TP 2 - Développement par les tests

L'objectif de ce TP est de poser les bases de ce qu'est la programmation orientée objet à partir de la classe `MaDate` vue en TD, tout en adoptant une méthodologie de développement s'appuyant sur les tests.

Nous allons dans un premier temps rappeler la structure de la classe `MaDate`, puis nous verrons comment utiliser un outil de tests pour vérifier que votre code est conforme à vos attentes.

1 Classe `MaDate`

1.1 Attributs

La classe (on dit aussi le type) `MaDate` a pour but de représenter une date composée d'un jour, d'un mois et d'une année (toutes trois étant des informations représentées par un nombre entier).

Question 1
Déclarer et écrire la classe `MaDate` avec ses attributs.

1.2 Constructeur vide

Dans une classe, on peut définir des constructeurs. Ceux-ci sont des méthodes particulières permettant de créer des instances de la classe (appelées aussi objets), ou dit autrement, de créer des valeurs d'un certain type.

Le but des constructeurs est double :
— réserver de la place en mémoire (dans une zone appelée tas) pour stocker les informations relatives à l'objet créé ;
— de donner des valeurs par défaut aux attributs de l'objet en question.

Une classe peut contenir plusieurs constructeurs, à conditions qu'ils n'aient pas la même signature.

Dans la classe `MaDate`, nous avons un premier constructeur vide (constructeur sans paramètre) permettant de créer une date par défaut (en l'occurrence le 1er janvier 1970).

Question 2

Ajouter un constructeur vide à la classe `MaDate` qui construit une date par défaut correspondant au 1^{er} janvier 1970.

1.3 Création d'objet

Dans un programme (c'est-à-dire dans la fonction `main` d'une classe donnée), on souhaite créer un objet de type `MaDate` par défaut.

Question 3

Écrire une classe `MainDateVide` et sa fonction `main` qui construit une date par défaut (correspondant au 1^{er} janvier 1970).

Si l'on souhaite vérifier que la date est celle attendue, on peut par exemple afficher son jour, son mois et son année.

Question 4

Affichez à l'écran les valeurs des attributs de l'objet de type `MaDate` que vous avez créé.

1.4 Constructeur avec paramètre

Nous souhaitons également un constructeur non vide permettant de créer un date quelconque (connue par un jour `j`, un mois `m` et une année `a`) :

Question 5

Écrire un constructeur qui prend trois entiers `j`, `m` et `a` en paramètre et qui construit une date correspondant à la date `j/m/a`.

Question 6

Dans une classe `MainDateParam`, écrire une fonction `main` qui crée un objet correspondant à la date du 1^er Septembre 2022 et qui affiche son jour, son mois et son année à l'écran.

2 Tests unitaires et validation

Enfin, une fois que la classe est construite, il faut valider qu'elle fonctionne correctement. C'est l'objectif des tests unitaires.

2.1 Test unitaire

Un test unitaire a pour objectif de vérifier qu'une opération se déroule bien. Chaque test correspond à une méthode constituée de deux parties :
- le morceau de programme dont on veut tester le résultat,
- les vérifications qui valideront le résultat.

2.2 Exemple du test du constructeur vide

Si l'on souhaite valider le constructeur vide, on va donc :
- tester le morceau de programme `MaDate x = new MaDate();`
- vérifier que l'objet `x` a pour jour le 1er jour du mois, pour mois le 1er mois de l'année et pour année 1970, conformément à l'énoncé.

La vérification se fait en utilisant une méthode `assertEquals` avec 3 paramètres `a`, `b` et `c` tels que :
- a désigne le résultat attendu (dans ce cas 1),
- b désigne ce qu'on vérifie (dans ce cas entre autre que `x.jour` qui doit être égal à 1),
- c désigne un message d'erreur (dans ce cas "le jour de x devrait etre 1").

Le test du constructeur vide sera donc :

```
1  /**
2   * test du constructeur vide
3   */
4  public void test1_constructeurVide()
5  {
6    //on construit la date
7    MaDate x =new MaDate();
8    //on verifie que son jour est 1
9    assertEquals("le jour de x devrait etre 1", 1, x.jour);
10   //on verifie que son mois est 1
11   assertEquals("le mois de x devrait etre 1", 1, x.mois);
12   //on verifie que son annee est 1970
13   assertEquals("l'annee de x devrait etre 1970", 1970, x.annee);
14 }
```

2.3 Exemple du test du constructeur avec paramètres

Pour les paramètres, on va tester sur un exemple. On peut prendre la date (10,2, 2020).

