.temporal.TemporalAdjusters;
import java.util.Calendar;

public class Ex03_FirstDayLastDay {
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E)";
        // Java 7 이전
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        Calendar calendar = Calendar.getInstance();
        int first = calendar.getActualMinimum(Calendar.DAY_OF_MONTH);
        int last = calendar.getActualMaximum(Calendar.DAY_OF_MONTH);
        System.out.println(sdf.format(calendar.getTime()) + " : " + first + "~"
+ last);
        calendar.set(2000, 2-1, 11); // 2000년 2월
        first = calendar.getActualMinimum(Calendar.DAY_OF_MONTH);
        last = calendar.getActualMaximum(Calendar.DAY_OF_MONTH);
        System.out.println(sdf.format(calendar.getTime()) + " : " + first + "~"
+ last);

        System.out.println();

        // Java 8 이후
        LocalDate localDate = LocalDate.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern(pattern);
        LocalDate firstDay = localDate.withDayOfMonth(1); // 날짜 변경
        first =  firstDay.getDayOfMonth(); // 일얻기
        // lastDay = localDate.withDayOfMonth(localDate.lengthOfMonth());
        // last = lastDay.getDayOfMonth();
        last = localDate.lengthOfMonth();
        System.out.println(formatter.format(localDate) + " : " + first + "~" +
last);

        localDate = LocalDate.of(2000, 2, 11);
        firstDay = localDate.withDayOfMonth(1);
        first =  firstDay.getDayOfMonth();
        // lastDay = localDate.withDayOfMonth(localDate.lengthOfMonth());
```

```
        // last = lastDay.getDayOfMonth();
        last = localDate.lengthOfMonth();
        System.out.println(formatter.format(localDate) + " : " + first + "~" +
last);
        System.out.println();

        //  Java에서 제공하는 YearMonth class 사용
        YearMonth month = YearMonth.from(localDate);
        LocalDate start = month.atDay(1);
        LocalDate end   = month.atEndOfMonth();
        first = start.getDayOfMonth();
        last = end.getDayOfMonth();
        System.out.println(start + " ~ " + end + " : " + first + "~" + last);

        // 만약, 이번달의 1일과 마지막 날짜를 구하고 싶다면 아래와 같이 할 수 있습니다.
        start = YearMonth.now().atDay(1);
        end   = YearMonth.now().atEndOfMonth();
        first = start.getDayOfMonth();
        last = end.getDayOfMonth();
        System.out.println(start + " ~ " + end + " : " + first + "~" + last);
        System.out.println();

        // TemporalAdjusters 사용
        start = localDate.with(TemporalAdjusters.firstDayOfMonth());
        end   = localDate.with(TemporalAdjusters.lastDayOfMonth());
        first = start.getDayOfMonth();
        last = end.getDayOfMonth();
        System.out.println(start + " ~ " + end + " : " + first + "~" + last);

        localDate = LocalDate.now();
        start = localDate.with(TemporalAdjusters.firstDayOfMonth());
        end   = localDate.with(TemporalAdjusters.lastDayOfMonth());
        first = start.getDayOfMonth();
        last = end.getDayOfMonth();
        System.out.println(start + " ~ " + end + " : " + first + "~" + last);
    }
}
```

결과

```
2021-10-08(금) : 1~31
2000-02-11(금) : 1~29

2021-10-08(금) : 1~31
2000-02-11(금) : 1~29

2000-02-01 ~ 2000-02-29 : 1~29
2021-10-01 ~ 2021-10-31 : 1~31

2000-02-01 ~ 2000-02-29 : 1~29
2021-10-01 ~ 2021-10-31 : 1~31
```

4. 년도의 시작일과 마지막일 구하기

```
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.TemporalAdjusters;
import java.util.Calendar;
import java.util.Date;

public class Ex04_FirstDayOfYearLastDayOfYear {
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E)";
        // Java 7 이전
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        Calendar calendar = Calendar.getInstance();
        calendar.set(calendar.get(Calendar.YEAR),1-1,1);
        Date firstDate = calendar.getTime();
        calendar.set(calendar.get(Calendar.YEAR),12-1,31);
        Date lastDate = calendar.getTime();
        System.out.println(sdf.format(firstDate) + " ~ " +
sdf.format(lastDate));
        System.out.println();

        // Java 8 이후
        LocalDate localDate = LocalDate.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern(pattern);
        LocalDate firstDay = localDate.withDayOfMonth(1).withMonth(1); // 날짜 변
경
        LocalDate lastDay = localDate.withDayOfMonth(31).withMonth(12);
        System.out.println(formatter.format(firstDay) + " ~ " +
formatter.format(lastDay));

