POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

TP 2 : Génération de cas de test basé sur SBST
Cours LOG6305

Prof. Giuliano Antoniol
Chargé de Lab :
Houssem Ben Braiek

Date de lab : 30/01/19
Date d'échéance 1 : 12/02/19
Date d'échéance 2 : 26/02/19

Critères d'évaluation

1. Un rapport contenant la partie 1 sous format PDF et renommé comme *prénom_nom_de_famille_matricule.pdf*.
2. Le code complété de la partie 2 doit être compressé en zip et renommé comme *prénom_nom_de_famille_matricule.zip* ;

Ce travail devrait être soumis à l'adresse mail : houssem.ben-braiek@polymtl.ca au plus tard le 12/02/19 à 23 :59h pour la partie 1 et le 26/02/19 à 23 :59h pour la partie 2 ; Sinon des pénalités seront appliquées (**5 points** de déduction pour le retard de toutes les **24 heures**).

Travail Demandé

But : L'application de l'approche SBST (Seach-Based Software Testing) proposée par Wegener (LOG6305_sbse-testing-09-EN, Page 95-96) au problème de génération des trois côtes du triangle.

Procédure :

1. Générez les données initiales (3 côtés d'un triangle) de façon aléatoire ;
2. Comparez les données générées avec un chemin d'exécution, où chaque nœud du chemin doit être satisfait par les données ;
3. Ajustez les données et comparez-les à nouveau du nœud supérieur du chemin. Répétez cette étape jusqu'à ce que tous les nœuds d'un chemin soient satisfaits par les données ;
4. Répétez les deux étapes 2-3 pour tous les chemins d'exécution de la classe *triangle.java*.

Partie 1 - Rapport

Objectif 1. Conditions uniques : Énumérez et nommez toutes les conditions uniques dans la classe *triangle.java*.

Objectif 2. Chemins d'exécution : Représentez chaque chemin d'exécution par une séquence des conditions uniques sous la forme : $Chemin_i$: $\{cond_1, cond_2, ..., cond_n\}$.

Objectif 3. Graphe de flot de contrôle : Dessinez le graphe de flot de contrôle contenant tous les chemins d'exécution dans la classe *triangle.java*.

Partie 2 - Code

Objectif 1. SearchBasedTest : Ajoutez une nouvelle classe *prenom_nom.SearchBasedTest* qui hérite de la classe *tp6305.CoverageTest* :

1. Définissez les opérateurs de comparaisons comme $>, =, <, ...$(par exemple, dans un *enum*) ;
2. Représentez les données d'entrée du triangle (par exemple, une variable pour chacun des côtés) ;

3. Ajoutez une classe/structure de données ExecPath pour stocker tous les chemins d'exécution possibles de la classe triangle (par exemple, un *ArrayList*) de manière que vous pouvez itérer sur les chemins d'exécutions et leurs nœuds correspondants sur lequels vous calculez le fitness des données générées ;
4. Implémentez une méthode pour initialiser les branches de la classe *triangle.java* ;
5. Copiez la méthode *computeBranchCoverage* de la classe *RandomCoverageTest*.

Objectif 2. FitnessFunction : Implémentez une nouvelle classe *FitnessFunction* qui permet de calculer le fitness basé sur l'approche de Wegener.

Objectif 3. Génération des données : Implémentez une nouvelle méthode *prenom_nom.SearchBasedTest.generateTestData(StringBuilder, float[])* qui se base sur l'approche SBST.

Objectif 4. Documentation : Les chemins d'exécution doivent être clairement représentés dans le code ainsi que chaque condition ou chemin d'exécution doit être clairement commenté ; Sinon des points de pénalité seront appliqués. Assurez-vous que vos chemins d'exécutions représentent bien la logique de la classe *triangle.java* et pourraient fonctionner avec d'autres fonctions de fitness.