3-11.日付操作クラス

提出する 評定を受ける

■日付操作クラス

はじめに

本カリキュラムで説明するDateクラスやCalendarクラスを使用すると、
特定の日付と時刻に関するプログラムについて操作することができます。
オラクルが提供するjava.utilパッケージ内には、コレクションフレームワークの他に日付操作クラスなど様々なクラスがあります。

現場では、プログラムの処理中の **「値」** だけではなく、
「時間」 も大切な情報として記録することがほとんどなので、しっかり身につけましょう！

Step1: Dateクラス

Dateクラスは、日付を管理しているクラスです。
簡単に現在の日付と時刻を取得することができるクラスです。
ただし、日付の計算などを行う場合には、適していませんのでご注意ください。

例題

以下の通りコーディングし、Dateクラスのインスタンスを生成後、コンソールへ出力しなさい。

```
import java.util.Date;

public class DateMain{

    public static void main(String[] args){
        Date now = new Date();
        System.out.println(now);
    }
}
```

【実行結果】

```
Mon Dec 12 14:25:22 JST 2011
```

補足

・タイムゾーン
出力内容の JST のことで、日本標準時刻が世界標準+9時間であることを示しています。

世界標準時刻は、 **GMT（グリニッジ標準時）** と呼ばれ、
その **GMT** を基準として、日本であれば **JST** 、アメリカであれば **PST** のように表記し、
「 **GMT ± n時間の差** 」があることを明示的に示します。

上記 **GMT** に微調整を加えた、 **UTC（協定世界時）** というものもありますが、
今は時刻の基準があるんだなーという認識で問題ありません。
（現場に出て、こういった時刻を扱うことが出てきたら「タイムゾーン」、「GMT」、「UTC」などについて詳しく調べてみるといいでしょう！

Step2: Calendarクラス

Calendarクラスは、日時の計算などを行う場合に使用します。
例えば、 **「今日から1ヶ月と2週間後（45日後）」** というような
少しややこしい日付の計算をしなければいけない時に、この Calendarクラス が活躍してくれます。
Step1に書いてある Dateクラス とは、ここが違うポイントになりますね。

例題（Calendarクラスから呼び出し可能な日付や時刻）

以下の通りコーディングし、カレンダークラスの `getTime()` メソッドを実行しなさい。

```
import java.util.Calendar;

public class DateMain2 {

    public static void main(String[] args){
        Calendar cal = Calendar.getInstance();
        System.out.println(cal.getTime());
    }
}
```

【実行結果】

```
Thu Jun 06 19:22:17 JST 2019
```

`new Date` の際の出力内容と同じフォーマットの結果値が得られます。

Calendarクラスに用意されている「主要フィールド」一覧

フィールド	用途
YEAR	年
MONTH	月（0〜11）
DATE, DAY_OF_MONTH	現在の月の何日目かを表す
WEEK_OF_MONTH	現在の月の何週目かを表す
DAY_OF_WEEK	現在の週の何日目かを表す
AM_PM	午前午後を表す（午前0、午後1）
HOUR	午前または午後の何時かを表す（0〜11）
HOUR_OF_DAY	24時制で何時かを表す
MINUTE	何分かを表す
SECOND	何秒かを表す

例題

それでは、どのようにすれば **日付の計算** ができるのか、
下記に示す`Calendar`クラスのサンプルプログラムを作成して、実行してみましょう。

```
import java.util.Calendar;

public class CalendarMain {

    public static void main(String[] args) {
        // 今日が2019/03/06の場合の年月日を表示
        Calendar cal = Calendar.getInstance();

        System.out.println(cal.get(Calendar.YEAR));
        System.out.println(cal.get(Calendar.MONTH) + 1);
        System.out.println(cal.get(Calendar.DATE));
    }
}
```

【実行結果】

```
2019
3
6
```

カレンダークラスで定義されている `Calendar.MONTH`は
`JANUARY(実値=0)`〜`DECEMBER(実値=11)` まで定義が存在しています。
しかし、 0-11 だと、実際の暦を表す月(1-12)から 1 を引いた値であるため、
上記のサンプルコードでは、 `Calendar.MONTH` に `+1` することで実際に表示される月の値を調整しています。

ややこしい部分ですが、各月の英語名に対して対応番号が下記のようになります。

> JANUARY(0), FEBRUARY(1), MARCH(2), APRIL(3), MAY(4), JUNE(5), JULY(6), AUGUST(7), SEPTEMBER(8), OCTOBER(9), NOVEMBER(10), DECEMBER(11)

しかし、以下の場合は注意が必要です。

【上記サンプルコードから10ヶ月を加算した日付を取得】

```
import java.util.Calendar;

public class CalendarMain {

    public static void main(String[] args) {
        // 今日(2019/03/06)から10ヶ月後の年月日を表示
        Calendar cal = Calendar.getInstance();

        System.out.println(cal.get(Calendar.YEAR));
        System.out.println(cal.get(Calendar.MONTH) + 1 + 10);  // ←getメソッド使用時に10ヶ月を加算
        System.out.println(cal.get(Calendar.DATE));
    }
}
```

