JAVA : Date and Time

Prof. Youngchul Jung

YEUNGJIN UNIVERSITY

날짜와 시간 (Date and Time) 관련 클래스

✓ *java.util package*

Date → Calendar

since JDK 1.0 | since JDK 1.1

Module java.base
Package java.util

Class Date

java.lang.Object
    java.util.Date

All Implemented Interfaces:

Serializable, Cloneable, Comparable<Date>

Direct Known Subclasses:

Date, Time, Timestamp

```
public class Date
extends Object
implements Serializable, Cloneable, Comparable<Date>
```

Module java.base
Package java.util

Class Calendar

java.lang.Object
    java.util.Calendar

All Implemented Interfaces:

Serializable, Cloneable, Comparable<Calendar>

Direct Known Subclasses:

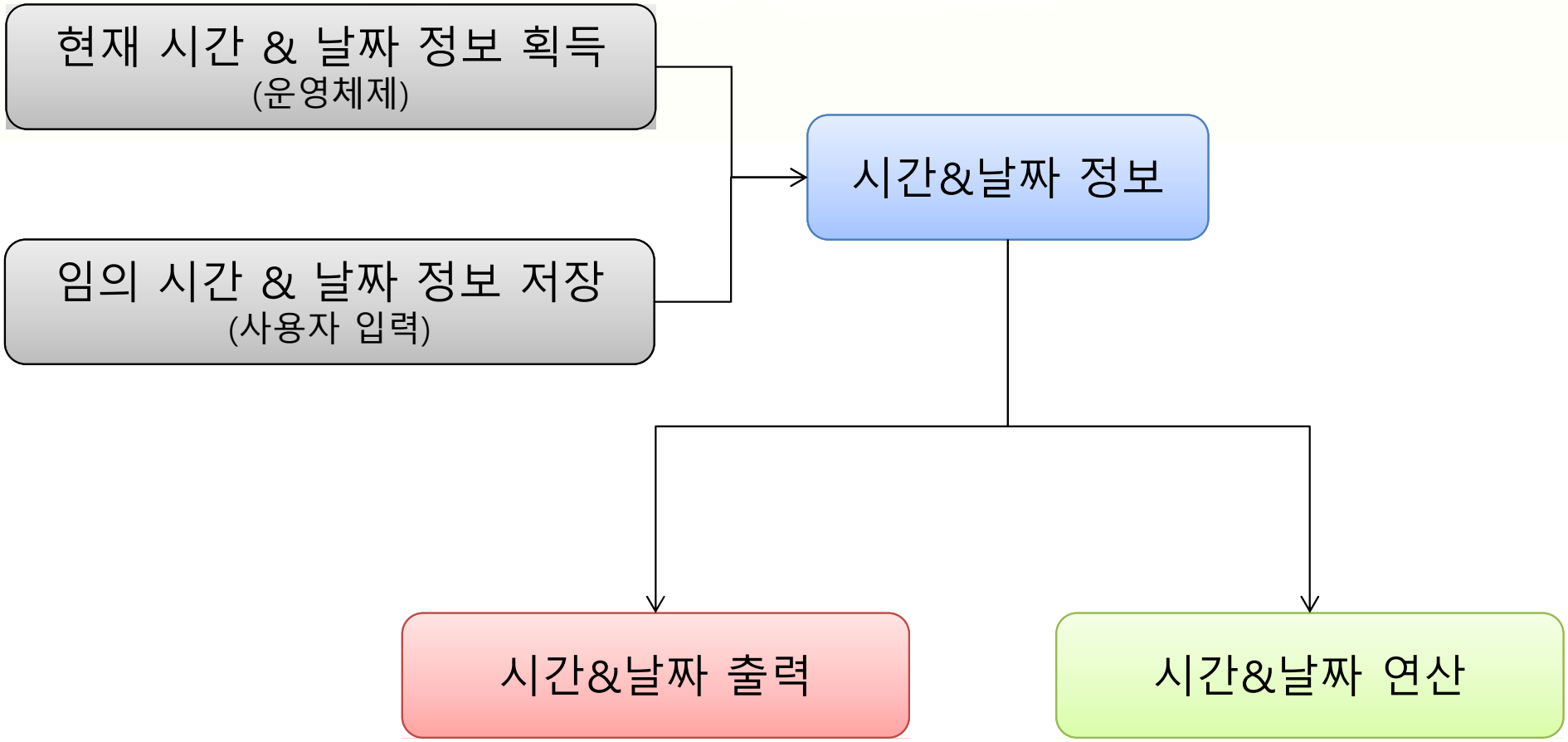
GregorianCalendar

```
public abstract class Calendar
extends Object
implements Serializable, Cloneable, Comparable<Calendar>
```

✓ *java.time package* since JDK 1.8

- **Instant** — Timestamp 값 저장
- **LocalDate** — 날짜 값 저장 및 가공
- **LocalTime** — 시간 값 저장 및 가공
- **LocalDateTime** — 날짜+시간 값 저장 및 가공
- **ZoneDateTime** — 날짜+시간+시간대 값 저장 및 가공
- **Duration** — 시간 - 시간
- **Period** — 날짜 - 날짜

날짜&시간 관련 작업 절차



java.util.
Date & Calendar

YEUNGJIN UNIVERSITY
Prof. Youngchul Jung

Calendar 클래스

- ✓ 시간 값을 캘린더 형식(년,월, 일, 시, 분, 초 등)의 값으로 저장
- ✓ 추상 클래스 -> new 연산 이용 직접 객체 생성 안됨
  - ✓ getInstance() 메서드 이용
- ✓ Time zone 지원
- ✓ Date 클래스와 변환 가능

Module java.base
Package java.util

Class Calendar

java.lang.Object
    java.util.Calendar

All Implemented Interfaces:

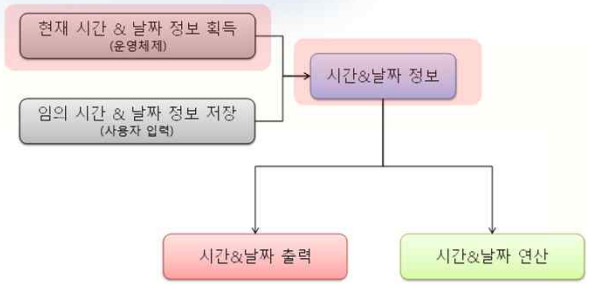
Serializable, Cloneable, Comparable<Calendar>

Direct Known Subclasses:

GregorianCalendar

```
public abstract class Calendar
extends Object
implements Serializable, Cloneable, Comparable<Calendar>
```

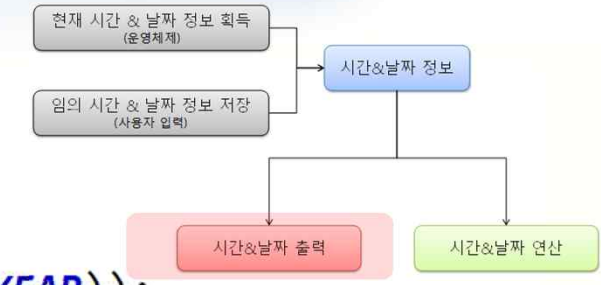
Calendar 클래스 활용 : 객체 생성 (날짜&시간 획득)




```
6  public class MyCalendar {
7      public static void main(String[] args) {
8          // 현 시스템(운영체제) 시간과 지역 설정에 따라 객체 생성
9          // 태국의 경우 BuddhistCalendar 그 외 지역은 GregorianCalendar
10
11         // Calendar 클래스는 추상클래스, static getInstance() 메서드 이용객체 생성
12         Calendar today = Calendar.getInstance();
13
14         if(today instanceof GregorianCalendar)
15             System.out.println("true"); // 결과 값 true
16     }
17 }
```

```
static Calendar    getInstance()
static Calendar    getInstance(Locale aLocale)
static Calendar    getInstance(TimeZone zone)
static Calendar    getInstance(TimeZone zone, Locale aLocale)
```

