SEMAINE 3 : TP2 - PROGRAMMER DE NOUVEAUX TYPES

Dans ce TP nous traitons de la manipulation des dates en Java :

- classes `Calendar` et `GregorianCalendar` (*classe* `Date`)
- définition d'une classe simplifiée représentant une date : `EasyDate`,
- classes de mise en forme des dates `SimpleDateFormat`.

GREGORIANCALENDAR

Historiquement c'est la classe `java.util.Date` qui permettait de représenter une date.
La plupart de ses méthodes de cette classe sont désormais dépréciées *(i.e. il est conseillé de ne plus les utiliser car elles sont appelées à disparaître)*.
Consultez la javadoc concernant cette classe Date.

On utilise de préférence la classe `java.util.GregorianCalendar`.

Consultez la javadoc concernant cette classe GregorianCalendar ainsi que la classe Calendar.

La méthode `toString()` fournie par cette classe retourne toutes les informations comme suit :

```
java.util.GregorianCalendar[time=1417418312906,areFieldsSet=true,areAllFieldsSet=true,lenient=true,zone=sun.util.calendar
.ZoneInfo[id="Europe/Paris",offset=3600000,dstSavings=3600000,useDaylight=true,transitions=184,lastRule=java.util.SimpleT
imeZone[id=Europe/Paris,offset=3600000,dstSavings=3600000,useDaylight=true,startYear=0,startMode=2,startMonth=2,startDay=
-1,startDayOfWeek=1,startTime=3600000,startTimeMode=2,endMode=2,endMonth=9,endDay=-
1,endDayOfWeek=1,endTime=3600000,endTimeMode=2]],firstDayOfWeek=2,minimalDaysInFirstWeek=4,ERA=1,YEAR=2014,MONTH=11,WEEK_
OF_YEAR=49,WEEK_OF_MONTH=1,DAY_OF_MONTH=1,DAY_OF_YEAR=335,DAY_OF_WEEK=2,DAY_OF_WEEK_IN_MONTH=1,AM_PM=0,HOUR=8,HOUR_OF_DAY
=8,MINUTE=18,SECOND=32,MILLISECOND=906,ZONE_OFFSET=3600000,DST_OFFSET=0]
```

Pour un affichage, en général, il est nécessaire d'extraire les valeurs des champs que l'on souhaite afficher.

Ces valeurs sont obtenues grâce :

- à la méthode `get()` à laquelle on transmet le n° du champ à afficher,
  - pour fournir ce n° on utilisera de préférence les **constantes de classe** *(`static`)* définies dans la classe `Calendar`.

Exemple de création puis affichage de la date du jour sous la forme jj/mm/aaaa :

```
GregorianCalendar date=new GregorianCalendar();

System.out.println(
      date.get( Calendar.DAY_OF_MONTH )+
      "/"+date.get( Calendar.MONTH ) +
      "/"+date.get( Calendar.YEAR )
);
```

Exercice :

Créer un classe `EssaiCalendar` dont la méthode `main()` :

1. Génère et affiche la date courante,
2. Génère aléatoirement et affiche 2 dates aléatoirement, une en décembre 1990, une autre en janvier 2010,
3. Génère aléatoirement et affiche deux dates de l'année courante,
4. Indique laquelle de ces deux dates précède l'autre,
5. Affiche le nombre de jours restant avant la fin de l'année (*tenir compte des années bissextiles*).

EASYDATE

Nous allons définir une classe `EasyDate` réutilisant la classe `GregorianCalendar` mais en en simplifiant l'usage.

Créer une classe `EasyDate,` possédant un unique attribut de type `GregorianCalendar,` qui propose les méthodes suivantes :

1. **Une méthode** `toString()` redéfinie affichant la date sous la forme : `jj/mm/aaaa`,
2. **Un constructeur sans paramètre**, créant une `EasyDate` avec la date courante,
3. **Un constructeur avec 2 paramètres** créant une `EasyDate` dans l'année courante avec le jour et le mois spécifié,
4. **Un constructeur avec 3 paramètres** créant une `EasyDate` avec les jour, mois, année (`aaaa`) spécifiés,
5. **Un constructeur acceptant une chaîne de caractères** représentant la date au format `jj/mm/aaaa`
   - en cas de format incorrect :
     - un message est affiché, et la `EasyDate` créée ne contient pas de date,
     - Modifier la méthode `toString()` pour que la chaîne retournée par la méthode `toString()` pour ces `EasyDate` non renseignée est "Date incorrecte"
6. Compléter en écrivant les méthodes `equals()` et `compareTo()`

Ecrire une classe `EssaiEasyDate` :

1. réalisant le même scénario que `EssaiCalendar` mais en s'appuyant sur la classe `EasyDate`.
2. Créant un tableau ordonné de 10 dates générées aléatoirement

POUR ALLER PLUS LOIN

SIMPLEDATEFORMAT

La classe **`SimpleDateFormat`** permet de formater et d'analyser une date en tenant compte d'une Locale.

Pour réaliser ces traitements, cette classe utilise un modèle (pattern) sous la forme d'une chaîne de caractères.

La classe propose plusieurs méthodes pour obtenir le modèle par défaut de la Locale courante :

- **`getTimeInstance(style)`**
- **`getDateInstance(style)`**
- **`getDateTimeInstance(styleDate, styleHeure)`**

Ces méthodes utilisent la **`Locale`** par défaut mais chacune de ces méthodes possède une surcharge qui permet de préciser une Locale.

De plus pour chacune de ces méthodes, quatre styles sont utilisables :

- **`SHORT`**, **`MEDIUM`**, **`LONG`** et **`FULL`**.

Ecrire une classe `EssaiSimpleDateFormat` produisant l'affichage suivant (*pour la date du 7/7/2015*) :

```
// En définissant le pattern :
        Format par défaut : 07/07/15 09:30

        2015/07/07

        07-7-2015 09:30:07

        07 juillet 2015 Heure d'été d'Europe centrale

        July 07 09:30:07 Central European Summer Time 2015

        mar., 07 juil. 2015 09:30:07 CEST

// En utilisant les styles prédéfinis :
         1: 09:43

         2: 09:43:46

         3: 09:43:46 CEST

         4: 07/07/15 09:43

         5: 7 juil. 2015 09:43

         6: 7 juillet 2015 09:43:46 CEST
```