Le test du constructeur avec paramètres sera donc :

```
/**
 * test du constructeur avec parametres
 */
public void test2_constructeurParam()
{
  //on construit la date
  MaDate x = new MaDate(10,2,2020);
  //on verifie que son jour est 10
  assertEquals("le jour de x devrait etre 10", 10, x.jour);
  //on verifie que son mois est 2
  assertEquals("le mois de x devrait etre 2", 2, x.mois);
  //on verifie que son annee est 2020
  assertEquals("l'annee de x devrait etre 2020", 2020, x.annee);
}
```

2.4 Fichiers de test

Pour pouvoir exécuter les tests précédemment définis, nous allons écrire un programme appelé `TestMaDate` qui va utiliser une bibliothèque de tests offrant une interface graphique (bibliothèque `libtest` fournie dans l'espace du cours sur Arche).

```
import static libtest.Lanceur.lancer;
import static libtest.OutilTest.*;

/**
 * classe chargee de tester les constructeur de rectangle
 */
public class TestMaDate {
  /**
   * test du constructeur vide
   */
  public void test1_constructeurVide()
  {
    //on construit la date
    MaDate x =new MaDate();
    //on verifie que son jour est 1
    assertEquals("le jour de x devrait etre 1", 1, x.jour);
    //on verifie que son mois est 1
    assertEquals("le mois de x devrait etre 1", 1, x.mois);
    //on verifie que son annee est 1970
    assertEquals("l'annee de x devrait etre 1970", 1970, x.annee);
  }

```

```
23    /**
24     * test du constructeur avec deux parametres
25     */
26    public void test2_constructeurParam()
27    {
28      //on construit la date
29      MaDate x = new MaDate(10,2,2020);
30      //on verifie que son jour est 10
31      assertEquals("le jour de x devrait etre 10", 10, x.jour);
32      //on verifie que son mois est 2
33      assertEquals("le mois de x devrait etre 2", 2, x.mois);
34      //on verifie que son annee est 2020
35      assertEquals("l'annee de x devrait etre 2020", 2020, x.annee);
36    }
37
38    /**
39     * lancement des tests
40     */
41    public static void main(String args[])
42    {
43      lancer(new TestMaDate(),args);
44    }
45  }
```

On remarque notamment que (1) ce programme contient les deux méthodes de tests décrites précédemment, et (2) la fonction `lancer` appelée dans le `main` a pour premier paramètre la classe `TestMaDate`.

Question 7

Placez le contenu du dossier `libtest` dans votre répertoire de travail (à côté de la classe `MaDate`), et ajoutez y la classe `TestMaDate` disponible sur arche.

2.5 Lancement des tests

Pour lancer les tests, vous devez dans un premier temps compiler les classes de la bibliothèque `libtest` (opération à ne faire qu'une seule fois) :

```
1  javac libtest/*.java
```

Il est possible qu'un message d'avertissement apparaisse pour indiquer que la bibliothèque utilise des éléments du langage java dépréciés (remplacés par des versions plus récentes). Ce message est sans conséquence.

Dans un second temps, vous pouvez exécuter les tests, pour cela il vous suffit de compiler la classe de test puis de la lancer (à refaire à chaque fois que vous modifiez vos tests) :

```
javac TestMaDate.java
java TestMaDate
```

Question 8
Compiler et lancer les tests.

2.6 Résultat des tests

Une fenêtre s'ouvre alors qui rend compte des résultats des tests (cf image 1).