Calendar 클래스 활용 : 현재 시점 기준 날짜 정보 출력



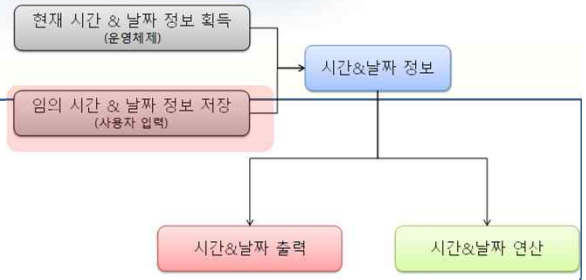

```
5  public class MyCalendar2 {
6      public static void main(String[] args) {
7          Calendar today = Calendar.getInstance();
8
9          System.out.println("년\t"    + today.get(Calendar.YEAR));
10         System.out.println("월\t"    + today.get(Calendar.MONTH));
11         System.out.println("일\t"    + today.get(Calendar.DATE));
12         System.out.println("요일\t"  + today.get(Calendar.DAY_OF_WEEK));
13         System.out.println("시\t"    + today.get(Calendar.HOUR));
14         System.out.println("분\t"    + today.get(Calendar.MINUTE));
15         System.out.println("초\t"    + today.get(Calendar.SECOND));
16         System.out.println("ms\t"    + today.get(Calendar.MILLISECOND));
17
18         System.out.println("주/월\t" + today.get(Calendar.DAY_OF_WEEK_IN_MONTH));
19         System.out.println("일/년\t" + today.get(Calendar.DAY_OF_YEAR));
20         System.out.println("주/년\t" + today.get(Calendar.WEEK_OF_YEAR));
21     }
22 }
```

```
년      2020
월      4
일      12
요일    3
시      3
분      14
초      15
ms      565
주/월   2
일/년   133
주/년   20
```

✓ 월(Month)는 0 ~ 11로 구성, 즉 1월은 0

✓ 요일은 일요일 : 1, 월요일 : 2 … 토요일 : 7로 구성

Calendar 클래스 활용 : 임의 시점 기준 날짜 정보 출력




```
5 public class MyClanedar1 {
6     public static void main(String[] args) {
7         // 현재 시간 획득
8         Calendar today = Calendar.getInstance();
9
10        // 시간 변경 -> 2020년 6월 16일 13시 20분 30초
11        today.set(2020, 05, 16, 13, 20, 30);
12
13        // 시간 정보 출력
14        System.out.println(today.get(Calendar.YEAR) + ":" + today.get(Calendar.MONTH)
15                + ":" + today.get(Calendar.DATE) + ":" + today.get(Calendar.HOUR_OF_DAY)
16                + ":" + today.get(Calendar.MINUTE) + ":" + today.get(Calendar.SECOND));
17    }
18 }
```

결과 값 : 2020:5:16:13:20:30

```
set(int field, int value)

set(int year, int month, int date)

set(int year, int month, int date, int hourOfDay,
int minute)

set(int year, int month, int date, int hourOfDay,
int minute, int second)
```

Calendar 클래스 활용 : 날짜 계산 (1)

```
기준일을 입력하세요(예 : 2020.05.16)
2020.05.01
오늘은 기준일로 부터 15째 되는 날입니다.
```

```
4  public class MyCalendar2 {
5      public static void main(String[] args) {
6          Scanner scn = new Scanner(System.in);
7          System.out.println("기준일을 입력하세요(예 : 2020.05.16)");
8
9          // 키보드로부터 입력 받은 기준일을 "." 구분으로 분리
10         String [] rcvdDate = scn.nextLine().split("\\.");
11
12         // 입력 받은 날짜 값을 Calendar 객체에 저장
13         Calendar baseDate   = Calendar.getInstance();
14         baseDate.set(Integer.valueOf(rcvdDate[0]), Integer.valueOf(rcvdDate[1]) - 1,
15                      Integer.valueOf(rcvdDate[2]));
16
17         // 현 시스템(OS)의 날짜 정보 획득
18         Calendar curDate    = Calendar.getInstance();
19
20         // 오늘 - 기준일 연산
21         long diffTimeStamp = curDate.getTimeInMillis() - baseDate.getTimeInMillis();
22
23         // 날짜로 환산 : 60초 * 1000ms * 60분 * 24시간
24         long diffDate =  diffTimeStamp / (3600000 * 24);
25
26         System.out.println("오늘은 기준일로 부터 " + diffDate + "째 되는 날입니다.");
27     }
28 }
```

Calendar 클래스 활용 : 날짜 계산 (2)

```
기준일로부터 100일째 되는 날 2000년 8월 9일
기준일로부터 100일전 되는 날 2000년 1월 22일
```

```
3 public class MyCalendar3 {
4     public static void main(String[] args) {
5         Calendar myDate = Calendar.getInstance();
6
7         // 2000년 5월 1일로 날짜를 설정
8         myDate.set(2000, 4, 1);
9
10        myDate.add(Calendar.DATE, 100);
11        System.out.println("기준일로부터 100일째 되는 날 " + prtDate(myDate));
12
13        // 2000년 5월 1일로 날짜를 설정
14        myDate.set(2000, 4, 1);
15        myDate.add(Calendar.DATE, -100);
16        System.out.println("기준일로부터 100일전 되는 날 " + prtDate(myDate));
17
18    }
19
20    public static String prtDate(Calendar argCal) {
21        return argCal.get(Calendar.YEAR) + "년 " + (argCal.get(Calendar.MONTH) + 1)
22                + "월 " + argCal.get(Calendar.DATE) + "일";
23    }
24 }
```

Calendar 클래스 활용 : 날짜 계산 (3)

```
기준일로부터 100일째 되는 날 2000년 4월 28일
기준일로부터 100일전 되는 날 2000년 5월 29일
```

```
3 public class MyCalendar3 {
4     public static void main(String[] args) {
5         Calendar myDate = Calendar.getInstance();
6 
7         // 2000년 5월 1일로 날짜를 설정
8         myDate.set(2000, 4, 1);
9 
10        myDate.add(Calendar.DATE, -3);
11        System.out.println("기준일로부터 100일째 되는 날 " + prtDate(myDate));
12 
13        // 2000년 5월 1일로 날짜를 설정
14        myDate.set(2000, 4, 1);
15        // roll은 add와 유사, 차이점 : 다른 필드에 영향을 미치지 않는다.
16        myDate.roll(Calendar.DATE, -3);
17        System.out.println("기준일로부터 100일전 되는 날 " + prtDate(myDate));
18 
19    }
20 
21    public static String prtDate(Calendar argCal) {
22        return argCal.get(Calendar.YEAR) + "년 " + (argCal.get(Calendar.MONTH) + 1)
23                + "월 " + argCal.get(Calendar.DATE) + "일";
24    }
25 }
```

달력 출력 프로그램 작성하기 (1)

```
기준일을 입력하세요(예 : 2020.05)
2020.06
Sun Mon Tue Wed Thu Fri Sat
                1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30
```

달력 출력 프로그램 작성하기 (2)

```
4  public class MyCalendar4 {
5      public static void main(String[] args) {
6          Scanner scn = new Scanner(System.in);
7          System.out.println("기준일을 입력하세요(예 : 2020.05)");
8
9          // 키보드로부터 입력 받은 기준일을 "." 구분으로 분리
10         String [] rcvdDate = scn.nextLine().split("\\.");
11
12         // 입력 받은 날짜 값을 Calendar 객체에 저장
13         Calendar baseDate   = Calendar.getInstance();
14         baseDate.set(Integer.valueOf(rcvdDate[0]), Integer.valueOf(rcvdDate[1]) - 1, 1);
15
16         // 현재 달 1일의 요일 구하기
17         int dayOfStartDate = baseDate.get(Calendar.DAY_OF_WEEK);
18
19         // <<-- 현재 달의 마지막일 구하기
20         Calendar endDate = Calendar.getInstance();
21         endDate.set(Integer.valueOf(rcvdDate[0]), Integer.valueOf(rcvdDate[1]), 1);
22         endDate.add(Calendar.DATE, -1); // 현재 달의 마지막일 = 다음달 1일 - 1
23         int endDateOfMonth = endDate.get(Calendar.DATE);
24         // -->>
```

