



INGENIERÍA DEL SOFTWARE I

Práctica 8, Sesión 2

EclEmma

Univ. Cantabria – Fac. de Ciencias

Laura Sánchez



Objetivos

- Realizar pruebas de caja blanca usando un plugin complemento de JUnit.
- Realizar automáticamente pruebas de caja blanca con distintos criterios de cobertura.
- Familiarizarse con el plugin EclEmma.



Contenido

- Recordatorio
- Combinación de técnicas de pruebas unitarias
- Pruebas de caja blanca con EclEmma
- Ejercicios prácticos



Recordamos...

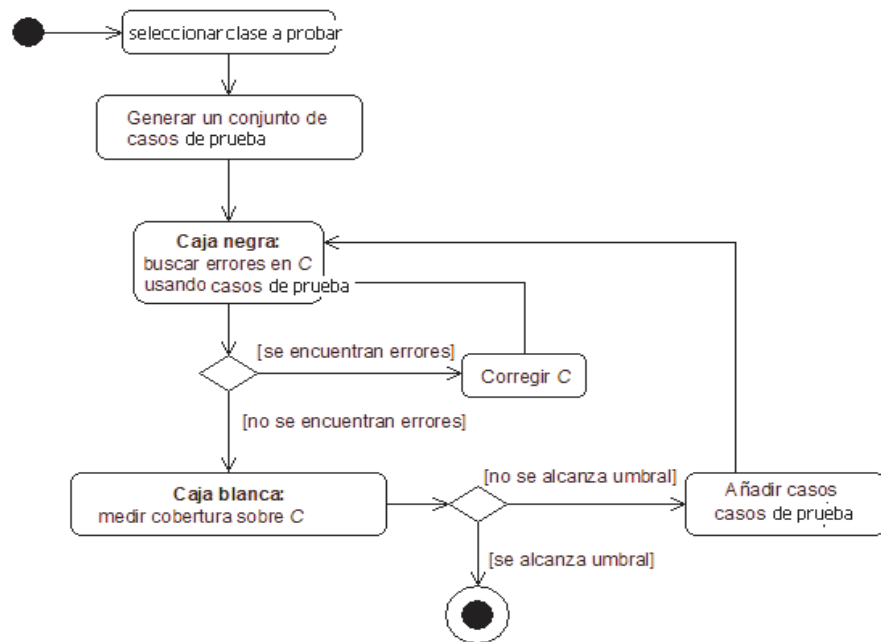
- En la sesión anterior se hicieron pruebas de caja negra sobre el sistema bancario.
- Se va a usar el plugin "EclEmma" para poder realizar pruebas de caja blanca



Técnicas de pruebas unitarias

- ¿por qué pruebas de caja negra y caja blanca?

–La idea es combinar los métodos de caja negra con los de caja blanca



Laura Sánchez - IS1



Pruebas de caja blanca

- Para instalarlo
 - Actualizar Eclipse introduciendo el site <http://update.eclemma.org/>
 - Help -> software update -> find and install -> search for new feature to install
 - Ir a la página <http://www.eclemma.org/> para más información

Laura Sánchez - IS1

P8-2.6



Pruebas de caja blanca

- Para ejecutar las pruebas de caja blanca
- Las marcas verdes indican sentencias ejecutadas, las rojas las no ejecutadas y las amarillas las parcialmente ejecutadas
- En la ventana inferior "coverage" se incluye el % de cobertura de sentencias.

Element	Coverage	Covered Instructio...	Total Instructions
ProyectoPruebas	25.8 %	204	791
src	25.8 %	204	791
dominio	30.9 %	133	430
Credito.java	0.0 %	0	190
Cuenta.java	50.9 %	82	161
Debito.java	47.1 %	16	34
Movimiento.java	65.5 %	19	29
Tarjeta.java	100.0 %	16	16
dominio.test	19.7 %	71	361

Laura Sánchez - IS1



Pruebas de caja blanca

- La cobertura puede realizarse por:
 - **bloques básicos:** un bloque básico es una secuencia de instrucciones bytecode sin saltos
 - **Líneas:** número de líneas de código java
 - **Instrucciones bytecode:** número de instrucciones bytecode dentro de los bloques básicos
 - **Métodos:** un método se considera cubierto si al menos un bloque básico del método ha sido ejecutado
 - **Tipos:** un tipo java es considerado cubierto si ha sido inicializado

Laura Sánchez - IS1

P8-2.8



Práctica 8

- Implementar una clase Lista que pueda incluir ciertos elementos y esté ordenada dependiendo del criterio de cada grupo:
 - P.e. la lista acepta números de 3 cifras mayores que cero
 - La lista incluye cadenas de caracteres y está ordenada por la longitud de la cadena
 - La lista incluye objetos de tipo *Persona* y está ordenada por la edad
 - La lista sólo incluye números múltiplos de 5, etc...
 - Aquí se valora la originalidad



Práctica 8

- Realizar pruebas de caja negra y caja blanca
 - Diseñar casos de prueba con los valores interesantes generados. Para el ejemplo de la lista que incluye números de 3 cifras mayores que cero:
 - Casos válidos $\rightarrow >0 \ \& \ \leq 999$
 - Casos no válidos $\rightarrow <0 \ \& \ \geq 999$
 - Ejemplo de caso de prueba:

```
Public void testOrdenar() {  
    lista.añadir(999);  
    lista.añadir(1);  
    lista.añadir(50);  
    lista.ordenar();  
    assertTrue(lista.getElement(0)==1)  
    assertTrue(lista.getElement(1)==50)  
    assertTrue(lista.getElement(2)==999)  
}
```



Práctica 8

- Una vez diseñados con java todos los casos de prueba, ejecutar JUnit.
- Si todas las pruebas son correctas, ejecutar Eclemma para comprobar cobertura de sentencias y alcanzar el mayor porcentaje de cobertura.
- ¿se puede realizar cobertura de condiciones también con Eclemma? Explicar por qué



Práctica 8

- Diseñar clases de equivalencia:
 - En este punto, se puede realizar una tabla con clases válidas y clases no válidas
- Cobertura one-wise y two-wise:
 - Valores interesantes {-1,0,999,1000}
 - Para 1 wise, que incluya en cada caso de prueba al menos una vez cada valor.

```
Public void testAñadir1 () {
    lista.añadir(-1);
    assertTrue(lista.getLength() == 0);
}
```
 - Para 2 wise, que incluya un par de valores interesantes cada caso de prueba.

```
Public void testAñadir2 () {
    lista.añadir(-1);
    lista.añadir(999);
    assertTrue(lista.getLength() == 1);
    assertTrue(lista.getElement(0) == 999);
}
```