Rattrapage de Test et Vérification
2016–2017

- Durée : 2h
- Type : ExamManager ou copie papier
- Rendu : une archive contenant les sources Java à déposer dans le dossier `rendu-test-verif`, et, pour les réponses aux autres questions, un fichier PDF à déposer dans le même dossier, ou une copie papier à rendre aux surveillants
- Tous documents autorisés.
- Toutes vos affaires (sacs, vestes, ...) doivent être placées à l'avant de la salle.
- Aucun téléphone ne doit se trouver sur vous ou à proximité, même éteint.
- Les déplacements et les échanges ne sont pas autorisés.
- Aucune question ne peut être posée aux enseignants, posez des hypothèses en cas de doute.

Test (10 points)

Soit la méthode `tri_sans_doublon` qui devrait retourner `true` si un tableau d'entiers passé en paramètre est trié sans doublon par ordre croissant. Par exemple, `tri_sans_doublon([1,2,3])` devrait retourner `true` alors que `tri_sans_doublon([1,1,2])` et `tri_sans_doublon([1,4,2])` devraient retourner `false`.

```
1  public static boolean tri_sans_doublon(int[] t) {
2      for(int i=1;i<t.length;i++){
3          if (t[i] < t[i-1]) return false;
4      }
5      return true;
6    }
```

1. Construisez le graphe de flot de contrôle de la méthode `tri_sans_doublon`. (2pt)
2. Listez les sommets et les arêtes parcourues par les 3 données de test de l'énoncé. (1,5pt)
3. Si possible, identifiez un cas de test qui exécute la faute, mais n'aboutit pas à un état d'erreur. (1pt)
4. Écrivez, dans une classe de test JUnit, les méthodes de test correspondant aux données de test de l'énoncé. (1,5pt)
5. Déduisez-en l'erreur et corrigez-la. (1pt)
6. Écrire une implémentation incorrecte de la méthode `tri_sans_doublon` qui passe tous les tests de l'énoncé. (2pt)
7. Écrire une nouvelle donnée de test qui montre que votre méthode est incorrecte. (1pt)

Vérification (10 points)

1. Vous venez de trouver un bug dans la méthode `tri_sans_doublon` initiale de la partie Test. En fait elle ne correspond pas à l'énoncé.
   (a) Reformuler l'énoncé pour que cette méthode devienne correcte. (1pt)
   (b) Ecrire sa pré-condition. (0.5pt)
   (c) Ecrire sa post-condition. (2pt)
   (d) Ecrire son invariant de boucle. (2pt)
2. La méthode `tri_sans_doublon` que vous avez corrigée doit correspondre à l'énoncé.
   (a) Ecrire sa pré-condition. (0.5pt)
   (b) Ecrire sa post-condition. (2pt)
   (c) Ecrire son invariant de boucle. (2pt)

Si vous disposez d'un ordinateur sous exammanager, vous pouvez vous aider de `junit.jar` et de `Why3` pour vérifier vos résultats.