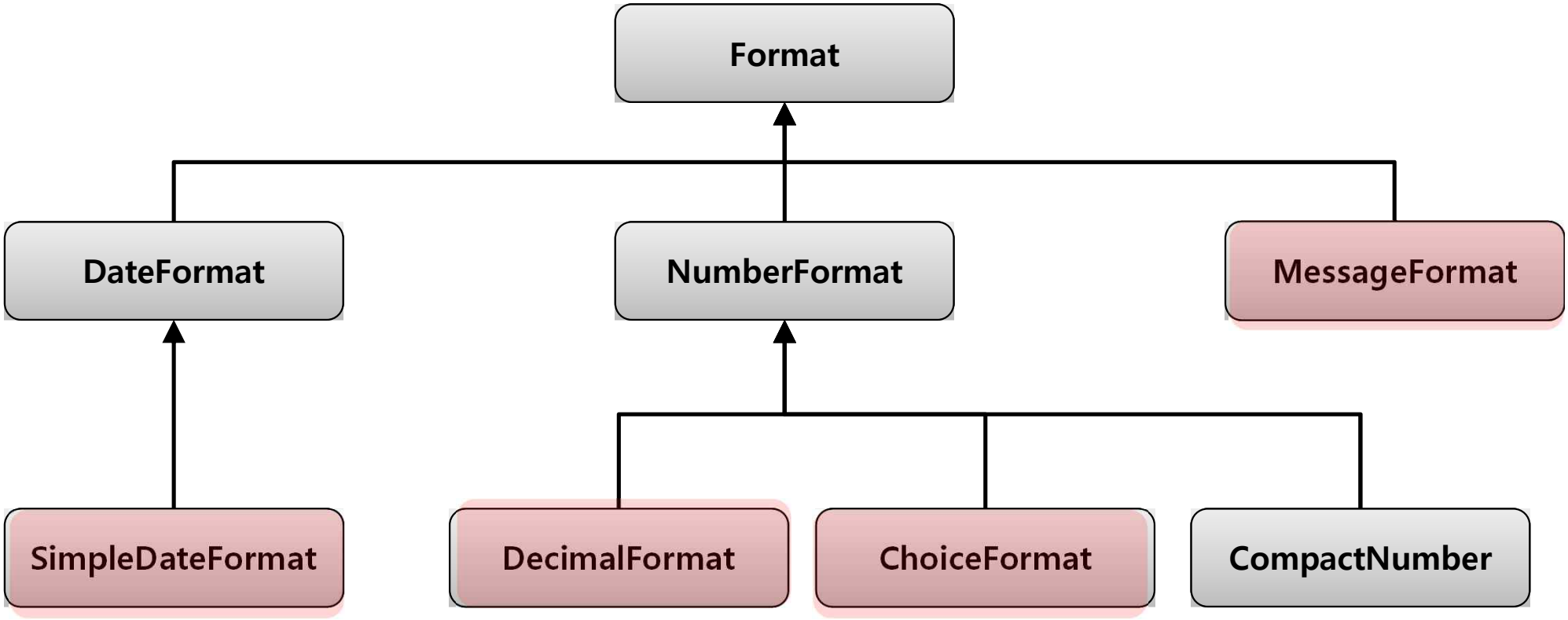
달력 출력 프로그램 작성하기 (3)

```
26          // <<-- 요일 출력 -->>
27          String day[] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
28          for(int i = 0 ; i < 7 ; i++)
29              System.out.print(day[i] + " ");
30          System.out.println();
31
32          // <<-- 첫째 주 공백 출력 -->>
33          for(int i = 1 ; i < dayOfStartDate ; i++)
34              System.out.print("    ");
35
36          // <<-- 날짜 출력 -->>
37          int dateCount = dayOfStartDate;
38          for(int i = 1 ; i <= endDateOfMonth; i++) {
39              System.out.print(( i < 10) ? "  "+ i : " " + i) + " ");
40              if(dateCount++ % 7 ==0)
41                  System.out.println();
42
43          }
44
45      }
46  }
```

Format Class

Format(형식화) 클래스

- 숫자, 날짜, 텍스트 데이터를 일정한 형식에 맞춰 출력할 수 있는 기능 제공



Format Class :

DecimalFormat, ChoiceFormat Class

YEUNGJIN UNIVERSITY
Prof. Youngchul Jung

Decimal Format

- 10진수 숫자를 패턴(정수, 실수, 지수, 백분율, 통화, 자리 맞춤 등)에 맞게 출력 또는 파싱

Module java.base
Package java.text

Class DecimalFormat

java.lang.Object
  java.text.Format
    java.text.NumberFormat
      java.text.DecimalFormat

All Implemented Interfaces:
Serializable, Cloneable

---

public class **DecimalFormat**
extends NumberFormat

```java
public class MyDate {
    public static void main(String[] args) {
        // #, 0은 10진수 출력
        // 차이점 0은 자리수 유지 : 빈자리 0으로 채운다, # 자리수 유지 X
        DecimalFormat df1 = new DecimalFormat("###");
        DecimalFormat df2 = new DecimalFormat("000");

        System.out.println(df1.format(12)); // 결과 값 : 12
        System.out.println(df2.format(12)); // 결과 값 : 012
    }
}
```

Decimal Format : 예제 (1)

```
 5 public class myFormat1 {
 6     public static void main(String[] args) {
 7         // "#" 이용 소수점 2자리 자리 맞춤
 8         DecimalFormat df1 = new DecimalFormat("#.##");
 9         System.out.println(df1.format(1234.567)); // 결과 값 1234.57, 반올림
10         System.out.println(df1.format(1234.564)); // 결과 값 1234.56
11         System.out.println(df1.format(1234.5));   // 결과 값 1234.5
12
13         // "0" 이용 소수점 4자리 자리 맞춤
14         DecimalFormat df2 = new DecimalFormat("#.0000");
15         System.out.println(df2.format(1234.56)); // 결과 값 1234.5600
16
17         // "#", "0" 이용 정수 자리 맞춤
18         DecimalFormat df3 = new DecimalFormat("###");
19         System.out.println(df3.format(12.56)); // 결과 값 13
20         // "#", "0" 이용 정수 자리 맞춤
21         DecimalFormat df4 = new DecimalFormat("000");
22         System.out.println(df4.format(12.56)); // 결과 값 013
23     }
24 }
```

Decimal Format : 예제 (2)

#.##E0

```
5  public class myFormat1 {
6      public static void main(String[] args) {
7          // "E" 이용 지수 표기 - 1
8          DecimalFormat df1 = new DecimalFormat("##.##E0");
9          System.out.println(df1.format(1234.567)); // 결과 값 12.35E2, 반올림
10
11         // "E" 이용 지수 표기 - 2
12         DecimalFormat df2 = new DecimalFormat("000000.E0");
13         System.out.println(df2.format(1234.56)); // 결과 값 123456E-2
14
15         // 단위구분
16         DecimalFormat df3 = new DecimalFormat("#,###");
17         System.out.println(df3.format(123456789)); // 결과 값 123,456,789
18
19         // % 백분율 환산
20         DecimalFormat df4 = new DecimalFormat("00.#%");
21         System.out.println(df4.format(0.1234)); // 결과 값 12.3
22
23         // 통화
24         DecimalFormat df5 = new DecimalFormat("\u00A4#");
25         System.out.println(df5.format(500)); // 결과 값 (통화: 원) 500
26
27         // +, -
28         DecimalFormat df6 = new DecimalFormat("#.#-");
29         System.out.println(df6.format(2.5)); // 결과 값 : 2.5-
30     }
31 }
```

Decimal Format : 예제 (3)

```
 5 public class myFormat2 {
 6     public static void main(String[] args) {
 7         // String -> Number 파싱
 8         DecimalFormat df1 = new DecimalFormat("\u00A4#,###");
 9         try {
10             String strValue = df1.format(1234567);
11             System.out.println(strValue); // W 1,234,567
12
13             // String ", W" 문자열 고려 후 숫자로 파싱
14             Number myNum = df1.parse(strValue);
15             System.out.println(myNum); // 1234567
16         } catch (Exception e) {
17             System.out.println("Exception : At DecimalFormat parsing");
18         };
19     }
20 }
```

Choice Format

- 특정 범위에 속하는 값을 문자열로 변환

```
X matches j if and only if limit[j] ≤ X < limit[j+1]
```

```
5 public class myFormat3 {
6     public static void main(String[] args) {
7         // 낮은 순으로 입력. 왜? -> X matches j if and only if limit[j] ≤ X < limit[j+1]
8         double limit[] = {0, 60, 70, 80, 90};
9         String grade[] = {"F", "D", "C", "B", "A"};
10
11        ChoiceFormat myForm = new ChoiceFormat(limit, grade);
12
13        System.out.println(myForm.format(20)); // F
14        System.out.println(myForm.format(65)); // D
15        System.out.println(myForm.format(75)); // C
16        System.out.println(myForm.format(85)); // B
17        System.out.println(myForm.format(95)); // A
18    }
19 }
```

