TP 5 - Méthodes avec résultats (classe `Personne`)

En TD, nous avons passé plusieurs séances sur la classe `Personne`. Celle-ci a servi de support à l'introduction des concepts de *type*, *encapsulation*, *constructeurs*, *méthodes avec et sans valeur de retour*, et également d'*aggrégation*.

À retenir

L'aggrégation réfère au fait de définir des classes dont certains attributs sont d'un type non-atomique (c'est-à-dire des attributs qui sont eux-même des objets).

Dans notre cas, l'aggrégation correspond au fait que la classe `Personne` utilise les classes `Adresse` et `MaDate`. L'aggrégation offre un moyen de spécifier un type par réutilisation de types existants.

Dans ce TP, nous allons revenir sur les accès `private` et `public` et les méthodes avec résultats.

1 Droit d'accès

Soit la classe `MaDate` suivante déposée sur arche.

```
public class MaDate {

  /**
   * attributs MaDate
   */
  public int jour;
  public int mois;
  public int annee;

  /**
   * constructeur MaDate
   *
   * @param j jour de la date (entre 1 et 31)
   * @param m mois de la date (entre 1 et 12)
   * @param a annee de la date
   */
  public MaDate(int j, int m, int a) {

    this.jour = j;
    // verifie que le jour est correct
    if (this.jour < 1)
```

```
      this.jour = 1;
    if (this.jour > 31)
      this.jour = 31;

    this.mois = m;
    // verifie mois correct
    if (this.mois < 1)
      this.mois = 1;
    if (this.mois > 12)
      this.mois = 12;

    this.annee= a;
  }

  /**
   * constructeur par copie
   */
  public MaDate(MaDate d) {
    if (d != null) {
      this.jour = d.jour;
      this.mois = d.mois;
      this.annee = d.annee;
    } else {
      // d inexistant => date par defaut
      this.jour = 1;
      this.mois = 1;
      this.annee = 1970;
    }
  }

}
```

Soit la classe `ProgDate` suivante en charge de créer et d'afficher des dates.

```
class ProgDate {

  public static void main(String[] args) {
     MaDate d = new MaDate(-1,-1,2000);
     d.afficher();

     // force une valeur pour l'attribut jour
     d.jour = -1;
     d.afficher();
  }
}
```

Question 1

Écrire la méthode `afficher` de la classe `MaDate` devant afficher une date. Vérifiez

que la date affichée dans `ProgDate` est bien correcte avant la modification brutale de l'attribut et incorrecte aprés cette modification.

Question 2

Transformer les attributs `public` de `MaDate` en attributs `private`. Recompiler les classes. Prenez le temps de lire l'erreur obtenue et de la comprendre. Quel est le problème à la compilation ? (ligne et erreur)

Question 3

Ajouter de getter et de setter dans la class `MaDate` pour modifier l'attribut `jour`. Penser à vérifier que le jour est bien correct (on supposera que les jours < 31 sans vérifier les mois).

Question 4

Modifier `ProgDate` pour qu'il compile en conservant les attributs privés. Vérifier qu'il n'est plus possible de créer un objet avec un jour négatif.

Afin d'avoir une méthode complète, on souhaite pouvoir avoir accès au nombre de jours d'un mois donné. Pour rappel, février possède 29 jours pour une année bissextile et une année est bissextile si c'est un multiple de 400 ou si c'est un multiple de 4 qui n'est pas un multiple de 100.

Par exemple, 2000 est bissextile (multiple de 400), 1900 ne l'est pas (multiple de 100) et 2004 l'est (multiple de 4).

Question 5

Écrire la méthode `int getNbJours()` dans la classe `MaDate`. Cette méthode retourne le nombre de jours associé au mois et à l'année de l'objet `MaDate` qui exécute la méthode.

Question 6

Corriger la méthode `setJour` pour utiliser le nombre de jours dans les vérifications et utiliser cette méthode dans le constructeur de `MaDate` pour faire les vérifications.

Question 7

Dans `ProgDate`, créer la date `29/02/2020` et la date `29/02/2021`. Vérifier que les dates créées sont bien correctes (2021 n'est pas bissextile). Vérifier aussi que la date `32/13/2020` donne bien la date `31/12/2020`.

2 Jour suivant

On souhaite écrire la méthode `MaDate demain()` de la classe `MaDate` qui retourne le lendemain de la date sur laquelle la méthode s'applique.

