tp2_gentest

tp2 de generacion automatica de tests

Autor: Ezequiel Dario Gambaccini Año: 2017 1C

El trabajo consiste en 2 generadores de tests (RandoopTestGenerator, EvoTestGenerator), que usan a randoop y evosuite para generar tests.

Estos generadores derivan de una clase base TestGenerator, la cual provee metodos para ejecutar comandos (ya que los generadores solo devuelven el string de los comandos a ejecutar), y metodos que generan comandos para:

- Compilar
- Correr JUnit con jacoco
- Correr PITest
- Procesar resultados de jacoco

Luego, hay un par de metodos para procesar los .csv de resultados de jacoco y pitest y generar un csv con la compilacion de resultados de ejecución de todas las pruebas.

Para generar los tests, se crea un objeto de tipo TestGenerator basado en alguna de las herramientas (randoop o evo), pasando como parámetro y se llama al método generateTestSuites(numero_de_test_suites_a_generar).

Esto va a crear la carpeta tests/(herramienta)/src si no existe, y va escribir los tests generados ahi.

Para hacer compilación y análisis, se llama al método run(numero_de_test_suites_a_generar), que genera las carpetas tests/(herramienta)/bin, reports/(herramienta) y reports/(herramienta)/(testclass)/{0..numero_de_test_suites_a_generar}.

Luego de generar las carpetas de binarios y reportes, genera comandos para compilar los tests generados, ejecutar los tests con junit y jacoco, ejecutar los tests con PITest, y generar los .csv a partir de los resultados de junit y jacoco a través de jacococli.

Después, ejecuta todos esos comandos generados en el siguiente orden:

1. Compilar
2. Pruebas con PITest
3. Pruebas con JUNIT y JACOCO
4. Procesar resultados de JUNIT y JACOCO en un csv

Una vez ejecutados los comandos, la función generateResults(numero_de_test_suites_a_generar) procesa todos los csvs generados por PITest y Jacoco para luego escribir a un .csv la clase, herramienta, % line coverage, % branch coverage, % mutation score, y emitir el resultado promedio de esos valores para {numero_de_test_suites_a_generar}.

Para la ejecución realizada de 30 tests por clase, por herramienta, los resultados son los siguientes:

	Avg Line Coverage		Avg Branch Coverage		Avg Mutation Score	
Class	**Randoop**	**Evosuite**	**Randoop**	**Evosuite**	**Randoop**	**Evosuite**
collections.comparators.FixedOrderComparator	0.852	0.962	0.700	0.931	0.400	0.857
collections.iterators.FilterIterator	0.564	0.847	0.083	0.600	0.429	0.817
collections.map.PredicatedMap	0.607	1.000	0.412	1.000	0.482	0.996
math.genetics.ElitisticListPopulation	0.950	0.978	0.833	0.939	0.600	0.789

En el archivo test_generator.py, en el main hay una variable boleana generate que indica si generar o no los tests, y una variable test_suites que indica cuantos tests a generar, ejecutar y analizar.

Para ejecutar el script es necesario python 3 y java 8.

La ejecución consiste en python test_generator.py. Probado en Windows. En *nix debería funcionar también.