Format Class :

SimpleDate Class

SimpleDate Format

- 날짜, 시간을 지정된 포맷으로 출력

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
Y	Week year	Year	2009; 09
M	Month in year (context sensitive)	Month	July; Jul; 07
L	Month in year (standalone form)	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day name in week	Text	Tuesday; Tue
u	Day number of week (1 = Monday, ..., 7 = Sunday)	Number	1
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800
X	Time zone	ISO 8601 time zone	-08; -0800; -08:00

SimpleDate Format : 예제 (1)

```
 6 public class myFormat5 {
 7     public static void main(String[] args) {
 8         Calendar today = Calendar.getInstance();
 9 
10         SimpleDateFormat myForm1 = new SimpleDateFormat("오늘은 y년 M월 d일 입니다");
11         SimpleDateFormat myForm2 = new SimpleDateFormat("현재 시간은 a  k시 m분 s초 입니다");
12 
13         // SimpleDateFormat은 Date 클래스만 지원
14         // 따라서 Calendar 사용 시, getTime() 메서드 이용 Date 클래스로 변환
15         System.out.println(myForm1.format(today.getTime()));
16         System.out.println(myForm2.format(today.getTime()));
17     }
18 }
```

SimpleDate Format : 예제 (2)

```
 8 public class myFormat5 {
 9     public static void main(String[] args)  {
10         SimpleDateFormat myForm1    = new SimpleDateFormat("y/M/d");
11         SimpleDateFormat myForm2    = new SimpleDateFormat("y년 M월 d일");
12
13         try {
14             // 문자열 "2020/05/18"을 Date형으로 변환. 문자열 날짜/시간 정보를 timestamp 값으로 변경
15             Date inputDate = myForm1.parse("2020/05/18");
16
17             // 변환 된 timestamp 값을 다시 문자열 형식으로 변환 후 출력
18             System.out.println(myForm2.format(inputDate));
19         } catch (Exception e) {
20             System.out.println("Parsing Exception");
21         }
22     }
23 }
```

Format Class :
MessageFormat Class

FormatClass

- **문자열 템플릿 기능 제공**

출력
- ✓ 문자열 템플릿
- ✓ 입력 값

→ format() → 문자열 템플릿+입력값

파싱
- ✓ 파싱 템플릿
- ✓ 입력 값

→ parse() → 파싱 된 값

```
5  public class myFormat6 {
6      public static void main(String[] args) {
7          Object obj[] = {"Youngchul Jung", "22", "010-1234-5678"};
8          String strTemplate = "이름 : {0}, 나이 : {1}, 전화번호 : {2}";
9
10         System.out.println(MessageFormat.format(strTemplate, obj));
11         // 결과 값 : 이름 : Youngchul Jung, 나이 : 22, 전화번호 : 010-1234-5678
12     }
13 }
```

FormatClass : 파싱 예제

```
5  public class myFormat7 {
6      public static void main(String[] args) {
7          MessageFormat myForm = new MessageFormat("{0}\t{1}\t{2}");
8
9          try {
10             Object values[] = myForm.parse("Youngchul Jung  22  010-1234-5678")
11             for(int i = 0 ; i < values.length ; i++)
12                 System.out.println((String)values[i]);
13             System.out.println();
14         } catch (Exception e) {
15             System.out.println("Parsing Exception");
16         }
17         // 결과 값 : Youngchul Jung 22 010-1234-5678
18     }
19 }
```

java.util.time package

YEUNGJIN UNIVERSITY
Prof. Youngchul Jung

Java.time package

- **기존 Date. Calendar 클래스의 오류 및 성능 개선을 위해 JDK 1.8부터 제공**
- **날짜와 시간을 별개의 클래스로 관리**
  - ✓ 날짜 : LocalDate, 시간 : LocalTime
  - ✓ 날짜+ 시간 : LocalDateTime
  - ✓ 날짜 + 시간 + 시간대 : ZonedDateTime
- **Immutable 속성**

✓ ***java.time package*** since JDK 1.8

Instant	▪ Timestamp 값 저장
LocalDate	▪ 날짜 값 저장 및 가공
LocalTime	▪ 시간 값 저장 및 가공
LocalDateTime	▪ 날짜+시간 값 저장 및 가공
ZonedDateTime	▪ 날짜+시간+시간대 값 저장 및 가공
Duration	▪ 시간 - 시간
Period	▪ 날짜 - 날짜

Java.time 패키지, 날짜/시간 작업 절차 : 객체 생성 (1)

✓ **now()**

현재 시간 & 날짜 정보 획득
(운영체제)

```
LocalTime       curTime      = LocalTime.now();
LocalDate       curDate      = LocalDate.now();
LocalDateTime   curDateTime  = LocalDateTime.now();
ZonedDateTime   curZDateTime = ZonedDateTime.now();
```

✓ **LocalTime**
✓ **LocalDate**
✓ **LocalDateTime**
✓ **ZoneDateTime**

시간&날짜 정보

시간&날짜 **출력**

시간&날짜 **설정**

시간&날짜 **연산**

✓ **of()**

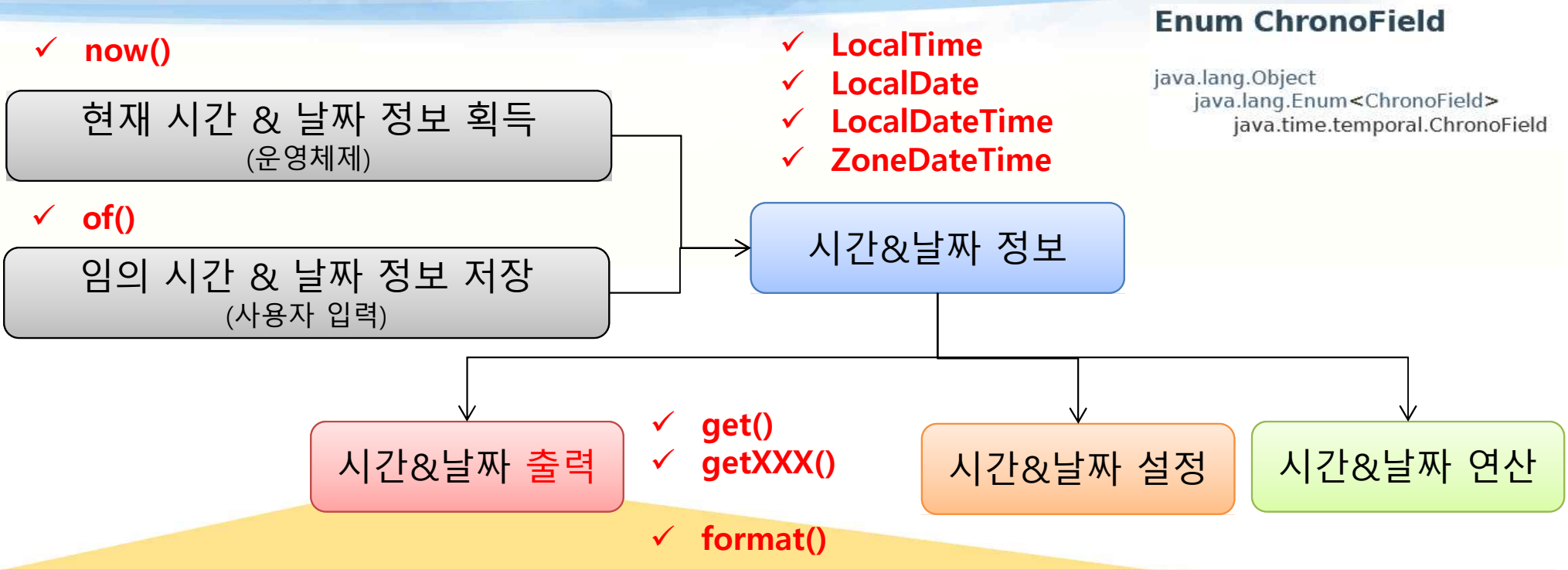
임의 시간 & 날짜 정보 저장
(사용자 입력)

```
LocalTime      setTime      = LocalTime.of(11,23,15);
LocalDate      setDate      = LocalDate.of(2020,05,19);
LocalDateTime  setDateTime  = LocalDateTime.of(setDate, setTime);
ZonedDateTime  setZDateTime = ZonedDateTime.of(setDateTime, ZoneId.of("Asia/Seoul"));
```