        // TemporalAdjusters 사용
        firstDay = localDate.with(TemporalAdjusters.firstDayOfYear());
        lastDay   = localDate.with(TemporalAdjusters.lastDayOfYear());
        System.out.println(formatter.format(firstDay) + " ~ " +
formatter.format(lastDay));
    }
}
```

결과

```
2021-01-01(금) ~ 2021-12-31(금)

2021-01-01(금) ~ 2021-12-31(금)
2021-01-01(금) ~ 2021-12-31(금)
```

5. 윤년 알아내기

```
import java.time.LocalDate;
import java.time.Year;
import java.util.GregorianCalendar;
```

```
public class Ex05_LeapYear {
    public static void main(String[] args) {
        // Java 7 이전
        for(int year = 1998;year<=2004;year++) {
            boolean isLeap = year%400==0 || year%4==0 && year%100!=0;
            System.out.println(year + "년은 " + (isLeap ? "윤" : "평") + "년 입니
다.");
        }
        System.out.println();
        GregorianCalendar gregorianCalendar =  new GregorianCalendar();
        for(int year = 1998;year<=2004;year++) {
            System.out.println(year + "년은 " +
(gregorianCalendar.isLeapYear(year) ? "윤" : "평") + "년 입니다.");
        }
        System.out.println();

        // Java 8 이후
        for(int year = 1998;year<=2004;year++) {
            System.out.println(year + "년은 " + ( Year.of(year).isLeap() ? "윤" :
"평") + "년 입니다.");
        }
        System.out.println();

        for(int year = 1998;year<=2004;year++) {
            System.out.println(year + "년은 " + (
LocalDate.now().withYear(year).isLeapYear() ? "윤" : "평") + "년 입니다.");
        }
    }
}
```

결과

```
1998년은 평년 입니다.
1999년은 평년 입니다.
2000년은 윤년 입니다.
2001년은 평년 입니다.
2002년은 평년 입니다.
2003년은 평년 입니다.
2004년은 윤년 입니다.

1998년은 평년 입니다.
1999년은 평년 입니다.
2000년은 윤년 입니다.
2001년은 평년 입니다.
2002년은 평년 입니다.
2003년은 평년 입니다.
2004년은 윤년 입니다.

1998년은 평년 입니다.
1999년은 평년 입니다.
2000년은 윤년 입니다.
2001년은 평년 입니다.
2002년은 평년 입니다.
2003년은 평년 입니다.
2004년은 윤년 입니다.

1998년은 평년 입니다.
```

```
1999년은 평년 입니다.
2000년은 윤년 입니다.
2001년은 평년 입니다.
2002년은 평년 입니다.
2003년은 평년 입니다.
2004년은 윤년 입니다.
```

6. 현재 주 리스트로

```
import java.text.SimpleDateFormat;
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.TemporalAdjuster;
import java.time.temporal.TemporalAdjusters;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
// 현재 일주일 알아내기
import java.util.List;
public class Ex06_Weekly {
    @SuppressWarnings("deprecation")
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E)";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        DateTimeFormatter fmt = DateTimeFormatter.ofPattern(pattern);
        // Java 7 이전
        List<Date> list1 = getWeekly7(new Date());
        for(Date date : list1) System.out.println(sdf.format(date));
        System.out.println();
        list1 = getWeekly7(new Date(100, 0, 17));
        for(Date date : list1) System.out.println(sdf.format(date));
        System.out.println();


        // Java 8 이후
        List<LocalDate> list2 = getWeekly8(LocalDate.now());
        for(LocalDate localDate : list2)
            System.out.println(fmt.format(localDate));
        System.out.println();
        list2 = getWeekly8(LocalDate.of(2000, 1, 17));
        for(LocalDate localDate : list2)
            System.out.println(fmt.format(localDate));
    }

    public static List<Date> getWeekly7(Date date){
        List<Date> weekly = new ArrayList<Date>();
        Calendar cal = Calendar.getInstance();
        cal.setTime(date);
        // 이전 일요일로 이동
        while(cal.get(Calendar.DAY_OF_WEEK)>1) {
            cal.add(Calendar.DATE, -1);
        }
        do {
            cal.add(Calendar.DATE, 1);// 일 증가
```

