Activté 3 C306 - Ingénierie du logiciel

Session 2021 - Semestre 2

Consortium International e-Miage - Auteur: *Sébastien Choplin* sebastien.choplin@u-picardie.fr

On récupérera le travail de l'activité 2 Le dossier de travail sur le dépôt pour cette activité est **"activite3"**. **Le devoir sera rendu sous la forme d'un rapport PDF synthétisant le travail, le code sera sur le dépôt (GIT ou SVN) exclusivement.**

Le projet doit déjà respecter l'arborescence Maven pour la compilation :

```
activite3/pom.xml
activite3/src/main/java
activite3/src/test/java
```

On utilisera la plateforme d'intégration continue Jenkins (contactez votre tuteur pour l'accès).

Ce travail peut se réaliser en groupe, suivez les instructions de votre tuteur à ce sujet.

1. Première implémentation

On ne cherche pas à évaluer les performances, cela fera l'objet de l'exercice suivant, l'implémentation devra donc juste répondre au problème, ce n'est pas gênant si la solution n'est pas performante.

On n'utilisera pas de bibliothèque externe et on ne récupérera pas de solution toute faite, l'intérêt est de mettre en oeuvre les problématiques du cours.

Les adresses des ressources utilisées (serveur de versionning, plateforme d'intégration continue) seront fournies par le tuteur.

1. Donner une copie d'écran des rapports fournis par Jenkins (checkstyle, spotbugs, PMD, couverture de code) sur ce qui a été produit à l'activité 2.
   S'il reste des erreurs signalées dans les rapports, elles doivent être corrigées.
   Si on ne souhaite pas considérer certaines catégories d'erreurs, il ne faut pas les laisser mais modifier les fichiers de configuration de l'outil de vérification et adapter le pom.xml pour qu'il en tienne compte. (Exemple: interdire les 'magic number' dans les tests peut être vu comme une contrainte superflue, dans checkstyle cela peut être réalisé avec une exclusion de règle: https://maven.apache.org/plugins/maven-checkstyle-plugin/examples/suppressions-filter.html.)
2. Ecrire l'interface pour un 'resolveur' de grille de sudoku défini dans l'activité précédente.
3. Ecrire les tests correspondant aux méthodes définies dans votre interface.
4. Ecrire une implémentation simple de 'resolveur' (une version récursive utilisant un parcours en profondeur s'écrit en quelques lignes).
5. Donner une copie d'écran des rapports fournis par Jenkins (checkstyle, SpotBugs, PMD, tests et couverture), il ne doit pas rester d'erreurs et la couverture doit être complète.

2. Optimisation

1. Evaluer les performances (en CPU et mémoire) du programme
2. Refactoriser pour trouver des améliorations sans changer votre algorithme (et donner un nouveau rapport de performances)
3. Enrichissez votre algorithme pour écrire un solveur plus 'intelligent' et évaluez ses performances.

Les performances peuvent être évaluées sur des grilles 9x9, 16x16 et 25x25 (9x9 c'est un peu trop facile).

Ressources: un lecteur de grille et quelques grilles https://students.webadvise.fr/emiage/c306/sudoku/.