10장(날짜와 시간 & 형식화)

Calendar와 Date

Calendar와 GregorianCalendar

- **Calendar**는 추상클래스이기 때문에 직접 객체를 생성할 수 없음
- **BuddhistCalendar 클래스** → **태국**인경우
- **GregorianCalendar 클래스** → 태국이외의 국가
- **get( ) 메서드로 월을 가져오면 1~12가 아닌 0~11로 가져오기때문에 +1을 해주어야함**
- **두 날짜간의 차이를 구학 위해서는 두 날짜를 최소단위인 초단위로 변경 후 차이를 구해야함**

Date와 Calendar간의 변환

- Calendar → Date

```
1  Calendar cal = Calendar.getInstance();
2  Date d = new Date(cal.getTimeInMillis());   //Date(long date)
```

- Date → Calendar

```
1  Date d = new Date();
2  Calendar cal = Calendar.getInstance();
3  cal.setTime(d);
```

add와 roll의 차이점

	차이점	예시
add	다른 필드에 영향을 줌	일필드를 31일 증가시키면 월 필드값도 1 증가
roll	다른 필드에 영향을 안 줌	일필드를 31일 증가시키면 월 필드값 변화없음 예외) 일필드가 말일일 때, 월필드를 변경하면 일필드에 영향을 미칠 수 있음

boolean isLeapYear(int year)
: 매개변수 year가 윤년이면 true를 그렇지 않으면 false를 반환한다.
int dayDiff(int y1, int m1, int d1, int y2, int m2, int d2)
: 두 날짜간의 차이를 일단위로 반환한다.
int getDayOfWeek(int year, int month, int day)
: 지정한 날짜의 요일을 반환한다.(1~7, 1이 일요일)
String convertDayToDate(int day)
: 일단위의 값을 년월일의 형태의 문자열로 변환하여 반환한다.
int convertDateToDay(int year, int month, int day)
: 년월일을 입력받아서 일단위로 변환한다.

DecimalFormat

▼ **패턴에 사용되는 기호**

기호	의미	패턴	결과(1234567.89)
0	10진수(값이 없을 때는 0)	0 0.0 0000000000.0000	1234568 1234567.9 0001234567.8900
#	10진수	# #.# ##########.####	1234568 1234567.9 1234567.89
.	소수점	#.#	1234567.9
-	음수부호	#.#- -#.#	1234567.9- -1234567.9
,	단위 구분자	#,###.## #,####.##	1,234,567.89 123,4567.89
E	지수기호	#E0 0E0 ##E0 00E0 ####E0 0000E0 #.#E0 0.0E0 0.000000000E0 00.00000000E0 000.0000000E0 #.##########E0 ##.########E0 ###.#######E0	.1E7 1E6 1.2E6 12E5 123.5E4 1235E3 1.2E6 1.2E6 1.234567890E6 12.34567890E5 123.4567890E4 1.23456789E6 1.23456789E6 1.23456789E6
;	패턴구분자	#,###.##+;#,###.##-	1,234,567.89+ (양수일 때) 1,234,567.89- (음수일 때)
%	퍼센트	#.#%	123456789%
\u2030	퍼밀(퍼센트 x 10)	#.#\u2030	1234567890‰
\u00A4	통화	\u00A4 #,###	₩ 1,234,568
'	escape문자	'#'#,### ''#,###	#1,234,568 '1,234,568

DecimalFormat

- 숫자를 형식화하는데 사용하는 클래스
- **parse( )** → 기호와 문자가 포함된 문자열을 숫자로 쉽게 변환할 수 있다.
  - **Integer.parseInt( ) 는 콤마(,)를 변환 할 수 없음 따라서 사용 못함**

- 변환 후 **doubleValue( )** or **intVlaue( )** 등으로 사용

▼ **예시**

```java
Number num = 1234567.89;
DecimalFormat df = new DecimalFormat("#,###.##");
String result = df.format(num);

System.out.println(result);

try {
    Number num2 = df.parse(result); //Number형인 수

    System.out.println(num2);

    double d = num2.doubleValue();  //Number형인 수를 double형으로 변경

} catch (ParseException e) {

    e.printStackTrace();
}
```

사용방법

- 원하는 출력형식의 패턴을 작성하여 **DecimalFormat 인스턴스** 생성
- 출력하고자 하는 문자열로 **format메서드**를 호출

```java
double number = 1234567.89;
DecimalFormat df = new DecimalFormat("#.#E0");
String result = df.format(number);

System.out.println(result); //1.2E6
```