```
            weekly.add(cal.getTime());// 추가
        }while(cal.get(Calendar.DAY_OF_WEEK)<=6);// 토요일까지
        cal.add(Calendar.DATE, 1);// 일요일로 이동
        weekly.add(cal.getTime());// 추가
        return weekly;
    }

    public static List<LocalDate> getWeekly8(LocalDate localDate){
        List<LocalDate> weekly = new ArrayList<LocalDate>();
        if(localDate.getDayOfWeek()!=DayOfWeek.MONDAY) { // 월요일이 아니면
            TemporalAdjuster temporalAdjuster =
TemporalAdjusters.previous(DayOfWeek.MONDAY);
            localDate = localDate.with(temporalAdjuster); // 이전 월요일로 이동
        }
        for(int i=0;i<7;i++) { // 7일
            weekly.add(localDate);
            localDate = localDate.plusDays(1); // 일 증가
        }
        return weekly;
    }
}
```

결과

```
2021-10-11(월)
2021-10-12(화)
2021-10-13(수)
2021-10-14(목)
2021-10-15(금)
2021-10-16(토)
2021-10-17(일)

2000-01-17(월)
2000-01-18(화)
2000-01-19(수)
2000-01-20(목)
2000-01-21(금)
2000-01-22(토)
2000-01-23(일)

2021-10-11(월)
2021-10-12(화)
2021-10-13(수)
2021-10-14(목)
2021-10-15(금)
2021-10-16(토)
2021-10-17(일)

2000-01-17(월)
2000-01-18(화)
2000-01-19(수)
2000-01-20(목)
2000-01-21(금)
2000-01-22(토)
2000-01-23(일)
```

7. 현재 월 리스트로 알아내기

```
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.TemporalAdjuster;
import java.time.temporal.TemporalAdjusters;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
// 현재 월을 리스트로
import java.util.List;
public class Ex07_Monthly {
    @SuppressWarnings("deprecation")
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E)";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        DateTimeFormatter fmt = DateTimeFormatter.ofPattern(pattern);
        // Java 7 이전
        List<Date> list1 = getMonthly7(new Date());
        for(Date date : list1) System.out.println(sdf.format(date));
        System.out.println();
        list1 = getMonthly7(new Date(100, 1, 17));
        for(Date date : list1) System.out.println(sdf.format(date));
        System.out.println();


        // Java 8 이후
        List<LocalDate> list2 = getMonthly8(LocalDate.now());
        for(LocalDate localDate : list2)
            System.out.println(fmt.format(localDate));
        System.out.println();
        list2 = getMonthly8(LocalDate.of(2000, 2, 17));
        for(LocalDate localDate : list2)
            System.out.println(fmt.format(localDate));
    }

    public static List<Date> getMonthly7(Date date){
        List<Date> monthly = new ArrayList<Date>();
        Calendar cal = Calendar.getInstance();
        cal.setTime(date);
        cal.set(Calendar.DAY_OF_MONTH, 1);
        int last = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        for(int i=1;i<=last;i++) {
            monthly.add(cal.getTime());// 추가
            cal.add(Calendar.DATE, 1);// 일 증가
        }
        return monthly;
    }

    public static List<LocalDate> getMonthly8(LocalDate localDate){
        List<LocalDate> monthly = new ArrayList<LocalDate>();
        TemporalAdjuster temporalAdjuster = TemporalAdjusters.firstDayOfMonth();
        localDate = localDate.with(temporalAdjuster);
        for(int i=0;i<localDate.lengthOfMonth();i++) {
            monthly.add(localDate.plusDays(i));
        }
```

```
            return monthly;
        }
    }
```

결과

```
2021-10-01(금)
2021-10-02(토)
.
.
.
2021-10-30(토)
2021-10-31(일)

2000-02-01(화)
2000-02-02(수)
.
.
.
2000-02-28(월)
2000-02-29(화)

2021-10-01(금)
2021-10-02(토)
.
.
.
2021-10-30(토)
2021-10-31(일)

