

IAP : TP8

BUT/INFO

version 2021-2022

Table des matières

- [1. Des fonctions pour JourSuivant](#)
- [2. Réflexion sur le stockage des informations](#)
- [3. Préambule à propos des commentaires](#)
- [4. Saisie d'une date](#)
- [5. Validité d'une date](#)
- [6. Calcul du jour suivant](#)
- [7. Verdict](#)
- [8. Surlendemain](#)
- [9. Aller plus loin](#)
 - [9.1. Sursursur...surlendemain](#)
 - [9.2. Barycentre d'anniversaire](#)
- [10. Avant de partir](#)

1. Des fonctions pour JourSuivant

L'algorithme du programme **JourSuivant.java** peut s'écrire, au premier niveau de raffinement, ainsi :

```
DEBUT
    * -- Étape 1 -- *
    * Saisie d'une date *

    * -- Étape 2 -- *
    * La date saisie est-elle valide ? *

    * -- Étape 3 -- *
    SI dateValide ALORS
        * Calcul du jour suivant *
        * Affichage du jour suivant *
    SINON
        * Affichage du message d'erreur : date invalide *
    FINSI

FIN
```



Identification des sous-problèmes

La lecture de cet algorithme montre que nous pouvons décomposer la résolution de notre problème en la confection de 3 sous-algorithmes :

- saisie d'une date par l'utilisateur au clavier,
- vérification de la validité d'une date,
- calcul du jour suivant à partir d'une date valide.

2. Réflexion sur le stockage des informations

Pour la première fois, la manière dont on va stocker les informations manipulées par nos programmes n'apparaît pas de façon évidente. Ici, on doit décider sous quelle forme on va stocker une date quelconque. La date est en effet une entité un peu spéciale, puisqu'elle est caractérisée par un jour, un mois et une année.

On pourrait opérer de la même manière que dans le TP2, par exemple, où l'on manipulait les dates au moyen de trois variables indépendantes, toutes de type entier, la première représentant le jour, la seconde le mois et la troisième l'année. Tout se passait alors très bien : lorsque l'on voulait écrire un sous-programme qui avait une date en entrée, on passait simplement trois paramètres (un jour, un mois et une année). Mais tout se passait bien, parce que nous n'avions pas eu besoin d'écrire un sous-programme qui renvoyait une date. Que faire dans ce cas-là ? Ça n'est pas immédiat : on se souvient qu'on ne peut en effet renvoyer qu'une seule valeur de retour, et ici, on voudrait en renvoyer trois (le jour, le mois et l'année).

On verra bientôt en SDA (Structures de Données et Algorithmes, module suivant immédiatement IAP) comment régler ce genre de problème de manière pleinement satisfaisante. Pour l'instant, nous utiliserons la structure de donnée du pauvre : le tableau.

Dans ce TP, nous prenons la convention de stocker chaque date dans un tableau. Plus précisément, une date sera un tableau d'entiers à trois cases, dont la première sera le jour de cette date, la seconde le mois et la troisième l'année.

3. Préambule à propos des commentaires

Nous vous demandons désormais d'être strict-e-s quant à l'écriture de vos sous-programmes (fonctions ou procédures) java, et vous proposons d'adopter, dans un but de compréhension mutuelle (chaque personne comprend facilement le code d'une autre personne), les conventions suivantes.

- Chaque sous-programme est précédé d'un commentaire javadoc,
- chaque paramètre est décrit par son nom, sa caractéristique d'entrée/sortie (IN pour un paramètre d'entrée, OUT pour un paramètre de sortie, IN/OUT pour un paramètre d'entrée/sortie) et un commentaire sur son rôle,
- la valeur de retour est décrite par sa signification.

Lancer BlueJ, créer et nommer (nous vous suggérons TP8_NumeroDeGroupe_Nom_Prenom) un nouveau projet où vous souhaitez placer vos sources java.

Créer la nouvelle classe suivante :

