Outils associés à Java Modeling Language (JML)

Exercice 1 : JML et outils associés
Dans cet exercice, nous nous intéressons au programme ExempleDate1 (listing 1).

1.1 Lire le texte de la classe ExempleDate1, la compiler avec `javac` et l'exécuter avec `java`.

Listing 1 – Programme manipulant les dates

```
1  // Que penser du programme suivant ?
2  public class ExempleDate1 {
3      public static void main (String args []) {
4          // Construire les dates
5          Date d1 = new Date(20, 5, 1989);
6          Date d2 = new Date(13, 2, 1993);
7          Date d3 = new Date(31, 6, 2001);
8
9          // Afficher les dates
10         System.out.println("d1 = " + d1);
11         System.out.println("d2 = " + d2);
12         System.out.println("d3 = " + d3);
13     }
14 }
```

1.2 Compiler maintenant l'application en utilisant jmlc. Il suffit de taper :

```
jmlc Date.java ExempleDate1.java
```

1.3 Exécuter le programme en tapant :

```
jmlrac ExempleDate1
```

Commenter.

1.4 Engendrer la documentation en utilisant jmldoc. Les contrats sont pris en compte !
Remarque : Le compilateur `jmlc` instrumente les classes listées sur la ligne de commande. Instrumenter signifie ici ajouter des instructions pour vérifier les contrats exprimés dans les commentaires de comportement. Si un contrat n'est pas satisfait, une exception est levée. C'est une exception qui n'a pas à être récupérée. Elle signale une erreur de programmation.

Attention, seules les classes listées sur la ligne de commande sont instrumentées. Les autres ne le sont pas. En argument de `jmlc`, on peut également donner des répertoires. Dans ce cas, tous les fichiers `.java` de ces répertoires sont instrumentés.

Objectifs de ce sujet :
- Comprendre les outils fournis par JML ;
- Comprendre l'intérêt d'exprimer les propriétés d'un programme ;
- Utiliser les contrats comme une aide à la mise au point.