2000-02-01(화)
2000-02-02(수)
.
.
.
2000-02-28(월)
2000-02-29(화)
```

8. 특정일자 사이의 차이 구하기

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.Date;
public class Ex08_DiffDay {
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        DateTimeFormatter fmt = DateTimeFormatter.ofPattern(pattern);
        String strStartDate = "2021-01-01";
        String strEndDate = "2021-12-31";
```

```
        // Java 7 이전
        try {
            Date startDate = sdf.parse(strStartDate);
            Date endDate = sdf.parse(strEndDate);
            long diffDate = getDiffDay7(startDate, endDate);
            System.out.println(sdf.format(startDate) + "~" + sdf.format(endDate)
+ " : " + diffDate);

            strStartDate = "0001-01-01";
            strEndDate = "2021-12-31";
            startDate = sdf.parse(strStartDate);
            endDate = sdf.parse(strEndDate);
            diffDate = getDiffDay7(startDate, endDate);
            System.out.println(sdf.format(startDate) + "~" + sdf.format(endDate)
+ " : " + diffDate);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        System.out.println();

        // Java 8 이후
        LocalDate startLocalDate = LocalDate.of(2021, 1, 1);
        LocalDate endLocalDate = LocalDate.of(2021, 12, 31);
        long diffDate = getDiffDay8(startLocalDate, endLocalDate);
        System.out.println(fmt.format(startLocalDate) + "~" +
fmt.format(endLocalDate) + " : " + diffDate);
        startLocalDate = LocalDate.of(1, 1, 1);
        endLocalDate = LocalDate.of(2021, 12, 31);
        diffDate = getDiffDay8(startLocalDate, endLocalDate);
        System.out.println(fmt.format(startLocalDate) + "~" +
fmt.format(endLocalDate) + " : " + diffDate);
        System.out.println();

        // 사용자 정의
        diffDate = getTotalDay(2021, 12, 31) - getTotalDay(2021, 1, 1);
        System.out.println(fmt.format(startLocalDate) + "~" +
fmt.format(endLocalDate) + " : " + diffDate);
        diffDate = getTotalDay(2021, 12, 31) - getTotalDay(1, 1, 1);
        System.out.println(fmt.format(startLocalDate) + "~" +
fmt.format(endLocalDate) + " : " + diffDate);

    }

    public static long getDiffDay7(Date startDate,Date endDate){
        // 1000밀리초는 1초로 계산되므로 getTime()으로 구한 값을 밀리초를 1000으로 나누
면 초를 얻습니다.
        // 이 초를 기본으로 하여 다른 시간 단위들도 계산할 수 있습니다.
        // 초 : / 1000
        // 분 : / (1000 * 60)
        // 시 : / (1000 * 60 * 60)
        long diffDay = 0;
        diffDay = (endDate.getTime() - startDate.getTime())/(60*60*24*1000);
        return diffDay;
    }

    public static long getDiffDay8(LocalDate startLocalDate, LocalDate
endLocalDate){
```

```
        // 두 날짜간 전체 일 수 차이를 구하려면 ChronoUnit 클래스의 between() 메소드를
사용하면 됩니다.
        // ChronoUnit 클래스는 enum 클래스이며 기간 단위별로 enum 타입이 정의되어 있습니
다.
        // ChronoUnit.YEARS : 년
        // ChronoUnit.MONTHS : 월
        // ChronoUnit.WEEKS : 주
        // ChronoUnit.DAYS : 일
        // ChronoUnit.HOURS : 시간
        // …
        long diffDay = 0;
        diffDay = ChronoUnit.DAYS.between(startLocalDate, endLocalDate);
        return diffDay;
    }
    // ----------------------------------------------------------------------------
----------------------
    // 윤년 판단
    private static boolean isLeapYear(int year) {
        return year%400==0 || year%4==0 && year%100!=0;
    }
    // 년월의 마지막 날짜
    private static int getLastDay(int year, int month) {
        int[] m = {31,28,31,30,31,30,31,31,30,31,30,31};
        if(month==2) m[1] = isLeapYear(year) ? 29 : 28;
        return m[month-1];
    }
    // 지정일까지의 총일수
    private static long getTotalDay(int year,int month, int date) {
        // 전년도 까지의 총일 수
        long days = (year-1) * 365 + (year-1)/4 - (year-1)/100 + (year-1)/400;
        // 전월까지의 총일수
        for(int i=1;i<month;i++) days += getLastDay(year, i);
        // + 일
        days += date;
        return days;
    }
}
```

결과

```
2021-01-01~2021-12-31 : 364
0001-01-01~2021-12-31 : 738156

2021-01-01~2021-12-31 : 364
0001-01-01~2021-12-31 : 738154

0001-01-01~2021-12-31 : 364
0001-01-01~2021-12-31 : 738154
```

