Annexe 9 - Notes sur les classes LocalDate, LocalTime LocalDateTime, YearMonth

## Les trois classes :

* encapsule en un objet les différents champs ( jour, mois, année, heure… ) relatifs à une date
* font partie du package java.time
* utilise plusieurs méthodes statiques, entre autres pour créer des objets représentant ces dates / heures.

## Différence :

 **LocalDate** / **LocalTime** / **LocalDateTime** / **YearMonth** encapsulent, sous forme d’objets immuables, les différents champs (jour, mois, année, heure, minute, seconde…) relatifs à une date ou une heure.

 Elles font toutes partie du package java.time.

 Elles proposent de nombreuses méthodes statiques pour créer, modifier ou comparer des instances.

**Différences principales**

| **Classe** | **Contenu** |
| --- | --- |
| **LocalDate** | année, mois, jour (sans heure) |
| **LocalTime** | heure, minute, seconde (sans date) |
| **LocalDateTime** | date + heure (année, mois, jour, heure, minute, seconde) |
| **YearMonth** | année + mois (sans jour ni heure) |

## Classe LocalDateTime

* Créer un objet contenant la date et l’heure actuelle du système:

LocalDateTime present = LocalDateTime.now();

## Classe LocalDate

* Créer un objet contenant une date quelconque :

LocalDate noel = LocalDate.of (2024,12,25);

* représente le 25 décembre 2024
* Pour aller chercher un attribut / variable de l’objet : différentes méthodes get

noel.getDayOfMonth() 🡪 = 25

if ( present.getMonth()== Month.*OCTOBER*) // getMonth retourne un enum ( JANUARY, FEBRUARY … )

ou

if ( present.getMonth().getValue()) == 10

* Pour créer un nouvel objet en modifiant un ancien objet : méthodes withDayOfYear, withMonth, withYear, withHour

LocalDateTime futur = present.withYear(2027);  
System.*out*.println(futur.getYear());

* PLUS UTILE : Pour augmenter ou diminuer la valeur d’un attribut, méthode plus ou minus, retourne un nouvel objet

LocalDateTime futurProche = present.plusDays(2) 🡪 on est rendus 2 jours plus tard

* Pour savoir si un objet LocalDateTime est chronologiquement avant un autre : méthode isBefore

present.isBefore(futur) 🡪 retourne true

## Classe DateTimeFormatter,

* utile lorsqu’on veut afficher des dates contenues dans un objet LocalDate ou LocalDateTime
* fonctionne en créant un modèle qui sera passé dans l’appel de la méthode ofPattern du DateTimeFormatter
* Les caractères principaux utilisés dans le modèle sont :

|  |  |  |
| --- | --- | --- |
| Caractères | Correspondant à… | Exemple |
| yyyy | Année | 1989 |
| MMM ( 3 ou plus ) | Mois dans l’année ( en lettres ) | Juillet |
| MM ( 2 caractères ) | Mois dans l’année ( en chiffres ) | 07 |
| dd | Jour dans le mois | 22 |
| HH | Heure dans le jour ( 00 – 23 ) | 15 |
| mm | Minutes | 56 |
| ss | Secondes | 43 |

* il existe d’autres variantes : [voir l’API](https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html)

Exemple :

LocalDate date = LocalDate.now();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MMMM/yyyy");

String text = date.format(formatter);

* 3/octobre/2024 ( par exemple )

**Modifier une partie de l’objet (immutabilité) :**

**LocalDateTime futur = present.withYear(2027);**

**LocalDateTime deuxJoursPlusTard = present.plusDays(2);**

**Comparer deux dates :**

**boolean avant = present.isBefore(futur);**

**Classe YearMonth**

* **Créer un objet pour un mois et une année :**
  + **YearMonth ym = YearMonth.of(2025, Month.MAY); // Mai 2025**
* **Récupérer l’année ou le mois :**
  + **int annee = ym.getYear(); // 2025**
  + **int mois = ym.getMonthValue();**
* **Combiner avec un jour pour obtenir une LocalDate :**
  + **LocalDate premierMai = ym.atDay(1); // 2025-05-01**