DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Ingeniería de software II

Trabajo Práctico N°3

Integrante	LU	Correo electrónico
Esteban Luciano Rey	657/10	estebanlucianorey@gmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

http://www.exactas.uba.ar

Índice

1. Respuestas **2**

1.1. Ejercicio 1 2

1.2. Ejercicio 2 2

1.3. Ejercicio 3 2

1.4. Ejercicio 5 2

1. Respuestas

1.1. Ejercicio 1

Randoop generó 386 tests y ninguno falló

1.2. Ejercicio 2

JaCoCo reporta 46 de 53 líneas cubiertas y 25 de 26 branches cubiertos.

1.3. Ejercicio 3

Al ejecutar Randoop por 1 minuto con el nuevo código, se generan 1621 tests de los cuales 188 fallan

1.4. Ejercicio 5

Pi test construye 50 mutaciones y da como score %80 El mejor score que se pudo lograr mediante el agregado de tests custom fue del %94. Esto se logró refactorizando en una función estática a la función repOk: como los mutantes se encontraban en ella, se necesitaba poder hacer testing sobre la misma: es decir testear la función que decidía si mi estructura pasaba o no todos los tests. Para ello se generó una clase paralela a StackAr, llamada ExtendedStack la cual utiliza como repOk a la función estática de StackAr y presenta errores para poder testear a la función repOk.

Las mutaciones que no pudieron ser evitadas se efectuaron por parte a código aportado por la cátedra (hashCode, equals): en el caso del equals, el mutante se encuentra en la evaluación de readIndex al final de todas las clausulas, que como se respeta la representación de la estructura, una desigualdad por este atributo nunca se da (se generan mutantes en código muerto, con lo cual dan equivalencias). Por parte del hashCode, parte de las mutaciones se pudieron eliminar haciendo tests sobre la implementación del método en si: se especificó en el test el uso del primo 31 como número mágico, otra parte de las mutaciones no pudieron ser eliminadas ya que se abordaban temas de overflow de enteros para la generación del hash.

Para eliminar las mutaciones sobre el equals basta cambiar el orden de evaluación del predicado sobre readIndex hasta antes de que se evalúe la igualdad de los arrays elems. Por parte del hashCode, para poder eliminar completamente todos los mutantes, se debería poder tener información sobre el arreglo elems (el cual es privado y no se puede tener acceso externo a él)