9. 월의 특정 요일만 가져오기

```
import java.text.SimpleDateFormat;
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.TemporalAdjusters;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
// 월 특정 요일만 가져오기
public class Ex09_WeekDays {
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E)";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        DateTimeFormatter fmt = DateTimeFormatter.ofPattern(pattern);

        // Java 7 이전
        List<Date> weekDays7 = getWeekDay7(2021, 10, "일");
        for(Date date : weekDays7) System.out.println(sdf.format(date));
        System.out.println();
        weekDays7 = getWeekDay7(2021, 6, "수");
        for(Date date : weekDays7) System.out.println(sdf.format(date));
        System.out.println();
        // Java 8 이후
        List<LocalDate> weekDays8 = getWeekDay8(2021, 10, DayOfWeek.SUNDAY);
        for(LocalDate localDate : weekDays8)
System.out.println(fmt.format(localDate));
        System.out.println();
        weekDays8 = getWeekDay8(2021, 6, DayOfWeek.WEDNESDAY);
        for(LocalDate localDate : weekDays8)
System.out.println(fmt.format(localDate));
    }

    public static List<Date> getWeekDay7(int year, int month, String week){
        String weeks = "일월화수목금토";
        int intWeek = weeks.indexOf(week)+1;
        List<Date> weekDays = new ArrayList<Date>();
        Calendar cal = Calendar.getInstance();
        cal.set(year, month-1, 1);
        int last = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        for(int i=1;i<=last;i++) {
            cal.set(Calendar.DATE, i);// 일 변경
            if(cal.get(Calendar.DAY_OF_WEEK)==intWeek)
weekDays.add(cal.getTime());// 추가
        }
        return weekDays;
    }

    public static List<LocalDate> getWeekDay8(int year, int month, DayOfWeek
dayOfWeek){
        List<LocalDate> weekDays = new ArrayList<LocalDate>();
        LocalDate localDate = LocalDate.of(year, month, 1); // 1일로 만들기
        localDate = localDate.with(TemporalAdjusters.next(dayOfWeek)); // 다음요일
로 가기
        do {
```

```
            weekDays.add(localDate);
            localDate = localDate.with(TemporalAdjusters.next(dayOfWeek));
        }while(localDate.getMonth().getValue()==month); // 월이 같을때 까지
        return weekDays;
    }
}
```

결과

```
2021-10-03(일)
2021-10-10(일)
2021-10-17(일)
2021-10-24(일)
2021-10-31(일)

2021-06-02(수)
2021-06-09(수)
2021-06-16(수)
2021-06-23(수)
2021-06-30(수)

2021-10-03(일)
2021-10-10(일)
2021-10-17(일)
2021-10-24(일)
2021-10-31(일)

2021-06-02(수)
2021-06-09(수)
2021-06-16(수)
2021-06-23(수)
2021-06-30(수)
```

10. 몇일전, 몇일후, 몇달전, 몇달후, 몇년전, 몇년후

```
import java.text.SimpleDateFormat;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;
// 몇일전, 몇일후, 몇달전, 몇달후, 몇년전, 몇년후
public class Ex10_PrevNextDay {
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E) a hh:mm:ss";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        DateTimeFormatter fmt = DateTimeFormatter.ofPattern(pattern);

        // Java 7 이전
        Calendar calendar = Calendar.getInstance();
        System.out.println("오늘 : " + sdf.format(calendar.getTime()));
        // 몇년전, 몇년후
        calendar.add(Calendar.YEAR, 10);
        System.out.println("10년 후 : " + sdf.format(calendar.getTime()));
        calendar.add(Calendar.YEAR, -20);
        System.out.println("10년 전 : " + sdf.format(calendar.getTime()));
```

```
        // 몇달전, 몇달후
        calendar.add(Calendar.MONTH, 6);
        System.out.println("6개월 후 : " + sdf.format(calendar.getTime()));
        calendar.add(Calendar.MONTH, -12);
        System.out.println("6개월 전 : " + sdf.format(calendar.getTime()));

        // 몇일전, 몇일후
        calendar.add(Calendar.DAY_OF_MONTH, 3);
        System.out.println("3일 후 : " + sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, -6);
        System.out.println("3일 전 : " + sdf.format(calendar.getTime()));
        // 몇시간전, 몇시간후
        calendar.add(Calendar.HOUR_OF_DAY, 5);
        System.out.println("5시간 후 : " + sdf.format(calendar.getTime()));
        calendar.add(Calendar.HOUR_OF_DAY, -10);
        System.out.println("5시간 전 : " + sdf.format(calendar.getTime()));
        System.out.println();


        // Java 8 이후
        LocalDateTime localDateTime = LocalDateTime.now();
        System.out.println("오늘 : " + fmt.format(localDateTime));
        // 몇년전, 몇년후
        localDateTime = localDateTime.plusYears(10);
        System.out.println("10년 후 : " + fmt.format(localDateTime));
        localDateTime = localDateTime.minusYears(20);
        System.out.println("10년 전 : " + fmt.format(localDateTime));

        // 몇달전, 몇달후
        localDateTime = localDateTime.plusMonths(6);
        System.out.println("6개월 후 : " + fmt.format(localDateTime));
        localDateTime = localDateTime.minusMonths(12);
        System.out.println("6개월 전 : " + fmt.format(localDateTime));

        // 몇일전, 몇일후
        localDateTime = localDateTime.plusDays(3);
        System.out.println("3일 후 : " + fmt.format(localDateTime));
        localDateTime = localDateTime.minusDays(6);
        System.out.println("3일 전 : " + fmt.format(localDateTime));

        // 몇시간전, 몇시간후
        localDateTime = localDateTime.plusHours(5);
        System.out.println("5시간 후 : " + fmt.format(localDateTime));
        localDateTime = localDateTime.minusHours(10);
        System.out.println("5시간 전 : " + fmt.format(localDateTime));
    }
}
```

