Outils associés à Java Modeling Language (JML)

Attention : Ce sujet doit se faire dans un terminal et pas sous eclipse. Il existe un greffon OpenJML pour eclipse mais il n'est pas compatible avec la version d'eclipse installée.

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. Attention, cette partie doit se faire dans un terminal !
Commençons par définir la variable d'environnement OPENJML :

```
export OPENJML=/mnt/n7fs/ens/tp_cregut/openjml
```

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

```
java -jar ${OPENJML}/openjml.jar -rac Date.java ExempleDate1.java
```

L'option `-rac` permet d'instrumenter le code pour vérifier pendant l'exécution du programme les contrats (Runtime Assertion Check).
1.3. Exécuter le programme en tapant :

```
java -cp ${OPENJML}/jmlruntime.jar:. ExempleDate1
```

Commenter.

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.