제 34 강

java.time 패키지

교재:p202~217

목차

1. java.time 패키지

java.time 패키지?

: Java에서 날짜와 시간을 다루기 위한 패키지

[1] java.time 패키지의 하위 패키지

- JDK 1.8부터 추가된 패키지로서 기존의 Date 와 Calendar를 보완

패키지	설명
java.time	날짜와 시간을 나타내는 LocalDate,LocalTime 등을 포함한 패키지
java.time.format	날짜와 시간을 파싱하고 포맷팅하는 API 포함
java.time.chrono	여러가지 달력 시스템을 사용할 수 있는 API vhgka
java.time.temporal	날짜와 시간을 연산하기 위한 API 포함
java.time.zone	타임존을 지원하는 API 포함

1. java.time 패키지

[2] time 패키지의 시간과 날짜 클래스 종류

- **LocalDate 클래스** : 날짜 정보를 저장

- 날짜 정보를 저장하는 방법?

➢ 현재 날짜 지정: now( ) 메서드

➢ 특정 날짜 지정: of( ) 메서드

```
LocalDate ld = LocalDate.now( )
LocalDate mld=LocalDate.of(int year,int month, int dayOfMonth);
```

1. java.time 패키지

[2] time 패키지의 시간과 날짜 클래스 종류

- **LocalTime 클래스** : 시간 정보를 저장

- 시간 정보를 저장하는 방법?

➢ 현재 시간 지정: now( ) 메서드

➢ 특정 시간 지정: of( ) 메서드

```
LocalTime lt = LocalTime.now( )
LocalTime mlt=LocalTime.of(int hour,int minute, int second, int nanoOfSecond);
```

<of메서드의 오버로딩>

```
of(int hour, int minute);
of(int hour, int minute, int second);
of(int hour, int minute, int second, int nanoOfSecond);
```

1. java.time 패키지

[2] time 패키지의 시간과 날짜 클래스 종류

- **LocalDateTime 클래스** : LocalDate클래스와 LocalTime 클래스를 결합한 클래스

- 날짜 정보와 시간 정보 모두 저장

➢ 현재 날짜와 시간 지정: now( ) 메서드

➢ 특정 날짜와 시간 지정: of( ) 메서드

```
LocalDateTime ldt=LocalDateTIme.now( );
LocalDateTime mldt=LocalDateTime.of(int year,int month, int dayOfMonth,int hour, int minute, int second, int nanoOfSecond);
```

1. java.time 패키지

[2] time 패키지의 시간과 날짜 클래스 종류

- **ZonedDateTime 클래스** : ISO-8601 달력 시스템에서 정의하는 Time Zone에 따라 날짜와 시간 저장

➢ 형식: 2016-01-08T12:56:09.017+09:00(Asia/Seoul]

- 협정 세계시(UTC)와 차이 나는 시간(존 오프셋)이 따로 저장
- ZoneDateTime 클래스는 now( ) 메서드 뒤에 ZoneId를 매개값으로 넘겨야함

```
ZoneDateTime zdt = ZoneDateTime.now(ZoneId.of("UTC"));
```

1. java.time 패키지

[2] time 패키지의 시간과 날짜 클래스 종류

- **Instant 클래스** : 특정 시점의 타임 스탬프 객체

- 1970년 1월 1일부터 현재까지의 시간을 세는 객체
- Machine time에 유리

```
Instant i = Instant.now( );
```

1. java.time 패키지

[3] 클래스들의 정보 값 읽어오기

- **LocalDate/LacalTime 클래스**

- Calender 클래스와의 차이?

Calendar 클래스는 1월이 0으로 시작하나,

LocalDate는 1월은 1로. 시작

클래스	리턴 타입	메서드	설명
LocalDate	int	getYear( )	년도
	Month	getMonth( )	Month의 열거값
	int	getMonthValue( )	월
	int	getDayOfYear( )	1년의 몇 번째 일
	int	getDayOfMonth( )	월의 몇 번째 일
	DayOfWeek	getDayOfWeek( )	요일
	boolean	isLeapYear( )	윤년 여부
LocalTime	int	getHour( )	시간
	int	getMinute( )	분
	int	getSecond( )	초
	int	getNano( )	나노초