결과

```
오늘 : 2021-10-12(화) 오후 03:02:12
10년 후 : 2031-10-12(일) 오후 03:02:12
10년 전 : 2011-10-12(수) 오후 03:02:12
6개월 후 : 2012-04-12(목) 오후 03:02:12
6개월 전 : 2011-04-12(화) 오후 03:02:12
3일 후 : 2011-04-15(금) 오후 03:02:12
```

```
3일 전 : 2011-04-09(토) 오후 03:02:12
5시간 후 : 2011-04-09(토) 오후 08:02:12
5시간 전 : 2011-04-09(토) 오전 10:02:12

오늘 : 2021-10-12(화) 오후 03:02:12
10년 후 : 2031-10-12(일) 오후 03:02:12
10년 전 : 2011-10-12(수) 오후 03:02:12
6개월 후 : 2012-04-12(목) 오후 03:02:12
6개월 전 : 2011-04-12(화) 오후 03:02:12
3일 후 : 2011-04-15(금) 오후 03:02:12
3일 전 : 2011-04-09(토) 오후 03:02:12
5시간 후 : 2011-04-09(토) 오후 08:02:12
5시간 전 : 2011-04-09(토) 오전 10:02:12
```

11. 주말과 휴일을 제외한 날짜 구하기

```java
import java.text.SimpleDateFormat;
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoField;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
// 주말과 휴일을 제외한 날짜 구하기
public class Ex11_WorkDays {
    @SuppressWarnings("deprecation")
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E)";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        DateTimeFormatter fmt = DateTimeFormatter.ofPattern(pattern);

        // Java 7 이전
        List<Date> workDays7 = getWorkDays7(2021, 10);
        for(Date date : workDays7) System.out.println(sdf.format(date));
        System.out.println();

        List<Date> holidays = new ArrayList<Date>();
        holidays.add(new Date(121,9,1));
        holidays.add(new Date(121,9,3));
        holidays.add(new Date(121,9,4));
        holidays.add(new Date(121,9,11));
        workDays7 = getWorkDays7(2021, 10, holidays);
        for(Date date : workDays7) System.out.println(sdf.format(date));
        System.out.println();

        // Java 8 이후
        List<LocalDate> workDays8 = getWorkDays8(2021, 10);
        for(LocalDate localDate : workDays8)
            System.out.println(fmt.format(localDate));
        System.out.println();

        List<LocalDate> holidays2 = new ArrayList<LocalDate>();
        holidays2.add(LocalDate.of(2021, 10, 1));
```

```
        holidays2.add(LocalDate.of(2021, 10, 3));
        holidays2.add(LocalDate.of(2021, 10, 4));
        holidays2.add(LocalDate.of(2021, 10, 11));
        workDays8 = getWorkDays8(2021, 10, holidays2);
        for(LocalDate localDate : workDays8)
            System.out.println(fmt.format(localDate));
        System.out.println();
    }
    // parameter : 년, 월
    public static List<Date> getWorkDays7(int year, int month){
        List<Date> workDays = new ArrayList<Date>();
        Calendar cal = Calendar.getInstance();
        cal.set(year, month-1, 1);
        int last = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        for(int i=1;i<=last;i++) {
            cal.set(Calendar.DATE, i);// 일 변경
            if(cal.get(Calendar.DAY_OF_WEEK)!=Calendar.SUNDAY &&
cal.get(Calendar.DAY_OF_WEEK)!=Calendar.SATURDAY)
                workDays.add(cal.getTime());// 추가
        }
        return workDays;
    }
    // parameter : 년, 월, 휴일리스트
    public static List<Date> getWorkDays7(int year, int month, List<Date>
holidays){
        List<Date> workDays = new ArrayList<Date>();
        Calendar cal = Calendar.getInstance();
        cal.set(year, month-1, 1);
        int last = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        for(int i=1;i<=last;i++) {
            cal.set(Calendar.DATE, i);// 일 변경
            if(!isContains(holidays, cal.getTime()) &&
                    cal.get(Calendar.DAY_OF_WEEK)!=Calendar.SUNDAY &&
cal.get(Calendar.DAY_OF_WEEK)!=Calendar.SATURDAY) {
                workDays.add(cal.getTime());// 추가
            }
        }
        return workDays;
    }
    // 리스트에 포함 여부 판단
    private static boolean isContains(List<Date> holidays, Date date) {
        String pattern = "yyyyMMdd";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        for(Date d : holidays) {
            if(sdf.format(date).equals(sdf.format(d))) return true;
        }
        return false;
    }
    // parameter : 년, 월
    public static List<LocalDate> getWorkDays8(int year, int month){
        List<LocalDate> workDays = new ArrayList<LocalDate>();
        LocalDate localDate = LocalDate.of(year, month, 1);
        int last = localDate.lengthOfMonth();
        localDate = localDate.minusDays(1);
        for(int i=1;i<=last;i++) {
            localDate = localDate.plusDays(1);
            if(!isWeekend(localDate)) workDays.add(localDate);
        }
```

```
        return workDays;
    }
    // parameter : 년, 월, 휴일리스트
    public static List<LocalDate> getWorkDays8(int year, int month,
List<LocalDate> holidays){
        List<LocalDate> workDays = new ArrayList<LocalDate>();
        LocalDate localDate = LocalDate.of(year, month, 1);
        int last = localDate.lengthOfMonth();
        localDate = localDate.minusDays(1);
        for(int i=1;i<=last;i++) {
            localDate = localDate.plusDays(1);
            if(!holidays.contains(localDate) && !isWeekend(localDate))
workDays.add(localDate);
        }
        return workDays;
    }
    private static boolean isWeekend(LocalDate localDate) {
        DayOfWeek day = DayOfWeek.of(localDate.get(ChronoField.DAY_OF_WEEK));
        return day == DayOfWeek.SUNDAY || day == DayOfWeek.SATURDAY;
    }

}
```

