P10- Análisis de pruebas: Cobertura

Cobertura

Un análisis de la cobertura de nuestras pruebas nos permite conocer la **extensión** de las mismas. En esta sesión utilizaremos la herramienta Jacoco para analizar tanto la cobertura de líneas como de condiciones de nuestros tests.

Esta herramienta se integra con Maven, de forma que podemos incluir el análisis de cobertura en la construcción de nuestro proyecto.

Bitbucket

El trabajo de esta sesión también debes subirlo a *Bitbucket*. Todo el trabajo de esta práctica deberá estar en el directorio **P10-Cobertura** dentro de tu espacio de trabajo, es decir, dentro de tu carpeta: ppss-2020-Gx-apellido1-apellido2.

Recuerda que si trabajas desde los ordenadores del laboratorio primero deberás configurar git y clonar tu repositorio de Bitbucket.

➪ ➪ Ejercicio 1: Proyecto cobertura

Crea un proyecto Maven (en la carpeta *ppss-2020-Gx-apellido1-apellido2/P10-Cobertura/*) un proyecto maven, con groupId = **ppss**, y artifactID= **cobertura**. El nombre del proyecto IntelliJ será **cobertura**.

Añade la siguiente clase al paquete “ejercicio1”.

```
package ejercicio1;
public class MultipathExample {
      public int multiPath1(int a, int b, int c) {

         if (a > 5) {
            c += a;
         }
         if (b > 5) {
            c += b;
         }
         return c;
    }
}
```

A) ¿Cuál es la complejidad ciclomática (CC) del método multiPathExample()? Implementa un número mínimo de casos de prueba para conseguir una **cobertura del 100% de líneas y de condiciones**. ¿Cuál es ese número? Explica por qué es diferente del valor de CC.

B) Obtén un **informe de cobertura** para dicho proyecto y familiarízate con el informe obtenido. Puedes abrir directamente el fichero index.html del informe en el navegador Firefox desde el menú contextual, seleccionando"Open in Browser". Nota: el valor "n/a" significa "Not applicable".

> **Nota:** Cuando abrimos un fichero html en el navegador desde InteliJ, en ocasiones no se visualiza correctamente. Si eso ocurre, ábrelo directamente desde el Gestor de Archivos.

C) Añade el siguiente caso de prueba al conjunto (a=7, b=7, c=7, resultado esperado = 7). Explica lo que ocurre al intentar generar de nuevo el informe de cobertura. ¿Se debería haber generado un nuevo informe? ¿Puedes explicar qué está ocurriendo y qué podemos hacer para solucionar el problema?

D) Recuerda que siempre tienes que ejecutar mvn clean, antes de generar un nuevo informe. Cambia el caso de prueba del apartado anterior por ((a=3, b=6, c=2, resultado esperado = 8). Ahora añade el siguiente método a la clase MultipathExample y genera de nuevo el informe. Añade los casos de prueba necesarios (utilizando un test parametrizado) para conseguir una cobertura del 100% de condiciones y decisiones y vuelve a generar el informe. Observa las diferencias y justifica el valor de CC para el nuevo método que has añadido

```
public int multiPath2(int a, int b, int c )
   if ((a > 5) && (b < 5)) {
      b += a;
   }
   if (c > 5) {
      c += b;
   }
   return c;
}
```

E) Añade el siguiente método a la clase MultipathExample y genera de nuevo el informe. Añade los casos de prueba necesarios (utilizando un test parametrizado) para conseguir una cobertura del 100% de condiciones y decisiones y vuelve a generar el informe. Observa las diferencias y justifica el valor de CC para el nuevo método que has añadido

```
public int multiPath3(int a, int b, int c )
   if ((a > 5) & (b < 5)) {
      b += a;
   }
   if (c > 5) {
      c += b;
   }
   return c;
}
```

➪ ➪ Ejercicio 2: Informes de cobertura

En el proyecto anterior, vamos a crear un nuevo paquete "ejercicio2", al que deberás añadir las clases de la carpeta/plantillas/ejercicio2. Cada fichero debes añadirlo donde corresponda.

A) Obtén los informes de cobertura de nuestros tests unitarios y de integración, teniendo en cuenta las nuevas clases (tendrás que modificar convenientemente el pom del proyecto.