Java.time 패키지, 날짜/시간 작업 절차 : 객체 생성 (2)

```
LocalTime      curTime      = LocalTime.now();
LocalDate      curDate      = LocalDate.now();
LocalDateTime  curDateTime  = LocalDateTime.now();
ZonedDateTime  curZDateTime = ZonedDateTime.now();

System.out.println(curTime);      // 11:31:34.981
System.out.println(curDate);      // 2020-05-17
System.out.println(curDateTime);  // 2020-05-17T11:31:34.981
System.out.println(curZDateTime); // 2020-05-17T11:31:34.982+09:00[Asia/Seoul]

LocalTime     setTime      = LocalTime.of(11,23,15);
LocalDate     setDate      = LocalDate.of(2020,05,19);
LocalDateTime setDateTime  = LocalDateTime.of(setDate, setTime);
ZonedDateTime setZDateTime = ZonedDateTime.of(setDateTime, ZoneId.of("America/Phoenix"));

System.out.println(setTime);      // 11:23:15
System.out.println(setDate);      // 2020-05-19
System.out.println(setDateTime);  // 2020-05-19T11:23:15
System.out.println(setZDateTime); //2020-05-19T11:23:15+09:00[America/Phoenix]
```

Java.time 패키지, 날짜/시간 작업 절차 : 출력



Enum ChronoField

java.lang.Object
java.lang.Enum<ChronoField>
java.time.temporal.ChronoField

```
LocalDateTime tInfo = LocalDateTime.now();

System.out.println(tInfo.get(ChronoField.YEAR));          // 2020 년
System.out.println(tInfo.get(ChronoField.MONTH_OF_YEAR));// 5 월, 월은 1부터 시작
System.out.println(tInfo.get(ChronoField.DAY_OF_MONTH)); // 19 일
System.out.println(tInfo.get(ChronoField.DAY_OF_WEEK));  // 2, 요일은 월->일, 1부터 시작

System.out.println(tInfo.getHour());     // 13 시
System.out.println(tInfo.getMinute());   // 54 분
System.out.println(tInfo.getSecond());   // 56 초
```

Java.time 패키지, 날짜/시간 작업 절차 : 출력 – get()

Enum Constant	Description
ALIGNED_DAY_OF_WEEK_IN_MONTH	The aligned day-of-week within a month.
ALIGNED_DAY_OF_WEEK_IN_YEAR	The aligned day-of-week within a year.
ALIGNED_WEEK_OF_MONTH	The aligned week within a month.
ALIGNED_WEEK_OF_YEAR	The aligned week within a year.
AMPM_OF_DAY	The am-pm-of-day.
CLOCK_HOUR_OF_AMPM	The clock-hour-of-am-pm.
CLOCK_HOUR_OF_DAY	The clock-hour-of-day.
DAY_OF_MONTH	The day-of-month.
DAY_OF_WEEK	The day-of-week, such as Tuesday.
DAY_OF_YEAR	The day-of-year.
EPOCH_DAY	The epoch-day, based on the Java epoch of 1970-01-01 (ISO).
ERA	The era.
HOUR_OF_AMPM	The hour-of-am-pm.
HOUR_OF_DAY	The hour-of-day.
INSTANT_SECONDS	The instant epoch-seconds.
MICRO_OF_DAY	The micro-of-day.
MICRO_OF_SECOND	The micro-of-second.
MILLI_OF_DAY	The milli-of-day.
MILLI_OF_SECOND	The milli-of-second.
MINUTE_OF_DAY	The minute-of-day.
MINUTE_OF_HOUR	The minute-of-hour.
MONTH_OF_YEAR	The month-of-year, such as March.
NANO_OF_DAY	The nano-of-day.
NANO_OF_SECOND	The nano-of-second.
OFFSET_SECONDS	The offset from UTC/Greenwich.
PROLEPTIC_MONTH	The proleptic-month based, counting months sequentially from year 0.
SECOND_OF_DAY	The second-of-day.
SECOND_OF_MINUTE	The second-of-minute.
YEAR	The proleptic year, such as 2012.
YEAR_OF_ERA	The year within the era.

주의 : 음영이 표시된 필드는 getLong() 이용

```
int  get(TemporalField field);
long getLong(TemporalField field);
```

https://docs.oracle.com/en/java/javase/14/docs/api/java.base/java/time/temporal/ChronoField.html

Java.time 패키지, 날짜/시간 작업 절차 : 출력 – getXXX()

✓ LocalDate getXXX()

int	getDayOfMonth()	Gets the day-of-month field.
DayOfWeek	getDayOfWeek()	Gets the day-of-week field, which is an enum DayOfWeek.
int	getDayOfYear()	Gets the day-of-year field.
IsoEra	getEra()	Gets the era applicable at this date.
long	getLong TemporalField field)	Gets the value of the specified field from this date as a long.
Month	getMonth()	Gets the month-of-year field using the Month enum.
int	getMonthValue()	Gets the month-of-year field from 1 to 12.
int	getYear()	Gets the year field.

Java.time 패키지, 날짜/시간 작업 절차 : 출력 – getXXX()

✓ LocalTime getXXX()

int	getHour()	Gets the hour-of-day field.
long	getLong(TemporalField field)	Gets the value of the specified field from this time as a long.
int	getMinute()	Gets the minute-of-hour field.
int	getNano()	Gets the nano-of-second field.
int	getSecond()	Gets the second-of-minute field.

https://docs.oracle.com/en/java/javase/14/docs/api/java.base/java/time/temporal/ChronoField.html

Java.time 패키지, 날짜/시간 작업 절차 : 출력 – format()

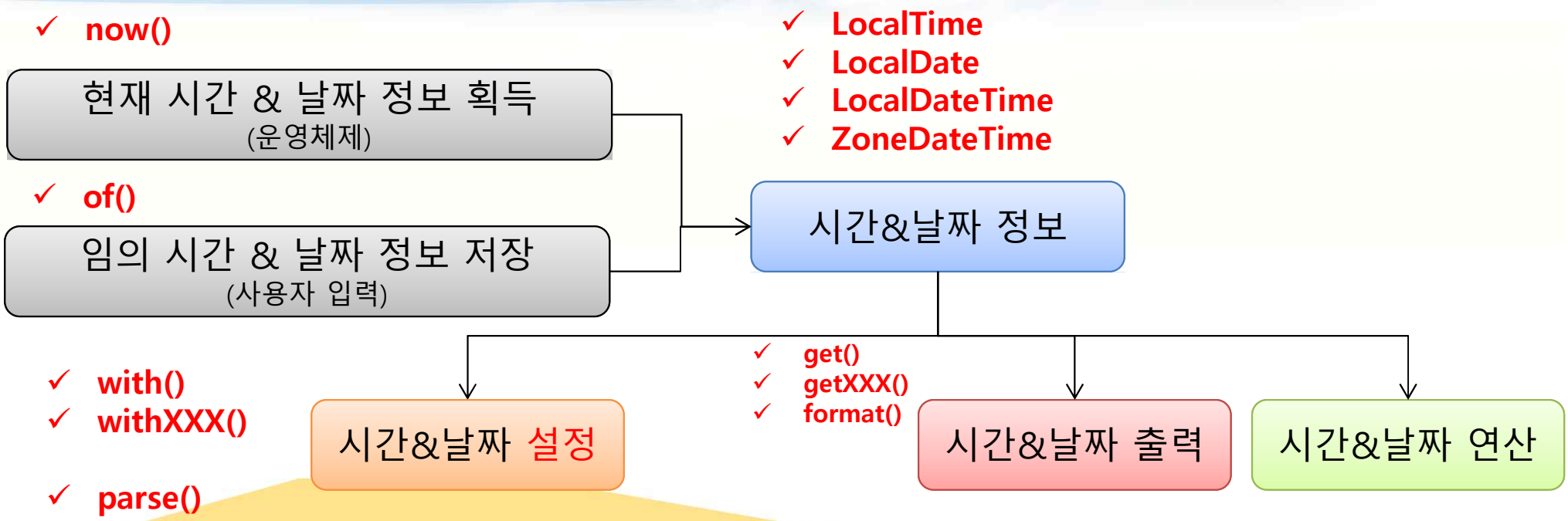
Module java.base
Package java.time.format