결과

```
2021-10-01(금)
2021-10-04(월)
2021-10-05(화)
.
.
.
2021-10-28(목)
2021-10-29(금)

2021-10-05(화)
2021-10-06(수)
2021-10-07(목)
2021-10-08(금)
2021-10-12(화)
.
.
.
2021-10-28(목)
2021-10-29(금)

2021-10-01(금)
2021-10-04(월)
2021-10-05(화)
.
.
.
2021-10-28(목)
2021-10-29(금)

2021-10-05(화)
2021-10-06(수)
2021-10-07(목)
```

```
2021-10-08(금)
2021-10-12(화)
.
.
.
2021-10-29(금)
```

12. Date와 Calendar객체의 비교

```
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class Ex01 {
    @SuppressWarnings("deprecation")
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E)";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        Date date = new Date(121,9,1);
        Date date1 = new Date(121,9,1);

        System.out.println(sdf.format(date));
        System.out.println(sdf.format(date1));

        // 같은 Date객체 비교
        System.out.println(date.equals(date1));
        System.out.println(date.compareTo(date1)==0);
        System.out.println(sdf.format(date).equals(sdf.format(date1)));
        System.out.println();

        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.DATE, 1);
        System.out.println(sdf.format(calendar.getTime()));

        // Date와 Calendar를 비교
        System.out.println(date1.equals(calendar.getTime()));
        System.out.println(date1.compareTo(calendar.getTime())==0);

System.out.println(sdf.format(date1).equals(sdf.format(calendar.getTime())));
        System.out.println();

        // Date와 Calendar를 Date로 바꿔서비교
        Date date2 = calendar.getTime();
        System.out.println(sdf.format(date2));
        System.out.println(date2.equals(date1));
        System.out.println(date2.compareTo(date1)==0);
        System.out.println(sdf.format(date1).equals(sdf.format(date2)));
        System.out.println();
    }
}
```

결과

```
2021-10-01(금)
```

```
2021-10-01(금)
true
true
true

2021-10-01(금)
false
false
true

2021-10-01(금)
false
false
true
```

