Projet encadré Java
Time, Date et Cie

Carl Esswein

V2.1
Dernière mise à jour : mars 2018.

Table des matières

1. Préambule

Dans le cadre du projet encadré java, ce document se veut être un *mini guide de prise en main rapide* des éléments de l'API Java pour la gestion élémentaire des dates et du temps. Il s'agit d'un recueil de petits exercices et d'une liste de points d'entrée intéressants sur le sujet.
Les exercices proposés sont à faire dans un projet (au sens Eclipse) séparé, puis les notions sont à appliquer, le cas échéant, au projet encadré.

2. En route

Depuis java 8, le package de gestion des dates et du temps est `java.time`[1].

Jusqu'à Java 7, les classes appropriées – encore disponibles aujourd'hui mais obsolètes – étaient dans `java.util` (`Date` et `Calendar` par exemple) ou bien dans `java.text` (`SimpleDateFormat`) !! Et les choses n'étaient pas particulièrement simples…
Cf. Annexe pour plus de détail si vous êtes confrontés à un projet java 7 ou antérieur.

La racine de ce package fournit les classes permettant de gérer une date (par exemple le 21 mars 2016) et/ou une heure (par exemple 16h26 et quelques secondes) : `LocalDate`, `LocalTime` et `LocalDateTime`. Commençons par regarder comment elles fonctionnent.

Comme la plupart des classes de ce package et de ses sous-packages, elles présentent plusieurs caractéristiques notables à commencer par les suivantes :

- les objets associés sont **immuables**, donc la classe ne fournit pas de setter,
- ils sont thread-safe,
- **aucun constructeur** n'est défini. Pour instancier ces classes il convient donc de passer par les *factory* (méthodes préfixées par `of`, `from` ou `parse`) ou par des méthodes de copie ou de conversion. (*Cf.* http://docs.oracle.com/javase/tutorial/datetime/overview/naming.html)

Vous trouverez au paragraphe 3 (p.3) un lien vers le tuto Oracle vous fournissant les billes pour réaliser les petits exercices suivants.

1. Dans une méthode de test, créez successivement :
   1.1. Un objet `date1` représentant la date en cours,
   1.2. Un objet `date2` représentant le jour se trouvant dans 26 semaines et 19 jours.
2. Affichez les deux objets sur la sortie standard puis pour chacun des deux, affichez le jour de la semaine
   2.1. en anglais,
   2.2. en français.

Peut-être faut-il passer par `DayOfWeek.getDisplayName(…)` ?

3. Créer maintenant un nouvel objet, `date3`, représentant votre date de naissance,
   3.1. et affichez le nombre de mois nous séparant de `date3`.
   3.2. Sans utiliser les méthodes `of(…)` ni `minusXXX(…)`, créez une copie de `date3`, disons `date4`, vous faisant naître le même jour mais en janvier.

4. En utilisant le package `java.time.format`, affichez maintenant sur la sortie standard les objets `date1`, `date3` et `date4` de la façon suivante :

```
Nous sommes le lundi 21 mars 2016.
Je suis né le mardi <même format>
Je suis quasiment né le <même format>
```

[1] *http://docs.oracle.com/javase/8/docs/api/java/time/package-summary.html*

Symétriquement au formatage *date➔texte, les formatters vous permettent également de construire des instances de* `LocalDate` *(etc.) en parsant une* `String`*.*
Cf. méthodes `parse()` *…*

La manipulation des heures fonctionne sur le même principe :

5. Dans une méthode de test, créez successivement :
   5.1. Un objet `time1` représentant l'heure en cours,
   5.2. Un objet `time2` représentant l'heure qu'il était il y a 19749 minutes.

Et il est bien sûr possible de combiner date et heure dans un même objet :

6. Dans une méthode de test :
   6.1. Créez une instance `ldt1` de `LocalDateTime` à partir de la date et de l'heure courante.
   6.2. Créez une instance `ldt2` de votre date de naissance horodatée, en imaginant que vous avez émergé à 5h40 (partez de `date3`, *cf.* Q3).
   6.3. Affichez, en heures, la durée de votre vie jusqu'à `ldt1`.

Très bien. A ce stade, vous devriez être capables de manipuler aisément des instances de dates et/ou d'heures, d'en créer des copies modifiées, de les convertir et de les afficher. Cela devrait être suffisant pour l'utilisation que vous allez en faire dans le cadre du projet encadré java.
Pour plus de détails, voir les références du paragraphe ci-dessous.

3. Quelques références web

Avec Java SE 8 :

- La meilleure source d'information qui soit :
  http://docs.oracle.com/javase/8/docs/api/java/time/package-summary.html
- Un tutoriel officiel :
  http://docs.oracle.com/javase/tutorial/datetime/index.html

Annexe

Mais comment donc faisait-on avant java 8 ?! La réponse ici-même.

Jusqu'à Java SE 7, les 2 classes les plus importantes pour manipuler des heures/dates étaient `java.util.Date` et `java.util.GregorianCalendar` (qui hérite de la classe abstraite `java.util.Calendar`).

`Date` et `GregorianCalendar` disposent chacune d'un constructeur sans paramètre qui initialise la date et l'heure à l'heure courante.

1. Essayez par exemple, dans un projet « test », d'exécuter le code suivant :

```
public static void main(String[] args){
  Date d = new Date();
  System.out.println("date : "+d);
  GregorianCalendar gcal = new GregorianCalendar() ;
  System.out.println(gcal);
}
```

2. Essayez maintenant d'accéder, pour chacun des deux objets créés, à la valeur du mois qu'il contient (par exemple « AVRIL ») puis affichez la.
   *!! Attention à ne pas utiliser les méthodes dépréciées (*deprecated*) ! Ce sont des méthodes obsolètes, 100% interdites !*

Il est fort possible que la méthode « `int get(int field)` » de `Calendar`, utilisée conjointement avec les constantes de `Calendar`, vous soit d'une aide certaine.

3. Ecrire maintenant un programme java affichant sur la sortie standard la date et l'heure courante sous la forme suivante :

```
Nous sommes le lundi 13 avril 2015, il est 19h20 et 40 secondes.
```

4. Quand vous aurez bien galéré, essayez de faire le programme de la question précédente en utilisant la classe `java.text.SimpleDateFormat`.

Au besoin, faites le tutoriel de Lars Vogel (*Cf.* référence ci-après).

5. Ecrire maintenant un programme java qui demande à l'utilisateur de fournir deux dates (au format JJ/MM/AAAA) via l'entrée standard, et qui affiche la différence entre ces deux dates sous la forme suivante :

```
Vos dates sont distantes de 3 années, 4 mois et 5 jours.
```

Pour la gestion de l'entrée standard, utilisez `java.util.Scanner` et `System.in`.

Références

Avec Java SE 7 :

- Un petit tutoriel très abordable que vous pourriez vouloir faire pour commencer : http://www.vogella.com/tutorials/JavaDateTimeAPI/article.html