Class DateTimeFormatter

java.lang.Object
    java.time.format.DateTimeFormatter

```
LocalTime time_1 = LocalTime.now();
// DateTimeFormatter <- java.time 패키지에서 날짜&시간 출력을 위해 새롭게 정의한 Format 클래스
String strCurTime = time_1.format(DateTimeFormatter.ofPattern("[a] h시 m분 s초"));

System.out.println(strCurTime); // 오후 12시 8분 47초

LocalDate date_1 = LocalDate.now();
String strCurDate = date_1.format(DateTimeFormatter.ofPattern("[E] M. d. y"));

System.out.println(strCurDate); // 목 5. 21. 2020

LocalDateTime dateTime_1 = LocalDateTime.now();
String strCurDateTime =
        dateTime_1.format(DateTimeFormatter.ofPattern("[E] M. d. y [a] h시 m분 s초"));

System.out.println(strCurDateTime); // 목 5. 21. 2020 오후 12시 8분 47초
```

Java.time 패키지, 날짜/시간 작업 절차 : 설정 – with()




```
LocalDateTime tInfo = LocalDateTime.now();

tInfo = tInfo.with(ChronoField.YEAR, 2000);
tInfo = tInfo.with(ChronoField.MONTH_OF_YEAR, 5);
tInfo = tInfo.with(ChronoField.DAY_OF_WEEK, 1);
tInfo = tInfo.with(ChronoField.HOUR_OF_DAY, 13);
tInfo = tInfo.with(ChronoField.MINUTE_OF_HOUR, 13);

System.out.println(tInfo); // 2000-05-15T13:13:35.518
```

Java.time 패키지, 날짜/시간 작업 절차 : 설정 – withXXX()

```
LocalDateTime tInfo = LocalDateTime.now();

tInfo = tInfo.withYear(2000);
tInfo = tInfo.withMonth(5);
tInfo = tInfo.withDayOfMonth(1);
tInfo = tInfo.withHour(14);
tInfo = tInfo.withMinute(30);
tInfo = tInfo.withSecond(20);

System.out.println(tInfo); // 2000-05-01T14:30:20.703
```

✓ **LocalDate**

LocalDate	withDayOfMonth (intdayOfMonth)
LocalDate	withDayOfYear (intdayOfYear)
LocalDate	withMonth (intmonth)
LocalDate	withYear (int year)

✓ **LocalTime**

LocalTime	withHour (inthour)
LocalTime	withMinute (intminute)
LocalTime	withNano (intnanoOfSecond)
LocalTime	withSecond (int second)

Java.time 패키지, 날짜/시간 작업 절차 : 설정 – parse()

```
// parse() 메서드 이용 날짜 정보가 포함된 문자열을 파싱하여, LocalDate 객체로 반환
// 파싱 패턴 정의 시 DateTimeFormatter 클래스 이용, 출력 Letter 종류는 자바 API 문서 참조
// DateTimeFormatter는 파싱뿐만 아니라, format()메서드에도 이용된다.
LocalDate date_1 = LocalDate.now().parse("2020/05/24", DateTimeFormatter.ofPattern("yyyy/MM/dd"));

System.out.println(date_1); // 출력 값 : 2020-05-24

// 좀 더 세부적인 파싱정보를 정의하기 위해 DateTimeFormatterBuilder 클래스 제공
DateTimeFormatter formatter = new DateTimeFormatterBuilder()
            .parseCaseInsensitive()          // 문자열 대소문자 구분 X, 알파벳 Month [예: June] 처리
            .appendPattern("uu.MMMM.dd")     // 파싱 패턴 정의
            .toFormatter(Locale.ENGLISH);    // DateTimeformatterBuilder -> DateTimeFormatter로 변환
                                             // 변환 시 날짜 관련 문자 언어를 영어로 설정, Month[January], Day[Mon.]

LocalDate outputDate = LocalDate.parse("20.May.24", formatter);

System.out.println(outputDate); // 출력 값 : 2020-05-24
```

Module java.base
Package java.time.format

Class DateTimeFormatter

java.lang.Object
    java.time.format.DateTimeFormatter

public final class **DateTimeFormatter**
extends Object

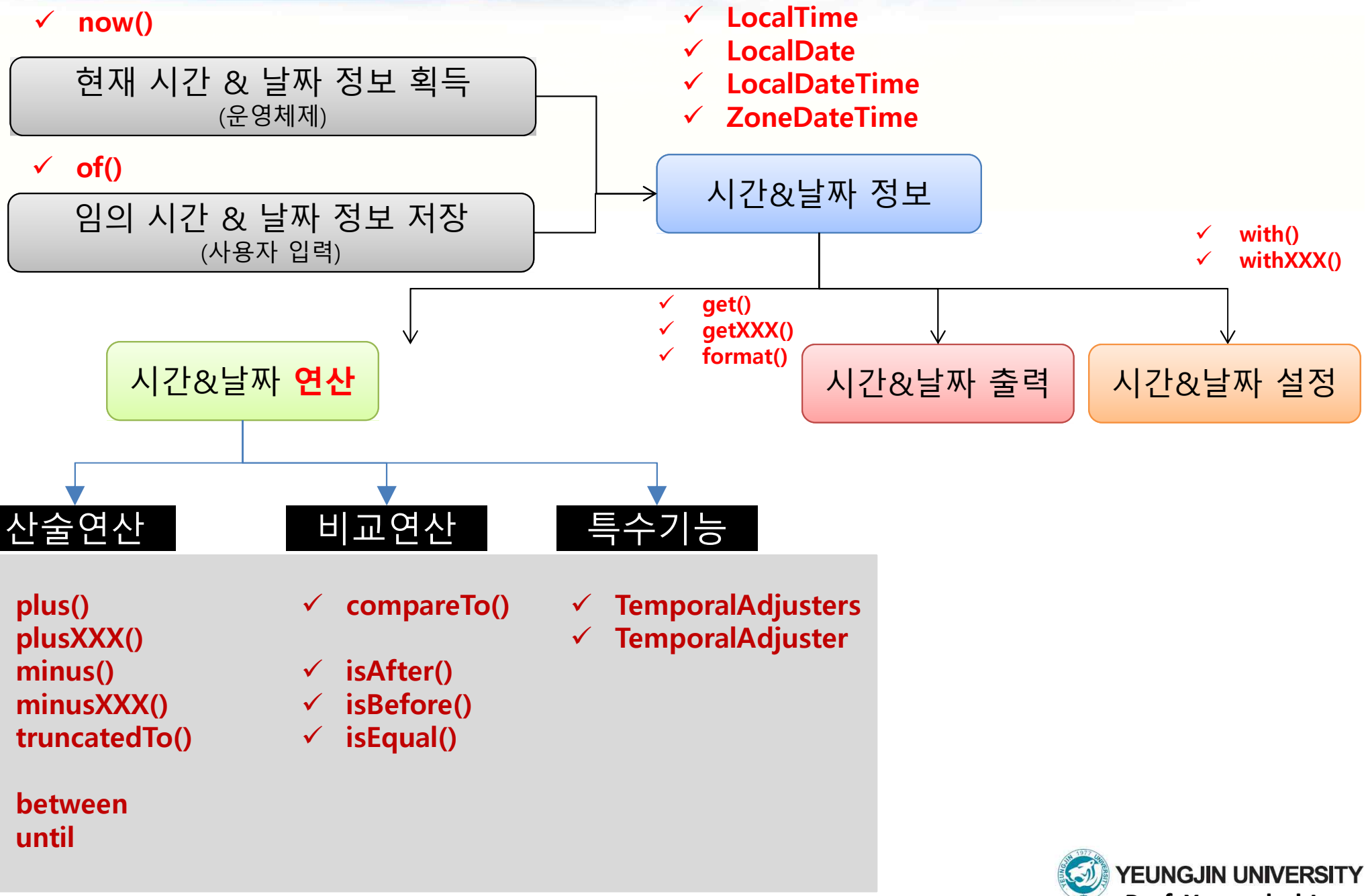
Module java.base
Package java.time.format

Class DateTimeFormatterBuilder

java.lang.Object
    java.time.format.DateTimeFormatterBuilder

public final class **DateTimeFormatterBuilder**
extends Object

Java.time 패키지, 날짜/시간 작업 절차 : 연산 (1)