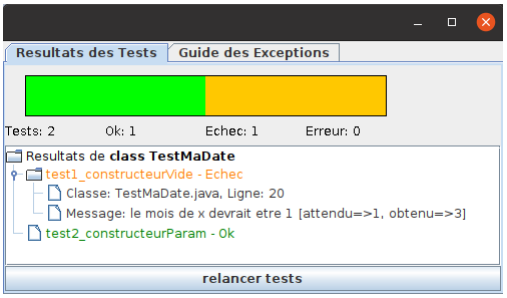

FIGURE 1 – Résultat avec un échec

Trois couleurs sont utilisés dans le rendu des tests
- le vert désigne un test réussi
- le rouge désigne un test qui conduit à une erreur (cela signifie que votre test fait "planter" votre programme - cela sera détaillé par la suite)
- l'orange désigne un test qui échoue (c'est à dire que les vérifications sont fausses)

La barre en haut du résultat donne un aperçu général. Dans ce cas, il y a deux tests. Un test a réussi mais un test a échoué. En cliquant sur l'échec, il est possible de savoir quelle vérification échoue. Ici, il s'agit du mois qui n'a pas la valeur attendue.

Nous pouvons alors corriger le programme. Aprés cette modification, tous les tests sont valides et la classe `MaDate` a bien été vérifiée (cf image 2).

2.7 Création de tests

La capacité de choisir les bons tests (cas représentatifs des différents usages d'une classe) est une qualité attendue des personnes réalisant des développements informatiques.

Dans les sujets de TP à venir, nous allons vous demander à chaque fois d'utiliser une méthodologie de développement par les tests, et nous vous fournirons des tests prédéfinis et vous demanderons d'y ajouter vos propres tests.

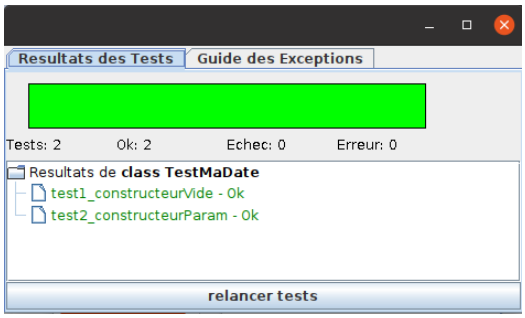

FIGURE 2 – Tous les tests sont validés

Plus précisément vous utiliserez la démarche incrémentale décrite ci-dessous.

À retenir

Chaque fois que vous écrirez une méthode :

1. Vous définirez son profil et écrirez sa javadoc.
2. Vous écrirez le corps de la méthode.
3. Vous testerez la méthode avec la classe de test fournie.
4. Vous corrigerez la méthode si elle ne passe pas le test.

Ne passez pas à la méthode suivante tant que la méthode courante n'est pas intégralement faite.

Par exemple, pour la classe `MaDate`, voici un résumé des tests à réaliser :

Méthode testée	**cas**	**numéro du test**
constructeur vide	cas normal	test 1
constructeur avec paramètres	cas normal	test 2
	jour ≤ 0	test 3
	jour > 31	test 4
	mois ≤ 0	test 5
	mois > 12	test 6

Ici, nous avons fourni les tests 1 et 2, mais pas les suivants. Imaginons que si l'on appelle un constructeur avec des paramètres erronés (par exemple un jour plus petit que 0) le constructeur doive utiliser le jour valide le plus proche.

Question 9

Écrivez et exécutez les tests 3, 4, 5 et 6. Corrigez la classe `MaDate` si besoin.

Vous pourriez par ailleurs proposer de nouveaux tests, par exemple que si le mois est 2 (février) alors le jour est inférieur ou égal à 28 (sauf en cas d'année bissextile où il doit être inférieur à 29).

Question optionelle 10
Écrivez ces deux tests additionnels.

3 Javadoc

En java, il existe un système de documentation de code intégré à l'environnement de développement : la javadoc. Cet outil permet de générer une documentation au format HTML (pages web) d'un code java à partir de commentaires présents dans le code et respectant un format particulier. La javadoc permet de commenter chaque classe et chaque attribut, constructeur et méthode.

Un commentaire (ou élément) de javadoc commence par ** et finit par *\. Chaque ligne de cet élément doit débuter par *

3.1 Commenter la classe et les attributs

La javadoc a pour objectif de décrire ce à quoi correspondent la classe et les attributs. A savoir :

```
/**
 * la classe MaDate permet de représenter des dates
 * (triplets jour - mois - annee)
 */
public class MaDate
{
  /**
   * attribut jour correspondant au jour dans le mois
   */
  int jour;

  /**
   * attribut mois correspondant au mois dans l'annee
   */
  int mois;

  /**
   * attribut annee correspondant a l'annee
   */
  int annee;

        ...
}
```

3.2 Commenter les constructeurs

Chaque constructeur est décrit par un élément javadoc. Les paramètres sont décrits avec le mot clef `@param` suivi du nom du paramètre et de son descriptif.