1. java.time 패키지

[4] 클래스들의 정보 더하기/빼기

각 클래스는 필드의 값을 변경하기 위한 메서드가 존재

메서드	설명	메서드	설명
minusYears(long)	년도 빼기	plusWeeks(long)	주 더하기
minusMonths(long)	월 빼기	minusHours(long)	시간 빼기
minusDays(long)	일 빼기	minusMinutes(long)	분 빼기
minusWeeks(long)	주 빼기	minusSeconds(long)	초 빼기
plusYears(long)	년도 더하기	minusNanos(long)	나노초 빼기
plusMonths(long)	월 더하기	plusHours(long)	시간 더하기
plusDays(long)	일 더하기	plusMinutes(long)	분 더하기

1. java.time 패키지

[4] 클래스들의 정보 더하기/빼기

<실습> Exam-78.java

날짜를 더하고 빼는 메서드 확인하기

```
public class TimePlus {
    public static void main(String[] args) {
        LocalDate ld= LocalDate.now();
        System.out.println(ld);

        LocalDate ld2=ld.minusYears(2).plusMonths(3).minusDays(4);
        System.out.println(ld2);

        //자동 변환
        LocalDate ld3 = ld2.minusDays(3);
        System.out.println(ld3);

        //주 더하기
        LocalDate ld4 = ld3.plusWeeks(3);
        System.out.println(ld4);
    }
}
```

<실행 결과>

```
2020-08-24
2018-11-20
2018-11-17
2018-12-08
```

1. java.time 패키지

[4] 클래스들의 정보 더하기/빼기

<실습> Exam-79.java

시간을 더하고 빼는 메서드 확인하기

```
public class TimePlus2 {
    public static void main(String[] args) {
        LocalDateTime ld = LocalDateTime.now();
        System.out.println("현재 시간: "+ld);
        LocalDateTime ld2= ld.minusHours(5).plusMinutes(30).minusSeconds(4);
        System.out.println("변경 시간: "+ld2);

        LocalDateTime ld3= ld2.minusHours(24);
        System.out.println("자동 변환 시간: "+ld3);
    }
}
```

<실행 결과>

```
현재 시간: 2020-08-24T20:10:19.816491
변경 시간: 2020-08-24T15:40:15.816491
자동 변환 시간: 2020-08-23T15:40:15.816491
```

1. java.time 패키지

[5] 클래스들의 정보 값 변경하기

각 클래스는 필드의 값들을 특정값으로 변경하기 위한
메서드가 존재

메서드	설명	메서드	설명
withYear(int)	년 변경	withHour(int)	시간 변경
withMonth(int)	월 변경	withMinute(int)	분 변경
withDayOfMonth(int)	월의 일 변경	withSecond(int)	초 변경
withDayOfYear(int)	년의 일 변경	withNano(int)	나노초 변경

1. java.time 패키지

[5] 클래스들의 정보 값 변경하기

<실습> Exam-80.java

간단하게 날짜를 변경하는 코드

```
public class WithPrac {
    public static void main(String[] args) {
            LocalDate ld=LocalDate.now();
            System.out.println(ld);
            LocalDate new_ld=ld.withYear(1999).withMonth(8).withDayOfYear(23);
            System.out.println(new_ld);
    }
}
```

withDayOfYear(int): 년을 기준으로 지난 일자
withDayOfMonth(int): 월을 기준으로 지난 일자

<실행 결과>

```
2020-08-24
1999-01-23
```

1. java.time 패키지

[5] 클래스들의 정보 값 변경하기

<실습> Exam-81.java

간단하게 시간을 변경하는 코드

```
public class WithPrac2 {
    public static void main(String[] args) {
        LocalTime lt=LocalTime.now();
        System.out.println(lt);

        LocalTime now_time=lt.withHour(3).withMinute(25).withSecond(24).withNano(33333);
        System.out.println(now_time);
    }
}
```

<실행 결과>

```
20:23:32.976311
03:25:24.000033333
```

1. java.time 패키지

[5] 클래스들의 정보 값 변경하기

<실습> Exam-82.java

1900년부터 2100년까지 윤년이 언제인지, 몇 번 있는지 알아보는 코드

```
public class Leap {
    public static void main(String[] args) {
        LocalDate ld= LocalDate.now();
        LocalDate new_ld;
        int count=0;


        for(int i=1900;i<=2100;i++) {
            new_ld=ld.withYear(i);
            if(new_ld.isLeapYear()) {
                System.out.println(new_ld.getYear()+"은 윤년입니다.");
                count++;
            }
        }
        System.out.println("1900년 부터 2100년까지 윤년은 총 "+count+"번 있습니다.");
    }
}
```

