Pruebas unitarias

Herramientas y métodos de Ingeniería del Software. Universidad de Almería

Joaquín Cañadas. Curso 2017

Crear de forma individual, un proyecto en Eclipse donde resolver los siguientes ejercicios, denominado Sesion05abc123 (sustituyendo abc123 por su usuario de la UAL)

Cada ejercicio se debe resolver en un paquete denominado **`ual.hmis.sesion05.ejercicioX`** , excepto el ejercicio 6, que tendrá su propio proyecto.

REQUISITO: plugin ECLEMMA para Eclipse (cálculo de la cobertura). Cualquier versión de Eclipse es válida.

Guardar dicho proyecto en un repositorio privado en GitHub en la cuenta personal (La resolución es **individual**). Dar acceso de lectura al profesor (usuario de GitHub: ualjjcanada).

Ejercicios Pruebas

1) Identifique las clases de equivalencia para el parámetro x en el siguiente método Java, e implemente los **casos de prueba** en JUnit necesarios para obtener un **100% de cobertura**:

```
public int transformar (int x) {
	if (x % 2 == 0)  // % Resto de una división entre enteros (mod)
		transformar (2 * x);
	else if (x % 3 == 0)
		transformar ( 3 * x );
	else if (x % 5 == 0)
		transformar (5 * x);
	else return x;

	return 0;
}
```

2) Escriba el código de un sencillo método en Java que reciba cómo parámetros de entrada dos valores tipo cadena de caracteres, **`username`** y **`password`**. El método debe comprobar que:

   - los dos valores sean distintos de cadena vacía
   - que la longitud de ambos sea menor de 30 caracteres.

   En tal caso, debe hacer la llamada a otro método o función denominado "**`compruebaLoginEnBD(username, password)`**" (que suponemos que existe y no hay que implementar) que devuelve TRUE si **`username`** y **`password`** son valores correctos existentes en la BD.

   Implemente además los **casos de prueba** necesarios para obtener un **100% de cobertura** del código del método utilizando JUnit, usando los **valores límite** adecuados donde proceda.

3) Escriba el código en Java de un método que tome un valor numérico entero cómo parámetro de entrada y devuelva una valor tipo cadena de caracteres cuyo contenido debe ser tantos caracteres '*' (asteriscos) como indica el parámetro de entrada. Por ejemplo, si el valor de entrada es 5 la cadena de salida sería '*****'. Si el valor de entrada es negativo la cadena de salida debe contener el texto 'número erróneo'.

   Implemente los **casos de prueba** en JUnit necesarios para obtener un **100% de cobertura**.

4) Escriba el código de un método en Java que tome cómo parámetros de entrada dos cadenas de caracteres, P1, P2, y devuelva otra cadena que contenga los caracteres de P1 que no están en P2.

   Implemente los **casos de prueba** en JUnit necesarios para obtener un **100% de cobertura**.

5) Escriba el código de un sencillo método en Java que reciba cómo parámetro de entrada una cadena de caracteres:

   `String subcadenaHastaPunto (String cadena)`

Y como salida devuelva la subcadena desde el primer carácter hasta el primer punto '.' (inclusive). Tenga en cuenta que:

a) si la "cadena" de entrada no tiene puntos, debe devolver "`Error: cadena sin punto`"; y

b) si "cadena" tiene un dígito (0..9) antes del primer punto, debe devolver el texto "`Error: cadena con dígito`".

Para la implementación, utilice los métodos Java:

`int strlen(String s)`: devuelve la longitud de la cadena
`boolean isDigit(char c)` : devuelve `true` si el carácter es un dígito (0..9)

Ejemplos:

Valor de entrada	**Resultado de salida**
"Mayor de edad"	"Error: cadena sin punto"
"Menor. De edad 3"	"Menor."
"Edad 3 años"	"Error: cadena con algún número"

Implemente los **casos de prueba** en JUnit necesarios para obtener un **100% de cobertura**, usando los **valores límite** adecuados donde proceda

6) A partir del código Java disponible en el repositorio GitHub de este ejercicio6, clone el repositorio en un repositoro privado en su cuenta GitHub personal. En él, implemente el método de la clase **`Alumno`** denominado **`calcularNotaActividad`** que recibe por parámetro el **`nombre`** de una actividad y que devuelva un **`double`** con la suma de la puntuación de todos los ejercicios de dicha actividad. En la ejecución de dicho método, se debe actualizar el valor de la propiedad (**`puntuacionTotal`**) de la clase **`Actividad`**.

   Implemente los **casos de prueba** en JUnit necesarios para obtener un **100% de cobertura** para el nuevo método **`calcularNotaActividad`**, usando los **valores límite** adecuados donde proceda.

7) Para cada uno de los ejercicios del 1 al 6, escriba una nueva clase de test que implemente los mismos casos de prueba, usando en este caso ***tests parametrizados***.