SimpleDateFormat

▼ 패턴에 사용되는 기호

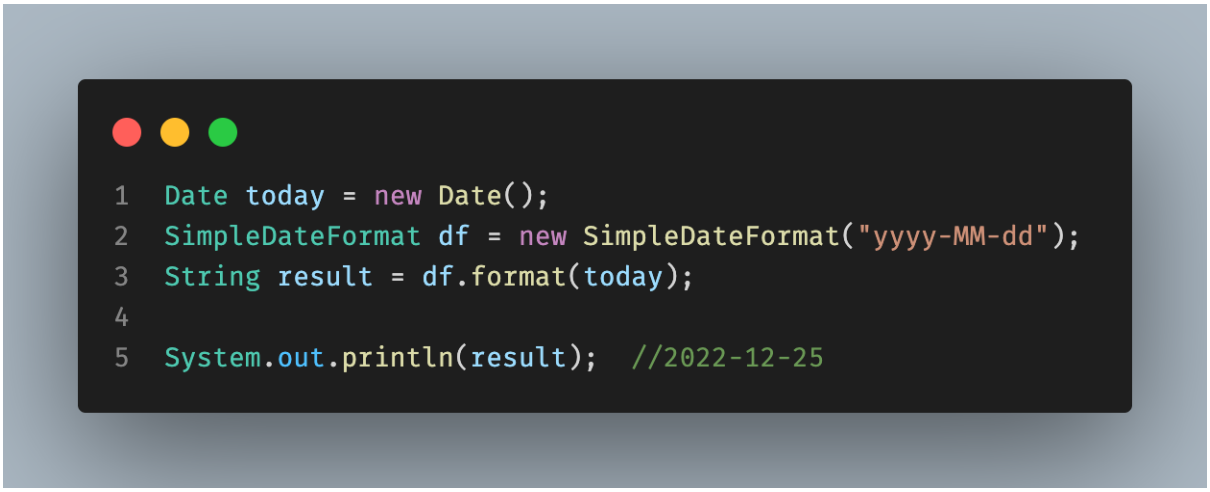
기호	의미	보기
G	연대(BC, AD)	AD
y	년도	2006
M	월(1~12 또는 1월~12월)	10 또는 10월, OCT
w	년의 몇 번째 주(1~53)	50
W	월의 몇 번째 주(1~5)	4
D	년의 몇 번째 일(1~366)	100
d	월의 몇 번째 일(1~31)	15
F	월의 몇 번째 요일(1~5)	1
E	요일	월
a	오전/오후(AM, PM)	PM
H	시간(0~23)	20
k	시간(1~24)	13
K	시간(0~11)	10
h	시간(1~12)	11
m	분(0~59)	35
s	초(0~59)	55
S	천분의 일초(0~999)	253
z	Time zone(General time zone)	GMT+9:00
Z	Time zone(RFC 822 time zone)	+0900
'	escape문자(특수문자를 표현하는데 사용)	없음

SimpleDateFormat

- **Date인스터스만 format( )에 사용될 수 있기 때문에 Calendar인스턴스는 Date인스턴스로 변환해야 함**
- **parse(String source) 메서드**
  - 문자열 **source**를 날짜 **Date인스턴스**로 변환
  - 포멧형식을 변경 할 때 **substring메서드**를 이용해 년, 월, 일을 뽑아내는 수고로움을 덜어줌

사용방법

- 원하는 출력형식의 패턴을 작성하여 **SimpleDateFormat 인스턴스** 생성
- 출력하고자 하는 **Data인스턴스**를 가지고 **format(Date d)** 호출
- 반환형은 String



```
1  Date today = new Date();
2  SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");
3  String result = df.format(today);
4
5  System.out.println(result);  //2022-12-25
```

ChoiceFormat

choiceFormat

- 연속적 or 불연속적인 범위의 값들을 처리
- 복잡하게 처리되던 코드를 간단하고 직관적으로 만듬
- **if - else if 문** or **switch - case 문** 대신 활용가능(?)

```java
import java.text.*;

class ChoiceFormatEx1 {
   public static void main(String[] args) {
       double[] limits = {60, 70, 80, 90};    // 낮은 값부터 큰 값의 순서로 적어야한다.
       // limits, grades간의 순서와 개수를 맞추어야 한다.
       String[] grades = {"D", "C", "B", "A"};

       int[] scores = { 100, 95, 88, 70, 52, 60, 70};

       ChoiceFormat form = new ChoiceFormat(limits, grades);

       for(int i=0;i<scores.length;i++) {
           System.out.println(scores[i]+":"
                              +form.format(scores[i]));
       }
   } // main
}
```

▼ 실행결과

```
100:A
95:A
88:B
70:C
52:D
60:D
70:C
```

- 치환될 문자열 갯수

  = 구분할 범위값 갯수

  필수

- 범위값 오름차순 정렬 필수

```java
import java.text.*;

class ChoiceFormatEx2 {
   public static void main(String[] args) {
       String pattern = "60#D|70#C|80<B|90#A";
       int[] scores = { 91, 90, 80, 88, 70, 52, 60};

       ChoiceFormat form = new ChoiceFormat(pattern);

       for(int i=0;i<scores.length;i++) {
           System.out.println(scores[i]+":"+form.format(scores[i]));
       }
   } // main
}
```

▼ 실행결과

```
91:A
90:A
80:C
88:B
70:C
52:D
60:D
```

- '#' → 이하를 의미
- '<' → 미만을 의미

MessageFormat

MessageFormat

- 데이터를 정해진 양식에 맞게 출력하게 도와줌
- { } - 의 숫자는 포멧할 데이터의 인덱스 값을 의미

```
import java.text.*;

class MessageFormatEx1 {
   public static void main(String[] args) {
       String msg = "Name: {0} \nTel: {1} \nAge:{2} \nBirthday:{3}";

       Object[] arguments = {
           "이자바","02-123-1234", "27", "07-09"
       };

       String result =
             MessageFormat.format(msg, arguments);
       System.out.println(result);
   }
}
```

▼ 실행결과

```
Name: 이자바
Tel: 02-123-1234
Age:27
Birthday:07-09
```