Java.time 패키지, 날짜/시간 작업 절차 : 연산 – 산술연산 : plus, plusXXX(), minus(), minusXXX() (1)

```
// 2000년 5월 1일 13시 00분
LocalDateTime tInfo = LocalDateTime.of(2000,5,1,13,00);

// plus(), plusXXX(), minus(), minusXXX()
tInfo = tInfo.plus(Duration.ofHours(1)); // 2000-05-01T14:00
tInfo = tInfo.plus(Period.ofMonths(1));  // 2000-06-01T14:00
tInfo = tInfo.plusHours(1);              // 2000-06-01T15:00
tInfo = tInfo.plusMonths(1);             // 2000-07-01T15:00
tInfo = tInfo.plusMonths(-1);            // 2000-06-01T15:00
tInfo = tInfo.minus(Period.ofDays(3));   // 2000-05-29T15:00
tInfo = tInfo.minusMonths(1);            // 2000-04-29T15:00
```

Java.time 패키지, 날짜/시간 작업 절차 : 연산 – 산술연산 : plus, plusXXX(), minus(), minusXXX() (2)

LocalTime	plus (longamountToAdd, TemporalUnit unit)
LocalTime	plus (TemporalAmount amountToAdd)
LocalTime	plusHours (longhoursToAdd)
LocalTime	plusMinutes (longminutesToAdd)
LocalTime	plusNanos (longnanosToAdd)
LocalTime	plusSeconds (longsecondstoAdd)

LocalDate	plus (longamountToAdd, TemporalUnit unit)
LocalDate	plus (TemporalAmount amountToAdd)
LocalDate	plusDays (longdaysToAdd)
LocalDate	plusMonths (longmonthsToAdd)
LocalDate	plusWeeks (longweeksToAdd)

LocalDateTime	plus (longamountToAdd, TemporalUnit unit)
LocalDateTime	plus (TemporalAmount amountToAdd)
LocalDateTime	plusDays (longdays)
LocalDateTime	plusHours (longhours)
LocalDateTime	plusMinutes (longminutes)
LocalDateTime	plusMonths (longmonths)
LocalDateTime	plusNanos (longnanos)
LocalDateTime	plusSeconds (longseconds)
LocalDateTime	plusWeeks (longweeks)
LocalDateTime	plusYears (longyears)

Java.time 패키지, 날짜/시간 작업 절차 : 연산 – 산술연산 : period, duration classes

```
// between(), until() -> LocalDate, LocalTime 클래스만 적용
//  1) 날짜 - 날짜 = Period
//  2) 시간 - 시간 = Duration
LocalDate date1 = LocalDate.of(2019, 4, 1);
LocalDate date2 = LocalDate.of(2020, 5, 1);
// Period 예제, Period는 날짜 계산용으로 사용
Period diff1 = Period.between(date1, date2);

System.out.println(diff1.getYears());  // 1
System.out.println(diff1.getMonths()); // 1
System.out.println(diff1.getDays());   // 0

LocalTime time1 = LocalTime.of(12, 30, 00);
LocalTime time2 = LocalTime.of(14, 35, 30);
// Duration 예제, Duration은 시간 계산용으로 사용
// Duration의 경우 시간값 획들을 위해 getSeconds(), getNanoSecondes()만 제공
Duration diff2 = Duration.between(time1, time2);

int hour    = (int)(diff2.getSeconds() / 3600);
int min     = (int)((diff2.getSeconds() - hour * 3600) / 60);
int sec     = (int)(diff2.getSeconds() - hour * 3600 - min * 60);

System.out.println(hour);  // 2
System.out.println(min);   // 5
System.out.println(sec);   // 30
```

Java.time 패키지, 날짜/시간 작업 절차 : 연산 – 산술연산 : ChronoUnit class

```
LocalDateTime oldDate = LocalDateTime.of(1970, Month.JANUARY, 1, 00, 00, 00);
LocalDateTime newDate = LocalDateTime.now();

System.out.println(oldDate); // 1970-01-01T00:00
System.out.println(newDate); // 2020-05-24T13:53:56.002086700

// 항목별 편차 계산
long years   = ChronoUnit.YEARS.between(oldDate, newDate);
long months  = ChronoUnit.MONTHS.between(oldDate, newDate);
long weeks   = ChronoUnit.WEEKS.between(oldDate, newDate);
long days    = ChronoUnit.DAYS.between(oldDate, newDate);
long hours   = ChronoUnit.HOURS.between(oldDate, newDate);
long minutes = ChronoUnit.MINUTES.between(oldDate, newDate);
long seconds = ChronoUnit.SECONDS.between(oldDate, newDate);
long milis   = ChronoUnit.MILLIS.between(oldDate, newDate);
long nano    = ChronoUnit.NANOS.between(oldDate, newDate);

System.out.println("\n--- Total --- ");      // --- Total ---
System.out.println(years + " years");        // 50 years
System.out.println(months + " months");      // 604 months
System.out.println(weeks + " weeks");        // 2629 weeks
System.out.println(days + " days");          // 18406 days
System.out.println(hours + " hours");        // 441757 hours
System.out.println(minutes + " minutes");    // 26505473 minutes
System.out.println(seconds + " seconds");    // 1590328436 seconds
System.out.println(milis + " milis");        // 1590328436002 milis
System.out.println(nano + " nano");          // 1590328436002086700 nano
```

Java.time 패키지, 날짜/시간 작업 절차 : 연산 – 산술연산 : LocalTime truncatedTo()

```
LocalTime time_1 = LocalTime.of(14, 25, 31,100100125);
System.out.println(time_1); // 14:25:31.100100125

// trucatedTo() -> 지정된 단위 미만은 0으로 조정
// LocalTime만 존재
time_1 = time_1.truncatedTo(ChronoUnit.MICROS);
System.out.println(time_1); // 14:25:31.100100

time_1 = time_1.truncatedTo(ChronoUnit.MILLIS);
System.out.println(time_1); // 14:25:31.100

time_1 = time_1.truncatedTo(ChronoUnit.SECONDS);
System.out.println(time_1); // 14:25:31

time_1 = time_1.truncatedTo(ChronoUnit.MINUTES);
System.out.println(time_1); // 14:25

time_1 = time_1.truncatedTo(ChronoUnit.HOURS);
System.out.println(time_1); // 14:00
```

Java.time 패키지, 날짜/시간 작업 절차 : 연산 – 비교 : compareTo() (1)

```
24          LocalTime time_1 =
25                  LocalTime.parse("14:30:00", DateTimeFormatter.ofPattern("HH:mm:ss"));
26          LocalTime time_2 =
27                  LocalTime.parse("14:30:05", DateTimeFormatter.ofPattern("HH:mm:ss"));
28          LocalTime time_3 =
29                  LocalTime.parse("16:50:00", DateTimeFormatter.ofPattern("HH:mm:ss"));
30
31          // 시간 A > 시간 B => 1, 시간 A == 시간 B => 0, 시간 A < 시간 B => -1
32          System.out.println(time_2.compareTo(time_1)); // 1,
33          System.out.println(time_2.compareTo(time_2)); // 0
34          System.out.println(time_2.compareTo(time_3)); // -1
35
36
37          LocalDate date_1 =
38                  LocalDate.parse("2020/05/15", DateTimeFormatter.ofPattern("yyyy/MM/dd"));
39          LocalDate date_2 =
40                  LocalDate.parse("2020/06/01", DateTimeFormatter.ofPattern("yyyy/MM/dd"));
41          LocalDate date_3 =
42                  LocalDate.parse("2020/06/03", DateTimeFormatter.ofPattern("yyyy/MM/dd"));
43
44          // date_1 > date_2 => Positive value, date_1 == date_2 => 0
45          // date_1 < date_2 => Negative value
46          // 비교 대상 날짜가 같은 달인 경우 날짜 편차 반환, 비교 대상 날짜가 다른 달인 경우 달의 편차 반환
47          System.out.println(date_2.compareTo(date_1)); // 1,
48          System.out.println(date_2.compareTo(date_2)); // 0
49          System.out.println(date_2.compareTo(date_3)); // -2
```

Java.time 패키지, 날짜/시간 작업 절차 : 연산 – 비교 : compareTo() (2)

```
// 2020년 5월 1일 0시 0분 0초
LocalDateTime dateTime_1 = LocalDateTime.of(2020,5,01,0,0,0);
// 2020년 5월 20일 12시 30분 0초
LocalDateTime dateTime_2 = LocalDateTime.of(2020,5,20,12,30,0);
// 2020년 8월 20일 15시 45분 0초
LocalDateTime dateTime_3 = LocalDateTime.of(2020,8,20,15,45,0);
// 2020년 5월 20일 15시 44분 0초
LocalDateTime dateTime_4 = LocalDateTime.of(2020,5,20,15,44,0);

System.out.println(dateTime_2.compareTo(dateTime_1)); // 19, Positive, day 편차
System.out.println(dateTime_2.compareTo(dateTime_2)); // 0,  날짜 + 시간 같음
System.out.println(dateTime_2.compareTo(dateTime_3)); // -3, Negative, Month 편차
System.out.println(dateTime_2.compareTo(dateTime_4)); // -1, Negative
```

