IN201 : 3 – JML

Author : Christophe Garion <garion@isae.fr>
Public : SUPAERO 2A
Date :

Résumé

Ce TP a pour but de vous faire manipuler la programmation par contrat (invariants, préconditions, postconditions) au travers de la conception et de l'implantation d'une classe `EnsembleEntier` en utilisant le langage de modélisation JML.

1 Objectifs

Les objectifs du TP sont les suivants :
– comprendre une spécification utilisant la programmation par contrat ;
– utiliser les contrats comme une aide à la mise au point ;
– réaliser une classe respectant ses contrats.
On trouvera des références sur le site présenté à la fin du sujet [1].

2 Manipulation de la classe `Date` et de JML

Récupérer la classe `Date` sur le site ainsi que les classes `TestDate` et `TestDateInc` dont les listings sont présentés respectivement sur les listings 1 et 2.

Listing 1– La classe `TestDate`

```
public class TestDate {
    public static void main (String args []) {
        // Construire les dates
        Date d1 = new Date(20, 5, 1989);
        Date d2 = new Date(13, 2, 1993);
        Date d3 = new Date(31, 6, 2001);

        // Afficher les dates
        System.out.println("d1 = " + d1);
        System.out.println("d2 = " + d2);
        System.out.println("d3 = " + d3);
    }
}
```

Listing 2– La classe `TestDateInc`

```
public class TestDateInc {
    public static void main (String args []) {
        // Construire une date
        Date d1 = new Date(20, 5, 1989);

        // Incrementer la date
        d1.augmenter(6);

        // Afficher la date
        System.out.println("d1 = " + d1);
    }
}
```

1. compiler les classes avec `javac` et les exécuter avec `java`. Que constatez-vous ?
2. compiler maintenant l'application avec `jmlc`. On peut choisir de placer les *bytecodes* instrumentés non pas dans le répertoire `classes` (qui sera réservé pour le code non instrumenté), mais dans un répertoire `classesJML`. Il suffit de faire : `jmlc -classpath $CLASSPATH:../classesJML -d ../classesJML Date.java TestDate.java`.
   Exécuter le programme avec `jmlrac`. Que constatez-vous ?
   NB : le compilateur `jmlc` permet d'instrumenter les classes qui lui sont passées en paramètres. 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 passées en paramètre de la ligne de commande sont instrumentées. Les autres ne le sont pas.
   On peut également passer un répertoire en paramètre de `jmlc`. Dans ce cas, tous les fichiers `.java` du répertoire sont instrumentés.

3 Présentation du problème

On cherche à modéliser un ensemble d'entiers sous la forme d'un objet. Pour cela on dispose d'une spécification d'une classe `EnsembleEntierTab` qui stocke les entiers dans un tableau. Un nouvel élément est ajouté à la fin du tableau. Cette modélisation va être utilisée pour implanter le crible d'Ératosthène.

4 Étude de la spécification de `EnsembleEntierTab`

Récupérer le source de la classe `EnsembleEntierTab` sur le site. À partir des spécifications écrites « en JML », répondre aux questions suivantes [1] :

1. peut-on ajouter un élément dans l'ensemble s'il est déjà présent ?
2. peut-on toujours ajouter un élément dans l'ensemble ?
3. si l'on ajoute trois fois l'élément 1 dans l'ensemble et qu'on le supprime (ôte) une fois, l'élément est-il encore présent dans l'ensemble ?
4. est-ce que la manière d'utiliser le tableau est bien spécifiée par les invariants de `EnsembleEntierTab` ? On se contentera d'une explication intuitive, informelle.

5 Implantation de la classe `EnsembleEntierTab`

La classe `EnsembleEntierTab` est partiellement implantée. On vous fournit trois classes qui vont permettre de tester l'ensemble d'entiers (`TestEnsembleEntierTab`) et de calculer les nombres premiers (`CribleEntier` et `NombresPremiersEntier`). Ces classes n'ont a priori pas à être modifiées.

1. compiler et exécuter la classe `TestEnsembleEntierTab` avec le JDK (la classe `EnsembleEntierTab` peut être compilée même si le code réel des méthodes a été omis). Que constatez-vous ? Est-il facile de savoir d'où viennent les erreurs qui doivent être corrigées ?
2. recommencer, mais cette fois-ci en utilisant `jmlc` et `jmlrac`. Que constatez-vous ?
3. compléter et/ou corriger le corps des méthodes de la classe `EnsembleEntierTab` ;
4. tester votre implantation en s'appuyant sur les deux programmes `TestEnsembleEntierTab` et `NombresPremiersEntier`. On activera bien entendu la vérification dynamique des contrats ;
   NB : compiler d'abord avec `javac`, car `jmlc` ne peut pas détecter toutes les erreurs de `javac`.
5. comparer le temps de calcul pour afficher les nombres premiers entre 2 et 100 avec ou sans instrumentation du code.

1. Les spécifications apparaissent dans la documentation javadoc et y sont plus lisibles.

Références

[1] *The Java Modeling Language (JML)*. URL : http://www.jmlspecs.org.