Práctica: Pruebas basadas en cobertura de grafos (II)

Departamento de Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación, URJC

4 de diciembre de 2017

Ejercicio 1 (individual)

En este ejercicio tienes que utilizar `travis-ci.org` como servidor de integración continua para que se ejecuten automáticamente los tests. El proyecto en Eclipse deberá ser creado como un proyecto Maven.

Configuración de travis

Entra en `travis-ci.org` utilizando tu cuenta de GitHub para realizar este ejercicio. Sincroniza tus respositorios y selecciona el proyecto GitHub de esta práctica para que travis pueda ejecutar los tests de tu proyecto.

Tienes que añadir a tu proyecto en GitHub el fichero `.travis.yml` con el siguiente contenido en una línea: `language: java`

Ojo, el espacio después de los dos puntos es necesario.

Configuración de un proyecto Eclipse como proyecto Maven

El código de esta práctica lo tienes que crear en un proyecto Maven dentro de Eclipse.

Para crear un proyecto Maven desde Eclipse:

1. Crea nuevo proyecto con `New Project`. Selecciona `Maven/Maven Project`.
2. Pulsa 3 veces en Next.
3. Escribe en la caja de texto Group id: `org.urjc`
4. Escribe en la caja de texto Artifact id: `isi.travis`
5. Pincha sobre `pom.xml` en el proyecto creado y cambia la versión de junit a la 4.10
6. Salva todo (`File->Save all`)

En un proyecto maven puedes desde la línea de comandos ejecutar los tests: `mvn test`

Recuerda asegurarte de que el fichero `pom.xml` forma parte del repositorio git.

Desarrolla este ejercicio en un repositorio de git en el que para cada tarea distinta utilices una rama distinta: escritura de un test nuevo, escritura de nueva funcionalidad o corrección de un fallo en el código que estás probando, etc. Ve mezclando en la rama `master` los commits de las ramas que vayas creando.

Para ir viendo cómo travis corre los tests, mantén siempre actualizado el repo GitHub que asocies con travis. Travis no ejecutará tus tests hasta que, una vez asociado correctamente el proyecto por primera vez (incluyendo el fichero .travis.yml), hagas después algún nuevo commit en el proyecto, ya sea en los tests o en el código que estás probando.

En cada *commit* escribe comentarios descriptivos que expliquen/justifiquen el código escrito o modificado en cada *commit*, y cualquier otro detalle que consideres relevante para entender la razón por la que has pensado que es oportuno crear el *commit*.

En este ejercicio tendrás que hacer pruebas unitarias de caja blanca utilizando cobertura del grafo de control de flujo del código.

Utiliza el código del fichero `PrintPrimes.java` para realizar este ejercicio.

1. La clase `PrintPrimes` permite calcular los primeros `n` números primos. Estudia el código. Compílalo y ejecútalo.

2. Dibuja el grafo de control de flujo del método `printPrimes()`. Etiqueta los nodos y los arcos con las líneas de código.

3. Escribe un test que visite el arco que va de la sentencia `while` a la sentencia `for` pero sin ejecutar el cuerpo del bucle `while`.

4. Escribe los requisitos de prueba que se deben satisfacer para garantizar la cobertura de nodos (*node coverage*). Haz lo mismo para la cobertura de arcos (*edge coverage*) y la de caminos principales (*Prime paths*).

5. Muestra un camino de prueba que satisfaga la cobertura de nodos (*node coverage*) pero que no satisfaga la cobertura de arcos (*edge coverage*). ¿Es viable dicho camino (es decir, puedes escribir un test que lo recorra)?

6. Muestra un camino de prueba que satisfga el criterio de cobertura de arcos (*edge coverage*) pero que no satisfaga el criterio de caminos principales (*prime paths*). ¿Es viable dicho camino (es decir, puedes escribir un test que lo recorra)?

7. Escribe en el fichero `PrintPrimesTest.java` pruebas unitarias que satisfagan los requisitos identificados para el criterio de cobertura de caminos principales (*primary paths*).

8. Explica si las pruebas escritas en el apartado anterior han servido para identificar algún fallo del código. En caso afirmativo describe cuál es el fallo y con qué prueba concreta lo has identificado.

Ejercicio 2 (individual)

Desarrolla este ejercicio en un repositorio de gitq

A continuación se muestra el código del método `equals()` de la clase `java.util.AbstractList<E>`:

```
public boolean equals (Object o)
{
  if (o == this) // A
    return true;
  if (!(o instanceof List)) // B
    return false;

  ListIterator<E> e1 = listIterator();
  ListIterator<?> e2 = ((List) o).listIterator();
  while (e1.hasNext() && e2.hasNext()) // C
  {
    E o1 = e1.next();
    Object o2 = e2.next();
    if (!(o1 == null ? o2 == null : o1.equals (o2))) // D
      return false;
  }
  return !(e1.hasNext() || e2.hasNext()); // E
}
```

1. Dibuja el grafo de control de flujo para este método.

2. Etiqueta nodos y arcos del grafo con el código. Puedes utilizar estas etiquetas para escribir los predicados en los arcos correspondientes:

```
A: o == this
B: !(o instanceof List)
C: e1.hasNext() && e2.hasNext()
D: !(o1 == null ? o2 == null : o1.equals(o2))
E: !(e1.hasNext() || e2.hasNext())
```

3. Sin escribir el código de pruebas en un fichero JUnit aparte, escribe los tests necesarios para cubrir todos los nodos (*node coverage*). Ejemplo de test:

```
// Test para llegar al 3er return
List<String> list1 = new ArrayList<String>();
List<String> list2 = new ArrayList<String>();

list1.add ("foo");
list2.add ("bar");

assumeFalse(list1.equals(list2));
```

4. Sin escribir el código de pruebas en un fichero JUnit aparte, escribe los tests necesarios para cubrir todas las parejas de arcos (*edge-pair coverage*). Ejemplo de test:

```
// Test para el requisito de prueba [1,3,4]
List<String> list1 = new ArrayList<String>();
List<String> list2 = new ArrayList<String>();

assumeFalse(list1.equals (null));
```

5. Sin escribir el código de pruebas en un fichero JUnit aparte, escribe los tests necesarios para satisfacer el criterio de cobertura de caminos primarios (*Prime path*). Ejemplo de test:

```
// Test para el requisito de prueba [1,3,5,6,7,8]
List<String> list1 = new ArrayList<String>();
List<String> list2 = new ArrayList<String>();

list2.add ("bat");
assumeFalse(list1.equals (list2));
```

Normas de entrega

Se dispondrá una tarea de entrega en el Aula Virtual en la que deberás especificar la url del repositorio GitHub de cada Ejercicio. Para el primer ejercicio deberás también proporcionar la url de `travis-ci.org`