Java.time 패키지, 날짜/시간 작업 절차 :
연산 – 비교 : isAfter(), istBefore(), isEqual()

```
LocalDateTime dateTime_1 = LocalDateTime.of(2020, 04, 20, 13, 0, 0);
LocalDateTime dateTime_2 = LocalDateTime.of(2020, 05, 01, 13, 0, 0);
LocalDateTime dateTime_3 = LocalDateTime.of(2020, 07, 30, 13, 0, 0);

// date1.isAfter(date2) : date1 >  date2 => true else false
// date1.isBefore(date2): date1 <  date2 => true else false
// date1.isEqual(date2) : date1 == date2 => true else false

System.out.println(dateTime_2.isAfter(dateTime_1)); // true
System.out.println(dateTime_2.isAfter(dateTime_2)); // false
System.out.println(dateTime_2.isAfter(dateTime_3)); // false

System.out.println(dateTime_2.isBefore(dateTime_1)); // false
System.out.println(dateTime_2.isBefore(dateTime_2)); // false
System.out.println(dateTime_2.isBefore(dateTime_3)); // true

System.out.println(dateTime_2.isEqual(dateTime_1)); // false
System.out.println(dateTime_2.isEqual(dateTime_2)); // true
System.out.println(dateTime_2.isEqual(dateTime_3)); // false
```

Java.time 패키지, 날짜/시간 작업 절차 : 특수기능 – TemporalAdjusters class (1)

Enum DayOfWeek

java.lang.Object
java.lang.Enum<DayOfWeek>
java.time.DayOfWeek

Enum Constant
FRIDAY
MONDAY
SATURDAY
SUNDAY
THURSDAY
TUESDAY
WEDNESDAY

✓ Plus, minus 외 자주 쓰이는 날짜 계산 기능 제공

Class TemporalAdjusters

java.lang.Object
java.time.temporal.TemporalAdjusters

```
// 2020년 5월 21일 14시 17분 00초
LocalDateTime dateTime_1 = LocalDateTime.of(2020, 5, 21, 14, 17, 00);

// 현재 달의 마지막 일(Day)을 획득
LocalDateTime lastDayOfMonth = dateTime_1.with(TemporalAdjusters.lastDayOfMonth());
System.out.println(lastDayOfMonth.getDayOfMonth()); // 31


LocalDateTime firstDayofWeek =
        // 현재 달의 첫 째주 일요일의 날짜 획득
        dateTime_1.with(TemporalAdjusters.firstInMonth(DayOfWeek.SUNDAY));
System.out.println(firstDayofWeek.getDayOfMonth()); // 3
```

Java.time 패키지, 날짜/시간 작업 절차 : 특수기능 – TemporalAdjusters class (2)

TemporalAdjuster	dayOfWeekInMonth (int ordinal, DayOfWeek dayOfWeek)	Returns the day-of-week in month adjuster, which returns a new date with the ordinal day-of-week based on the month.
static TemporalAdjuster	firstDayOfMonth()	Returns the "first day of month" adjuster, which returns a new date set to the first day of the current month.
static TemporalAdjuster	firstDayOfNextMonth()	Returns the "first day of next month" adjuster, which returns a new date set to the first day of the next month.
static TemporalAdjuster	firstDayOfNextYear()	Returns the "first day of next year" adjuster, which returns a new date set to the first day of the next year.
static TemporalAdjuster	firstDayOfYear()	Returns the "first day of year" adjuster, which returns a new date set to the first day of the current year.
static TemporalAdjuster	firstInMonth (DayOfWeek dayOfWeek)	Returns the first in month adjuster, which returns a new date in the same month with the first matching day-of-week.
static TemporalAdjuster	lastDayOfMonth()	Returns the "last day of month" adjuster, which returns a new date set to the last day of the current month.
static TemporalAdjuster	lastDayOfYear()	Returns the "last day of year" adjuster, which returns a new date set to the last day of the current year.
static TemporalAdjuster	lastInMonth (DayOfWeek dayOfWeek)	Returns the last in month adjuster, which returns a new date in the same month with the last matching day-of-week.
static TemporalAdjuster	next (DayOfWeek dayOfWeek)	Returns the next day-of-week adjuster, which adjusts the date to the first occurrence of the specified day-of-week after the date being adjusted.
static TemporalAdjuster	nextOrSame (DayOfWeek dayOfWeek)	Returns the next-or-same day-of-week adjuster, which adjusts the date to the first occurrence of the specified day-of-week after the date being adjusted unless it is already on that day in which case the same object is returned.
static TemporalAdjuster	ofDateAdjuster (UnaryOperator <LocalDate> dateBasedAdjuster)	Obtains a TemporalAdjuster that wraps a date adjuster.
static TemporalAdjuster	previous (DayOfWeek dayOfWeek)	Returns the previous day-of-week adjuster, which adjusts the date to the first occurrence of the specified day-of-week before the date being adjusted.
static TemporalAdjuster	previousOrSame (DayOfWeek dayOfWeek)	Returns the previous-or-same day-of-week adjuster, which adjusts the date to the first occurrence of the specified day-of-week before the date being adjusted unless it is already on that day in which case the same object is returned.

Java.time 패키지, 날짜/시간 작업 절차 : 특수기능 – TemporalAdjuster 인터페이스를 이용한 기능 확장 (1)

```
10 class AdmissionDateById implements TemporalAdjuster {
11     private String      StdId;
12     private LocalDate   admissionDate;
13
14     AdmissionDateById(String argStdId) {
15         // 입력 학번에서 년도 2자리 획득, 예) 1708735 -> 17년도
16         StdId = argStdId.substring(0, 2);
17     }
18
19     public Temporal adjustInto(Temporal temporal) {
20         // String 17을 4자리 년도로 변환, 예) 17 -> 2017
21         // Year는 java.time package 내 클래스, 연도 처리 관련 클래스
22         Year enteringYear = Year.parse(StdId, DateTimeFormatter.ofPattern("uu"));
23
24         // 학번 기준 연도, 3월 2일로 날짜 설정
25         admissionDate = LocalDate.of(enteringYear.getValue(), 3, 2);
26
27         // 입학일(Day) 조정, 입학일은 해당 년도 3월 1일 이후 첫 번째 평일
28         // 3월 2일이 주말(토, 일)일 경우 현재 day에서 1 ~ 2일 증가
29         if (admissionDate.getDayOfWeek().getValue() == 6 || admissionDate.getDayOfWeek().getValue() == 7) {
30             if (admissionDate.getDayOfWeek().getValue() == 6)
31                 admissionDate = admissionDate.plusDays(2);
32             else
33                 admissionDate = admissionDate.plusDays(1);
34         }
35
36         return admissionDate;
37     }
38 }
```

Java.time 패키지, 날짜/시간 작업 절차 :
특수기능 – TemporalAdjuster 인터페이스를 이용한 기능 확장 (2)

```
40 public class TP10 {
41     public static void main(String[] args) {
42         // 학생 학번을 입력하면, 해당 년도의 입학일을 반환
43         // 영진전문대학교 학번은 7자리로 구성, 학번 앞 두 자리는 입학년도 의미
44         // 입학일은 3월 1일 이후, 첫 번째 평일
45         LocalDate admissionDate_1 = LocalDate.now().with(new AdmissionDateById("2008735"));
46         System.out.println(admissionDate_1); // 2020-03-02
47 
48         LocalDate admissionDate_2 = LocalDate.now().with(new AdmissionDateById("1908735"));
49         System.out.println(admissionDate_2); // 2019-03-04
50 
51         LocalDate admissionDate_3 = LocalDate.now().with(new AdmissionDateById("1408735"));
52         System.out.println(admissionDate_3); // 2014-03-03
53 
54         LocalDate admissionDate_4 = LocalDate.now().with(new AdmissionDateById("1308735"));
55         System.out.println(admissionDate_4); // 2013-03-04
56     }
57 }
```

Thanks