Exemple 1. JourSuivant.java

```
import java.util.Scanner;

public class JourSuivant {

    /**
     * Fait saisir une date à l'utilisateur
     *
     * @param pfDate OUT : un tableau de trois cases
     représentant une
     * date. 1ere case : jour, 2nde case : mois,
     3eme case : annee
     */
    public static void saisieDate(int[] pfDate) {
        if (pfDate.length != 3) {
            System.out.print("Le tableau
représentant la date a une taille inattendue : ");
            System.out.println(pfDate.length + "
case(s) au lieu de 3 !");
        }
        Scanner clavier = new Scanner(System.in);
        // A vous
    }

    /**
     * Calcul la validité d'une date
     *
     * @param pfDate IN : date initiale
     @return true si et seulement si pfDate est
     valide
     */
    public static boolean dateValide(int[] pfDate) {
        if (pfDate.length != 3) {
            System.out.print("Un tableau
représentant une date a une taille inattendue : ");
            System.out.println(pfDate + " case(s) au
lieu de 3 !");
        }

        // A modifier
        return true ;
    }

    /**
     * Calcul du jour suivant
     *
     * @param pfDateJourCourant IN : date initiale
     * @param pfDateJourSuivant OUT : date du jour
     suivant
     */
    public static void jourSuivant(int[]
pfDateJourCourant, int[] pfDateJourSuivant) {
        if (pfDateJourCourant.length != 3 ||
pfDateJourSuivant.length != 3) {
            System.out.print("Un tableau
représentant une date a une taille inattendue : ");

            System.out.println(pfDateJourCourant.length + " ou "
+ pfDateJourSuivant.length + " case(s) au lieu
de 3 !");
        }
        // A vous
    }

    public static void principale() {

        /* Déclaration des variables */
        int[] date = new int[3] ;
        boolean valide = false ;

        /* -- Etape 1 -- */
        /* Saisie d'une date */
        // A vous

        /* -- Etape 2 -- */
        /* Vérification de la date saisie */
        // A vous

        /* -- Etape 3 -- */
        if (valide) {
            /* Calcul du jour suivant */
            /* Affichage du jour suivant */
            // A vous
        } else {
            System.out.println("La date du "
+ date[0] + "/" +
date[1] + "/" + date[2]
+ " n'est pas une
date valide.");
        }
    }
}
```

Vous constaterez l'utilisation des commentaires javadoc en début de chaque fonction.

4. Saisie d'une date

Nous vous rappelons que le comportement des tableaux est différent de celui des entiers ou des nombres à virgule lorsqu'ils sont passés en paramètre d'une fonction (ou d'une procédure) : si cette fonction modifie la valeur des cases du tableau, les cases seront vues comme modifiées à l'extérieur de la fonction.

Compléter la fonction `saisieDate()`.

Compléter également le code de la fonction `principale()`.

5. Validité d'une date

Comme vu dans le squelette de la classe, on a décidé de coder le sous-algorithme capable de *vérifier qu'une date est valide* sous la forme d'une fonction java.

Il a donc fallu déterminer la **signature** de la fonction : quels paramètres lui sont fournis et quel type de résultat fournit-elle ?

La **signature** retenue est la suivante :

```
boolean dateValide(int[] pfDate)
```



La **signature** précédente indique clairement que la fonction java **dateValide()**, prend un tableau en paramètre et retourne un booléen indiquant si et seulement s'il est vrai (**true**) que la date fournie est valide.

Coder la fonction **dateValide()**, compiler, tester ...

Compléter également le code de la fonction `principale()`.


6. Calcul du jour suivant

Coder le sous-algorithme capable de *calculer le jour suivant à partir d'une date valide* sous la forme d'une fonction java.

La **signature** retenue :

```
void jourSuivant(int[] pfDateJourCourant, int[]
pfDateJourSuivant)
```

indique clairement que la fonction java **jourSuivant()**, prend 2 tableaux en paramètres et ne retourne rien.



Notre convention (IN ou OUT ou IN/OUT) nous indique que le premier tableau pourra être lu, mais pas modifié, tandis que la fonction (au sens strict, la procédure) pourra écrire dans les cases du second tableau mais qu'avant cela, il ne faudra pas espérer y lire quelque chose d'intéressant.

Coder la fonction **jourSuivant()**, compiler, tester ...

Compléter également le code de la fonction `principale()`.

7. Verdict

Évidemment, vous aviez préalablement décrit un jeu d'essais sur lequel tester votre fonction `principale()`.

Vérifier que toutes les entrées du jeu d'essai

8. Surlendemain

Ajouter une fonction `surlendemain()` qui calcule le surlendemain d'une date donnée, ainsi qu'un appel à cette fonction depuis la fonction `principale()`.

Attention à ne pas dupliquer inutilement du code.

9. Aller plus loin

9.1. Sursursur...surlendemain

Pourquoi se limiter au surlendemain, finalement ? Écrire un algorithme qui calcule la date d'un jour situé à un nombre arbitraire de jours dans le futur d'une date donnée.

9.2. Barycentre d'anniversaire

9.2.1. Une idée peu répandue ...

Micheline et Jean-Eude aimeraient fêter leurs 80 ans ensemble, mais ne sont malheureusement pas nés le même jour. Pour éviter tout favoritisme, la fête se tiendra à la (l'une des deux) date(s) située au milieu entre leurs dates d'anniversaire.

Écrire un algorithme qui calcule cette date en fonction des deux dates de naissance. On supposera que ni l'une ni l'autre ne sont nés un 29 février.

9.2.2. ... mais qui prend de l'ampleur

Valérie adore l'idée, et déclare vouloir participer. Puis Georges. Puis Dominique.

Écrire un algorithme qui décide de la date d'anniversaire commune que pourront choisir un nombre quelconque de personnes pour fêter leurs 80 ans tous ensemble.

10. Avant de partir

- Enregistrer vos programmes sur **webetud2**
- N'oubliez-pas de vous déconnecter