<실행 결과>

...

```
2072은 윤년입니다.
2076은 윤년입니다.
2080은 윤년입니다.
2084은 윤년입니다.
2088은 윤년입니다.
2092은 윤년입니다.
2096은 윤년입니다.
1900년 부터 2100년까지 윤년은 총 49번 있습니다.
```

1. java.time 패키지

with( )메서드

: TemporalAdjuster 타입을 인자로 받으면 특정한 날짜를 리턴

메서드	설명
firstDayOfYear()	년도의 첫 번째 일
lastDayOfYear( )	년도의 마지막 일
firstDayOfMonth( )	달의 첫 번째 일
lastDayOfMonth( )	달의 마지막 일
firstInMonth(DayOfWeek dow)	달의 첫 번째 요일
lastInMonth(DayOfWeek dow)	달의 마지막 요일
next(DayOfWeek dow)	돌아오는 요일
nextOrSame(DayOfWeek dow)	오늘을 포함한 돌아오는 요일
previous(DayOfWeek dow)	지난 요일
previousOrSame(DayOfWeek dow)	오늘을 포함한 지난 요일

1. java.time 패키지

<실습> Exam-83.java

with( )메서드 실습

```
public class TemproralPrac {
    public static void main(String[] args) {
        LocalDateTime ldt = LocalDateTime.now();
        System.out.println(ldt);
        LocalDateTime new_ldt;

        new_ldt =ldt.with(TemporalAdjusters.firstDayOfYear());
        System.out.println("올해의 첫 번째 날: "+new_ldt);
        new_ldt=ldt.with(TemporalAdjusters.lastDayOfYear());
        System.out.println("올해의 마지막 날: "+new_ldt);

        new_ldt =ldt.with(TemporalAdjusters.firstDayOfMonth());
        System.out.println("이번 달의 첫 번째 날: "+new_ldt);
        new_ldt=ldt.with(TemporalAdjusters.lastDayOfMonth());
        System.out.println("이번 달의 마지막 날: "+new_ldt);

        new_ldt =ldt.with(TemporalAdjusters.firstInMonth(DayOfWeek.MONDAY));
        System.out.println("이번 달의 첫 번째 월요일: "+new_ldt);
        new_ldt =ldt.with(TemporalAdjusters.lastInMonth(DayOfWeek.SUNDAY));
        System.out.println("이번 달의 마지막 일요일: "+new_ldt);

        new_ldt =ldt.with(TemporalAdjusters.next(DayOfWeek.FRIDAY));
        System.out.println("돌아오는 금요일: "+new_ldt);
        new_ldt =ldt.with(TemporalAdjusters.nextOrSame(DayOfWeek.FRIDAY));
        System.out.println("오늘을 포함한 다음 금요일: "+new_ldt);

        new_ldt =ldt.with(TemporalAdjusters.previous(DayOfWeek.MONDAY));
        System.out.println("지난 월요일: "+new_ldt);
        new_ldt =ldt.with(TemporalAdjusters.previous(DayOfWeek.MONDAY));
        System.out.println("오늘을 포함한 지난 월요일: "+new_ldt);
    }
}
```

<실행 결과>

```
2020-08-24T20:43:24.864612
올해의 첫 번째 날: 2020-01-01T20:43:24.864612
올해의 마지막 날: 2020-12-31T20:43:24.864612
이번 달의 첫 번째 날: 2020-08-01T20:43:24.864612
이번 달의 마지막 날: 2020-08-31T20:43:24.864612
이번 달의 첫 번째 월요일: 2020-08-03T20:43:24.864612
이번 달의 마지막 일요일: 2020-08-30T20:43:24.864612
돌아오는 금요일: 2020-08-28T20:43:24.864612
오늘을 포함한 다음 금요일: 2020-08-28T20:43:24.864612
지난 월요일: 2020-08-17T20:43:24.864612
오늘을 포함한 지난 월요일: 2020-08-17T20:43:24.864612
```