B) Queremos "chequear" que se alcanzan ciertos niveles de cobertura con nuestras pruebas unitarias. Modifica el pom para que se compruebe de forma automática lo siguiente:

- ➡ A nivel de proyecto, queremos conseguir una CC con un valor mínimo del 90%, una cobertura de instrucciones mínima del 80%, y que no haya ninguna clase que no se haya probado.
- ➡ A nivel de clase, queremos conseguir una cobertura de líneas del 75%

C) Obtén de nuevo los informes de cobertura, y comprueba que el proceso de construcción falla. porque se incumplen dos de las reglas que hemos impuesto. Con respecto al nivel de proyecto, modifica la regla que no permite completar la construcción, cambiando el valor del contador correspondiente. Con respecto a nivel de clase, no queremos cambiar la regla, de forma que para que la construcción se lleve a cabo con éxito tendrás que excluir una de las clases del paquete ejercicio2, en la regla correspondiente. Para ello debes usar el elemento <exclude>:

```
<rule>
   <element>...</element>
   <excludes>
      <exclude>ejercicio2.NombreDeLaClase</exclude>
   </excludes>
   ...
</rule>
```

➪ ➪ Ejercicio 3: Proyecto Matriculacion

Para este ejercicio usaremos el proyecto multimódulo ***matriculacion*** de la práctica P07.

Para ello copia tu solución, es decir, la carpeta P07B-dbunit/matriculacion (y todo su contenido) en el directorio de esta práctica (*ppss-2020-Gx-apellido1-apellido2/P10-Cobertura/*).

Puedes borrar el fichero *matriculacion.iml* del proyecto (carpeta "matriculacion"), ya que no nos hará falta. A continuación simplemente abre el proyecto ***matriculacion*** desde IntelliJ (seleccionando la carpeta que acabas de copiar).

Se pide:

A) Modifica convenientemente el pom del módulo matriculacion-dao para obtener un informe de cobertura para dicho proyecto. Genera el informe a través del correspondiente comando maven.

B) Observa los valores obtenidos y justifícalos a nivel de proyecto y paquete ¿qué diferencia hay entre ellos?. Explica los valores de CC obtenidos a nivel de paquete y clase.

C) El proyecto matriculacion-dao también ejecuta las clases de matriculacion-comun. ¿por qué no aparecen en el informe?

D) Dado que tenemos un proyecto multimódulo, vamos a usar la goal jacoco:report-aggregate para generar un informe para las dependencias de cada módulo, aprovechándonos del mecanismo reactor de maven. (ver https://www.jacoco.org/jacoco/trunk/doc/report-aggregate-mojo.html)

   Para ello tendrás que comentar el plugin jacoco del proyecto matriculacion-dao, e incluirlo en el pom del proyecto matriculacion. En lugar de usar las goals report y report-integration, debes usar la goal report-aggregate. Si observas la documentación del enlace, verás que dicha goal no está asociada por defecto a ninguna fase de maven. Explica qué ocurrirá si no la asociamos a ninguna fase. ¿A qué fase deberíamos asociarla?

E) Ejecuta el comando maven correspondiente para obtener el informe agregado de cobertura para el proyecto multimódulo. Averigua dónde se genera dicho informe de cobertura. Observa las diferencias con el informe que hemos obtenido anteriormente.

Resumen

¿Qué conceptos y cuestiones me deben quedar CLAROS después de hacer la práctica?

NIVELES DE COBERTURA

- La cobertura es una métrica que mide la extensión de nuestras pruebas. Existen diferentes variantes de esta métrica, que se pueden clasificar por niveles, de menos a más cobertura. Es importante entender cada uno de los niveles.
- El cálculo de esta métrica forma parte del análisis de pruebas, que se realiza después de su ejecución..

HERRAMIENTA JaCoCo

- JaCoCo es una herramienta que permite analizar la cobertura de nuestras pruebas, calculando los valores de varios "contadores" (JaCoCo counters), como son: líneas de código, instrucciones, complejidad ciclomática, módulos, clases,... También se pueden calcular los valores a nivel de proyecto,, paquete, clases y métodos.
- JaCoCo puede usarse integrado con maven a través del plugin correspondiente. Es posible realizar una instrumentación de las clases on-the-fly, o de forma off-line. En nuestro caso usaremos la primera de las opciones.
- JaCoCo genera informes de cobertura tanto para los tests unitarios como para los tests de integración. Y en cualquier caso, se pueden establecer diferentes "reglas" para establecer diferentes niveles de cobertura dependiendo de los valores de los contadores, de forma que si no se cumplen las restricciones especificadas, el proceso de construcción no terminará con éxito.
- De igual forma, para proyectos multimódulo, se pueden generar informes "agregados", de forma que se tentan en cuenta todos y cada uno de los módulos del proyecto y sus dependencias entre ellos.