Génie Logiciel – TD 3

1. Code refactoring, threading et JUnit (< 2h20mn)

Cet exercice est à réaliser individuellement et à soumettre individuellement sur SVN au bout de 2h20 mn. Créez sous le package `training` trois packages nommés respectivement avec vos noms de famille pour y soumettre le code source adapté (`chrono`).

Retrouvez la 3ième version du code source de démonstration fourni par l'enseignant. Pour que le code fonctionne, vous devez ajouter la librairie externe JUnit dans votre projet Eclipse. Vous pouvez désormais créer un répertoire `lib` (en parallèle que `src`) pour y mettre les librairies externes utilisées par le projet.

Vous améliorez d'abord le programme en extrayant des constantes utilisées comme paramètres de l'interface graphique chronomètre. Ensuite, vous ajoutez un bouton `Stop` permettant à l'utilisateur d'arrêter le chronomètre. Le comportement du bouton `Start/Pause` reste inchangé. Après, vous trouvez une solution d'adaptation du comportement du bouton `Clear` pour que la réinitialisation du chronomètre ne soit possible que lors que le chronomètre est arrêté.

Etudiez les tests existants pour notre sujet arbre binaire qui se trouvent dans le package `test.tree`. Créez ensuite certains tests pour tester le fonctionnement des classes dans le package chrono. Respectez l'ordre de tests (4 classes de TestCase) :

1) Tester les compteurs de bases : `Counter` et `BoundedCounter`.
2) Tester le compteur cyclique `CyclicCounter`, surtout en mettez les valeurs bornées utilisées par le chronomètre.
3) Tester le chronomètre.
4) Tester également le calcul des coordonnées x et y basé sur le radian, utilisé dans l'interface graphique du chronomètre. Pour cela, vous faites des refactoring du code pour que ce calcul devienne indépendant de la classe `ChronometerGUI`.

Créez enfin une TestSuite qui regroupe toutes les classes de TestCase.

2. Répartition des tâches et l'avancement de projet (>1h20mn)

Vous avez fait vos efforts pour ébaucher votre première version du modèle fonctionnel (diagramme de classes) pour votre projet. Maintenant, vous réfléchissez à une solution vous permettant de travailler en parallèle (3 membres de l'équipe) sur la programmation du projet.

Une répartition des tâches sur le développement des modules fonctionnels du projet est ainsi nécessaire. Cette répartition doit assurer aussi une avancée du projet **testable** au fur et à mesure. Pensez donc à définir les priorités des tâches, en prenant compte les liens (dépendances) logiques entre elles. **En fin de cette séance de TD,** envoyez à l'enseignant un email intitulé **GL-NomProjet-RépartitionV1** dans lequel vous expliquez et justifiez brièvement votre stratégie de répartition de tâches.