Question 8

Écrire la méthode `MaDate demain()` de la classe `MaDate` qui retourne le lendemain de la date sur laquelle la méthode est appelée.

```
1  import static libtest.Lanceur.lancer;
2  import static libtest.OutilTest.*;
3  import java.awt.Color;
4
5  /**
6   * classe chargee de tester les constructeur de rectangle
7   */
8  public class TestDemain {
9
10   /**
11   * exemple de test a completer
12   */
13   public void test1_demain_normal(){
14     // donnees a tester
15     MaDate d = new MaDate(1,1,2000);
16     MaDate res = d.demain();
17
18     // verification (necessite de getters)
19     assertEquals("jour devrait avancer", 2, res.getJour());
20     assertEquals("mois devrait etre le meme", 1, res.getMois());
21     assertEquals("annee devrait etre la meme", 2000, res.getAnnee());
22   }
23
24   // ############################
25   // TODO : autres tests a ecrire ....
26   // ############################
27
28   /**
29    * lancement des tests
30    */
31   public static void main(String args[])
32   {
33     lancer(new TestDemain(),args);
34   }
35
36 }
```

Question 9

Compléter la classe de test `TestDemain.java` fournie sur arche pour vérifier que la méthode fonctionne correctement en identifiant quelques cas particuliers au préalable. N'oubliez pas d'ajouter les getter utiles pour faire les vérifications.

Question 10

Utiliser la classe `TestDemainCorrige.class` pour vérifier que votre classe `MaDate` fonctionne correctement.

3 Ville identique

3.1 `Personne` et `Adresse`

Les classes `Personne` et `Adresse` fournies ont leurs attributs déclarés en private.

3.2 Egalité de ville

On souhaite vérifier dans la classe `Adresse` si deux adresses sont situées dans la même ville.

À retenir

Pour comparer deux chaînes de caractères, on utilise la méthode `boolean equals (String s)` de la classe `String`. Cette méthode retourne le booléen `true` si et seulement si les deux chaines `this` et `s` ont des valeurs égales.

Question 11

Écrire la méthode `avoirMemeVille` dans la classe `Adresse` permettant de savoir si deux adresses sont situées dans la même ville. Prenez le temps de bien identifier la signature de la méthode demandée avant de répondre à la question.

3.3 Habiter dans la même ville

On souhaite désormais vérifier si deux personnes habitent dans la même ville.

Question 12

A l'aide de la méthode `avoirMemeVille`, écrire la méthode `habiterMemeVille` de la classe `Personne` qui permet de savoir si une personne habite dans la même ville qu'une autre personne passée en paramètre.

Question 13

Écrire un main `ProgVille` qui construit trois personnes, deux habitant dans la même ville et une troisième personne habitant dans une autre ville. Vérifier avec ce `main` que la méthode `avoirMemeVille` fonctionne correctement.

4 Comparaison (optionnel)

4.1 Comparaison de dates

On souhaite comparer des dates entre elles.

Question 14

Ajouter dans la classe `MaDate` une méthode `etreAvant` qui retourne un booléen qui vaut vrai si et seulement si, la date à qui on pose la question se trouve avant une autre date passée en paramètre. En cas d'égalité, la méthode retourne `false`.

Afin de vérifier la méthode `etreAvant`, on souhaite faire les tests suivants dans la classe `TestMaDate` partiellement fournie.

Méthode testée	**cas**	**num du test**
etreAvant	date en paramètre d'un jour avant	test 1
	date en paramètre d'un jour aprés	test 2
	date en paramètre égale	test 3
etreAvant	date décalée d'un jour aprés avec changement de mois	test 4
	date décalée d'un jour avant avec changement de mois	test 5
	date décalée d'un jour aprés avec changement d'année	test 6
	date décalée d'un jour avant avec changement d'année	test 7

Question 15

Choisir les objets pour faire les tests et écrire les tests unitaires décrits dans le tableau qui permettent de vérifier que la méthode `etreAvant` fonctionne correctement (au moins sur ces cas).

Question 16

Utiliser le fichier de `TestDateCorrige.class` fourni pour vérifier que votre classe `MaDate` fonctionne correctement.

4.2 Comparaison de `Personne`

On souhaite comparer des personnes entre elles avec la méthode `etrePlusJeune`. Cette méthode retourne un booléen qui vaut `true` si et seulement si la personne à qui on demande de se comparer est plus jeune (strictement) que la personne passée en paramètre.

Question 17

En utilisant la méthode `etreAvant`, écrire la méthode `etrePlusJeune`.

5 Calculer Age (optionnel)

Question 18

Ecrire la méthode `int calculerAge(MaDate d)` de la clase `Personne` qui retourne l'age qu'avait une personne à la date `d` donnée.

Question 19

Créer une classe de test `TestPersonneAge` qui vérifie que la méthode fonctionne correctement en identifiant quelques cas différents au préalable (personne pas encore née à la date donnée, personne qui vient juste d'avoir son age, ...).