1. java.time 패키지

[6] 클래스들의 정보 값 비교하기

time 패키지에서는 각 필드 값을 비교하는 메서드 사용

메서드	설명
isAfter( )	이전의 날짜인지 비교하여 boolean값 반환
isBefore( )	지나간 날짜인지 비교하여 boolean 값 반환
isEqual( )	동일 날짜인지 비교하여 boolean 값 반환
until( )	날짜나 시간의 차이를 반환
between( )	전체 날짜나 시간의 차이를 반환

1. java.time 패키지

<실습> Exam-84java

간단하게 날짜 변경 코드 실습

```
public class After1 {
    public static void main(String[] args) {
        LocalDateTime ldt1= LocalDateTime.of(2010,1,1,12,23,23,444);
        System.out.println(ldt1);

        LocalDateTime ldt2 = LocalDateTime.of(2010,12,25,1,12,2,232);
        System.out.println(ldt2);

        //ldt1이 ldt2보다 이후의 날짜인가?
        System.out.println(ldt1.isAfter(ldt2));

        //ldt1이 ldt 이전의 날짜인가?
        System.out.println(ldt1.isBefore(ldt2));

        //ldt1과 ldt2 는 같은 날짜인가?
        System.out.println(ldt1.equals(ldt2));
    }
}
```

<실행 결과>

```
2010-01-01T12:23:23.000000444
2010-12-25T01:12:02.000000232
false
true
false
```

1. java.time 패키지

<실습> Exam-85java

isAfter( ) 메서드를 사용해 유통기한 점검하는 실습

```
public class After2 {
    public static void main(String[] args) {
        LocalDateTime ldt= LocalDateTime.now();
        System.out.println(ldt);

        LocalDateTime end_time=LocalDateTime.of(2017,6,24,12,00);
        System.out.println(end_time);
        if(ldt.isAfter(end_time)) {
            System.out.println("유통기한이 지났습니다!");
        }
        else {
            System.out.println("유통기한이 아직 지나지 않ㄹ았습니다!");
        }
    }
}
```

<실행 결과>

```
2020-08-24T20:53:14.010271
2017-06-24T12:00
유통기한이 지났습니다!
```

2. Timer/TimerTask

Timer/TimerTask 클래스?

: 개발자가 원하는 특정한 시간에 코드를 실행하거나 특정 시간 간격으로 반복되는 작업 처리

<사용방법>

1. TimerTask를 상속받는 새로운 클래스 선언
2. TimerTask 클래스의 run 추상 메서드에서 하고 싶은 작업 오버라이딩
3. Timer 객체 생성 후 원하는 시간을 매개변수 값으로 전달

2. Timer/TimerTask

Timer/TimerTask 클래스?

<사용방법>

1. TimerTask를 상속받는 새로운 클래스 선언

   class Work1 extends TimerTask{ … }

2. TimerTask 클래스의 run 추상 메서드에서 하고 싶은 작업 오버라이딩

   public void run(){ … }

2. Timer/TimerTask

Timer/TimerTask 클래스?

<사용방법>

3. Timer 객체 생성 후 원하는 시간을 매개변수 값으로 전달

```
Timer t = new Timer(true);  프로그램 종료 시, 객체 자동 소멸
TimerTask w1= new Work1( );
t.schedule(w1,5000);  1000 == 1초
```

2. Timer/TimerTask

<실습> Exam-86java

타이머 실습

```
public class Timer1 {
    public static void main(String[] args) throws InterruptedException{
        Timer t= new Timer(true);
        TimerTask w1 = new Work1();
        TimerTask w2 = new Work2();
        t.schedule(w1, 3000);
        t.schedule(w2, 1000);
        Thread.sleep(4000);
        System.out.println("모든 작업 종료");
    }
}

class Work1 extends TimerTask{
    @Override
    public void run() {
        System.out.println("work1 실행!");
    }
}
class Work2 extends TimerTask{
    @Override
    public void run() {
        System.out.println("work2 실행!");
    }
}
```

<실행 결과>

```
work2 실행!
work1 실행!
모든 작업 종료
```

w2: 1초 후 실행

w1: 3초 후 실행

Thread.sleep(4000): 4초까지 기다리기

→ 4초 후 프로그램 종료