【実行結果】

```
2019  // ←繰り上がらない
13 // ←月の値がおかしい
6
```

月の値が13月という不可解な値になってしまっています。
あくまで月の値 + 11であるため、年の加算減算はしてくれません。
では、年や月の繰り上げや繰り下げの対応をしたい場合はどのような方法を取るべきかというと、
下記のようにaddメソッドを使用します。

```
cal.add(年月日, 調整したい値（マイナス値も可）)
```

【addメソッドを使用し、10ヶ月を加算した日付を取得】

```
import java.util.Calendar;

public class CalendarMain {

    public static void main(String[] args) {
        Calendar cal = Calendar.getInstance();

        // addメソッドを使用して+10する
        cal.add(Calendar.MONTH, 10);

        // 今日(2019/03/06)から10か月後の年月日を表示
        System.out.println(cal.get(Calendar.YEAR));
        System.out.println(cal.get(Calendar.MONTH) + 1);
        System.out.println(cal.get(Calendar.DATE));
    }
}
```

【実行結果】

```
2020
1
6
```

上記のようにすることで、正常な値を設定可能です。

補足

・うるう年
上記は正しい書き方にはなりますが、例外もあります。
それが「 **うるう年** 」の場合です。

【条件】

- 西暦年号が4で割り切れる年
- ただし西暦年号が100で割り切れて400で割り切れない年は平年(100で割り切れない場合と、100で割り切れても400で割れる場合はうるう年)

```
/**
 * うるう年判定
 * @param year
 */
public static void printLeapYear(final int year) {
    boolean isLeapYear = false;
    if (year % 4 == 0) {
        if ((year % 100) == 0) {
            isLeapYear = ((year % 400) == 0);
        } else {
            isLeapYear = true;
        }
    }
    System.out.println(year + "年はうるう年" + (isLeapYear ? "です。" : "ではありません。"));
}
```

カレンダーの設定が合っているのにもかかわらず想定している日付とならない場合は、
うるう年のチェックをしてみるといいでしょう！

Step3: Timeクラス

JDK1.8からTimeクラスが追加されました。
このクラスはDateクラスとCalendarクラスの両方の特徴があり、
データ保持と日付操作がTimeクラスひとつで行うことができます。

Timeクラスには、LocalDateとLocaltimeのインスタンスを持つLocalDateTime、LocalDateTimeに加え
オフセット（標準時との時差）を含むOffsetDateTime、OffsetDateTimeに加えタイムゾーンを含む
ZonedDateTimeがあります。

例

```
import java.time.LocalDateTime;
import java.time.OffsetDateTime;
import java.time.ZonedDateTime;

public class TimeMain {

    public static void main(String[] args) {
        LocalDateTime ldtNow = LocalDateTime.now();
        System.out.println(ldtNow.toString());

        OffsetDateTime odtNow = OffsetDateTime.now();
        System.out.println(odtNow.toString());

        ZonedDateTime zdtNow = ZonedDateTime.now();
        System.out.println(zdtNow.toString());
    }
}
```

【実行結果】

```
2019-03-06T13:49:52.685599
2019-03-06T13:49:52.686943+09:00
2019-03-06T13:49:52.690983+09:00[Asia/Tokyo]
```

Timeクラスは Dateクラスや Calendarクラスと異なり、直接、値を変更することはできません。
日時などの値を変更する場合はインスタンスを生成しなおす必要があります。

また、月の値の対応も Calendarクラスと異なっており、月が1の場合、そのまま1月として出力されます。

```
import java.time.LocalDateTime;

public class LocalDateWithMonth {

    public static void main(String[] args) {
        LocalDateTime ldt = LocalDateTime.now();
        System.out.println(ldt.getMonthValue() + "月");

        ldt = ldt.withMonth(12);
        System.out.println(ldt.getMonthValue() + "月");
    }
}
```

【実行結果】

```
3月
12月
```

補足

・ isLeap() **メソッド**
前述でうるう年の説明がありましたが、
Timeクラスには Year クラスというものが存在し、その Year クラスでは、
うるう年を判定する staticメソッドが用意されています。

【サンプル: うるう年判定】

```
// 変数 year が 2020の場合
System.out.println(year + "年は、うるう年" + (Year.isLeap(year) ? "です。" : "ではありません。"));
```

【出力結果】

```
2020年は、うるう年です。
```

とても簡単に使用できますね。
イチからロジックを考えることは非常に大切ですが、
開発者に優しいJava標準のAPIが実装されているならば、それを使わない手はありません！

参考

Calendarクラスのサマリー

上記リンクはオラクル公式のドキュメントで、Calendarクラスに関連するフィールドやメソッドの仕様が記載されています。
Wikiに記載されていない内容で確認したい、使ってみたいものがあった場合に一度目を通すといいでしょう。

課題

インポートした3-11のフォルダの中にプロジェクトがありますので、
指示通りにコーディングして、ファイルを提出して下さい。

評定概要

学生から秘匿	No
参加者	75
提出	50
要評定	1