La classe entièrement commentée sera la suivante

```
1  /**
2   * la classe MaDate permet de représenter des dates
3   * (triplets jour - mois - annee)
4   */
5  public class MaDate
6  {
7    /**
8     * attribut jour correspondant au jour dans le mois
9     */
10   int jour;
11
12   /**
13    * attribut mois correspondant au mois dans l'annee
14    */
15   int mois;
16
17   /**
18    * attribut annee correspondant a l'annee
19    */
20   int annee;
21
22   /**
23    * constructeur par defaut
24    * construit une date correspondant au 1er janvier 1970
25    */
26   public MaDate()
27   {
28     this.jour  = 1;
29     this.mois  = 1;
30     this.annee = 1970;
31   }
32
33   /**
34    * constructeur qui construit une date
35    * quelconque
36    *
37    * @param j jour de la date construite
38    * @param m mois de la date construite
39    * @param a annee de la date construite
40    */
41   public MaDate(int j, int m, int a)
42   {
43     this.jour  = j;
44     this.mois  = m;
45     this.annee = a;
```

```
46    }
47 }
```

Enfin, pour générer votre documentation, vous devez taper dans votre interpréteur de commandes (fenêtre DOS sous windows, terminal sous linux) :

```
1 javadoc MaDate.java
```

Cela va créer un ensemble de fichiers correspondant à la page web de la figure 3.

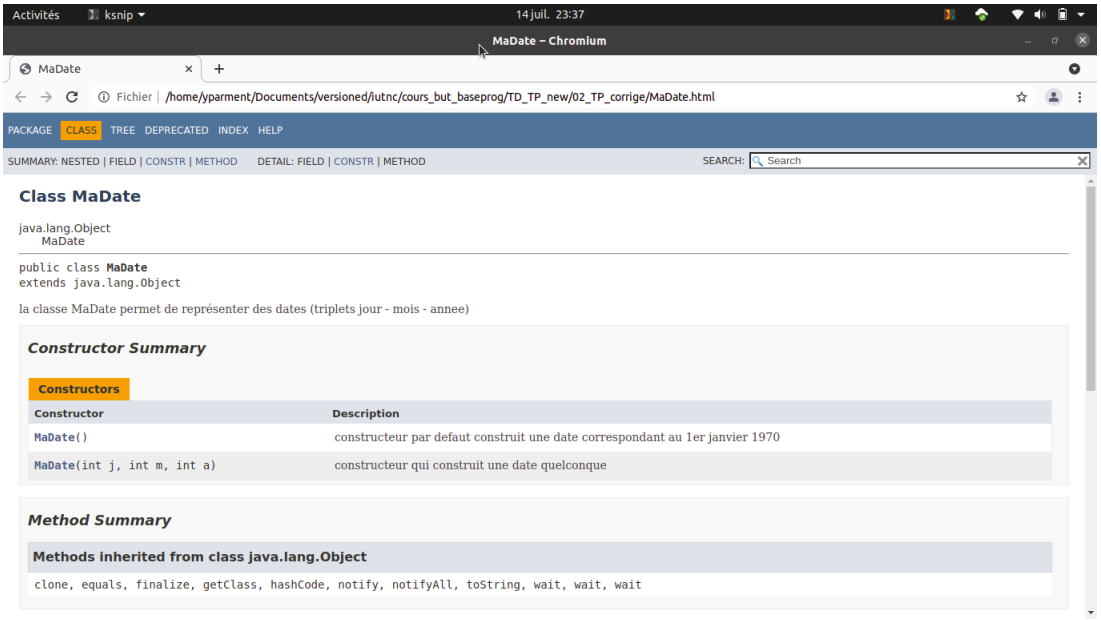

FIGURE 3 – Javadoc de la classe `MaDate`

Question 11
Générer la javadoc avec cette commande.