Práctica: Pruebas de caja negra (en parejas de alumnos)

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

9 de diciembre de 2020

Esta práctica se realiza en parejas. Utilizad un repositorio en GitHub para desarrollar las pruebas y el SUT (Software Under Test) que pase las pruebas que se describe en las cuatro secciones siguientes.

Para cada uno de los cuatro programas cada pareja tiene que:

1. Identificar los parámetros del SUT
2. Utilizando la técnica de la modelización de las entradas, crear caracterizaciones basadas en la funcionalidad, y sus particiones en bloques. Describid el criterio utilizado para crear las caracterizaciones, y para cada una, especificad los bloques en los que habéis dividido la entrada de cada parámetro.
3. Para crear casos de prueba, elige valores adecuados de cada bloque para cada parámetro. Presta atención a los casos frontera en pueda haber en cada bloque a la hora de elegir valores concretos para los tests.
4. Si hay varios parámetros, crea casos de prueba combinando bloques de cada parámetro con bloques de los otros parámetros.
5. Escribe el código de pruebas con JUnit, **antes de escribir el código del SUT**
6. Escribid el código del SUT y mejoradlo hasta que pase todas las pruebas.
7. Publicad en el foro de la asignatura el URL de vuestro repositorio. Utilizad los tests que habéis desarrollado para descubrir fallos en las implementaciones del SUT de otras parejas de compañeros.

1. Cálculo de años bisiestos

```
public class Bisiestos {

    // @param   a                un número entero positivo
    // @return  true             si a es un año bisiesto
    //          false            en caso contrario.
    // @throws  InvalidParameter  si a no es un parámetro válido.
    public boolean esBisiesto(int a) throws InvalidParameter {
        //
    }
}
```

2. Conversión de números romanos a base diez

```
// Ver https://es.wikipedia.org/wiki/Numeraci%C3%B3n_romana
public class RomanNumeral {

    // @param             s es un número romano
    // @return            el número s en base 10
```

```
    // @throws            InvalidParameter si s no es un número romano
    public int convierte(String s) throws InvalidParameter {
        //
    }
}
```

3. Embotelladora

```
class NoSolution extends Exception{
    String msg;

    NoSolution(String str) {
        msg = str;
    }

    public String toString(){
        return ("NoSolution: " + msg) ;
    }
}

public class Embotelladora {

    // @param      pequenas: número de botellas en almacén de 1L
    //             grandes : número de botellas en almacén de 5L
    //             total   : número total de litros que hay que embotellar
    // @return     número de botellas pequeñas necesarias para embotellar el total de líquido, teniendo
    //             en cuenta que hay que  minimizar el número de botellas pequeñas: primero
    //             se rellenan las grandes
    // @throws     NoSolution si no es posible embotellar todo el líquido
    public int calculaBotellasPequenas(int pequenas, int grandes, int total) throws NoSolution {
        //
    }

}
```

4. Descuento BlackFriday

```
public class DescuentoBlackFriday {

    // @param  precioOriginal es el precio de un producto marcado en la etiqueta
    //         porcentajeDescuento es el descuento a aplicar expresado como un porcentaje
    // @return el precio final teniendo en cuenta que si es black friday (29 de noviembre) se aplica
    //         un descuento de porcentajeDescuento
    // @throws InvalidParameter si precioOriginal es negativo

    public double precioFinal
    (double precioOriginal, double porcentajeDbescuento) throws InvalidParameter{
        //
    }

}
```

Entrega de la praćtica

Cada pareja tendrá que entregar un documento de texto de nombre `PRUEBA-IDM.txt` en cuya primera línea aparezca la url del proyecto en GitHub.

En este fichero hay que documentar, para cada uno de los 4 programas, el proceso de diseño de casos de pruebas (caracterizaciones, bloques, combinaciones de bloques, valores elegidos para cada test,...).

En el código de los tests es importante documentar en comentarios los nombres de las caracterizaciones de forma que se pueda saber a qué combinación de bloques de qué caracterizaciones corresponde cada caso de prueba. Ejemplo: *"Este test implementa la combinación de los bloques c1.bx, c2.by, c3.bz de las caracterizaciones c1, c2, c3"*.

Si encontráis dificultades relacionadas con la observabilidad o la controlabilidad al realizar alguno de los tests, documentadlas.

Documentad los fallos que gracias a vuestros tests habéis encontrado en el SUT desarrollado por otras parejas de compañeros, indicando el url del repo de los compañeros que identifique la versión concreta de su SUT en el que habéis localizado un fallo, y qué test de los vuestros es el que habéis usado para ello.