13. Date타입의 List에 특정 날짜 포함 여부

```
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

public class Ex13_Contains {
    @SuppressWarnings("deprecation")
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E)";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        List<Date> holidays = new ArrayList<Date>();
        holidays.add(new Date(121,9,1));
        holidays.add(new Date(121,9,3));
        holidays.add(new Date(121,9,4));
        holidays.add(new Date(121,9,11));

        for(Date date : holidays) System.out.println(sdf.format(date));
        System.out.println();

        Calendar cal = Calendar.getInstance();
        cal.set(2021, 9, 1);
        int last = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        for(int i=1;i<=last;i++) {
            cal.set(Calendar.DATE, i);// 일 변경
            Date date = cal.getTime();
            System.out.println(sdf.format(date) + " : " +
holidays.contains(date)+ ", " + isContains(holidays, date));
        }
    }
    private static boolean isContains(List<Date> list, Date date) {
        String pattern = "yyyyMMdd";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        for(Date d : list) {
            if(sdf.format(date).equals(sdf.format(d))) return true;
        }
        return false;
    }
```

```
}
```

결과

```
2021-10-01(금)
2021-10-03(일)
2021-10-04(월)
2021-10-11(월)

2021-10-01(금) : false, true
2021-10-02(토) : false, false
2021-10-03(일) : false, true
2021-10-04(월) : false, true
2021-10-05(화) : false, false
2021-10-06(수) : false, false
2021-10-07(목) : false, false
2021-10-08(금) : false, false
2021-10-09(토) : false, false
2021-10-10(일) : false, false
2021-10-11(월) : false, true
2021-10-12(화) : false, false
2021-10-13(수) : false, false
2021-10-14(목) : false, false
2021-10-15(금) : false, false
2021-10-16(토) : false, false
2021-10-17(일) : false, false
2021-10-18(월) : false, false
2021-10-19(화) : false, false
2021-10-20(수) : false, false
2021-10-21(목) : false, false
2021-10-22(금) : false, false
2021-10-23(토) : false, false
2021-10-24(일) : false, false
2021-10-25(월) : false, false
2021-10-26(화) : false, false
2021-10-27(수) : false, false
2021-10-28(목) : false, false
2021-10-29(금) : false, false
2021-10-30(토) : false, false
2021-10-31(일) : false, false
```

14. 기간에 특정 날짜 포함 여부

```
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;
import java.util.Date;

// 특정 날짜가 특정 기간 사이에 있는지 체크하기
public class Ex14_WithInRange {
    public static void main(String[] args) {
        String pattern = "yyyy-MM-dd(E)";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
```

```
        DateTimeFormatter fmt = DateTimeFormatter.ofPattern(pattern);

        // Java 7 이전
        Calendar calendar =  Calendar.getInstance();
        calendar.set(2021, 9, 8);
        Date date = calendar.getTime();
        calendar.set(2021, 9, 1);
        Date startDate = calendar.getTime();
        calendar.set(2021, 9, 31);
        Date endDate = calendar.getTime();
        System.out.println(sdf.format(date));
        System.out.println(sdf.format(startDate));
        System.out.println(sdf.format(endDate));
        System.out.println(isWithinRange(date, startDate, endDate));
        calendar.set(2021, 10, 1);
        Date date2 = calendar.getTime();
        System.out.println(sdf.format(date2));
        System.out.println(isWithinRange(date2, startDate, endDate));

        // Java 8 이후
        LocalDate localDate = LocalDate.of(2021, 10, 8);
        LocalDate startLocalDate = LocalDate.of(2021, 10, 1);
        LocalDate endLocalDate = LocalDate.of(2021, 10, 31);
        System.out.println(fmt.format(localDate));
        System.out.println(fmt.format(startLocalDate));
        System.out.println(fmt.format(endLocalDate));
        System.out.println(isWithinRange(localDate, startLocalDate,
endLocalDate));
        LocalDate localDate2 = LocalDate.of(2021, 11, 8);
        System.out.println(fmt.format(localDate2));
        System.out.println(isWithinRange(localDate2, startLocalDate,
endLocalDate));

    }
    public static boolean isWithinRange(Date date, Date startDate, Date endDate)
{
        boolean result = false;
        result = !startDate.after(date) && !endDate.before(date);
        return result;
    }
    public static boolean isWithinRange(LocalDate date, LocalDate startDate,
LocalDate endDate) {
        boolean result = false;
        result = !startDate.isAfter(date) && !endDate.isBefore(date);
        return result;
    }
}
```

결과

```
2021-10-08(금)
2021-10-01(금)
2021-10-31(일)
true
2021-11-01(월)
false
2021-10-08(금)
2021-10-01(금)
2021-10-31(일)
true
2021-11-08(월)
false
```