Tarea de Prueba de Software

La siguiente clase `MinMax` exporta una función `minMax` que, dado un array `xs` de enteros, devuelve un array de dimensión 2 con el mínimo y máximo valores contenidos en `xs`. Si `xs` no tiene elementos, se devuelve el array nulo.

```
public class MinMax {

    public static int[] minMax(int[] xs)
        { int mi, ma;
        int[] result = null;
        if (xs != null && xs.length != 0)
            { mi = ma = xs[0];
            for (int i = 1; i < xs.length; i++)
                { int n = xs[i];
                if (n > ma)
                    { ma =
                    n;
                } else if (n < mi)
                    { mi = n;
                }
            }
            result = new int[] { mi, ma };
        }
        return result;
    }
}
```

Pruebas de caja negra

Identifica las particiones equivalentes y valores límite para realizar una prueba de caja negra de la función `minMax`. Crea una clase de test `MinMaxTestCajaNegra`. Añade un método de prueba (`@Test`) por cada test de caja negra que pienses que es necesario para probar la función de manera satisfactoria. Usa nombres descriptivos para cada método de prueba.

Dado que la función recibe un array de números enteros, esos números deben ser un entero comprendidos en el siguiente rango:

- El valor mínimo es -2147483648 (-2 ^ 31).
- El valor máximo es 2147483647 (inclusive)(2 ^ 31 -1).

Las particiones equivalentes son:

- Array nulo
- Array con cero elementos
- Array con un solo elemento
- Array con varios elementos

Los límites de estos conjuntos son:

- Array nulo
- Array con varios elementos

Pruebas de caja blanca

Instalación de EclEmma

Para realizar esta parte de la tarea necesitas instalar el plugin EclEmma en eclipse. La forma más simple de instalar el plugin es utilizar el Eclipse Marketplace. Una vez instalado el plugin y reinicializado eclipse, en la barra de botones debe aparecer un nuevo botón similar al de ejecutar, pero con unos cuadrados verde y rojo debajo de la flecha blanca. Cada vez que ejecutes un programa con este nuevo botón, EclEmma monitoriza la ejecución y muestra un informe de cobertura en la vista `Coverage`.

La vista `Coverage`

La vista `Coverage` se puede gestionar con los iconos que aparecen en la esquina superior derecha. Los iconos más útiles son:

- **Doble aspa**: Los informes de cobertura son acumulativos. Con este botón pueden borrar los informes de las ejecuciones anteriores.
- **Flecha hacia abajo**: Abre un menú donde se puede seleccionar el tipo de cobertura que se desea. La nomenclatura de EclEmma a veces no coindice con la que hemos visto en clase. La correspondencia es la siguiente:
- `Line Counters` es cobertura de sentencias
- `Branch Counters` es cobertura de decisiones más cobertura de condiciones
- `Complexity` es la complejidad ciclomática (respecto a un grafo con nodos de condición)

Otras opciones interesantes de este menú son:

- **Show Types**: para que la información se muestre agrupada por clases
- **Hide Unused Elements**: para que no se muestren informes de clases que no se han usado

Cobertura de Pruebas

Crea una clase de prueba `MinMaxTestCajaBlanca` y añade los siguientes 4 métodos de prueba (`@Test`):

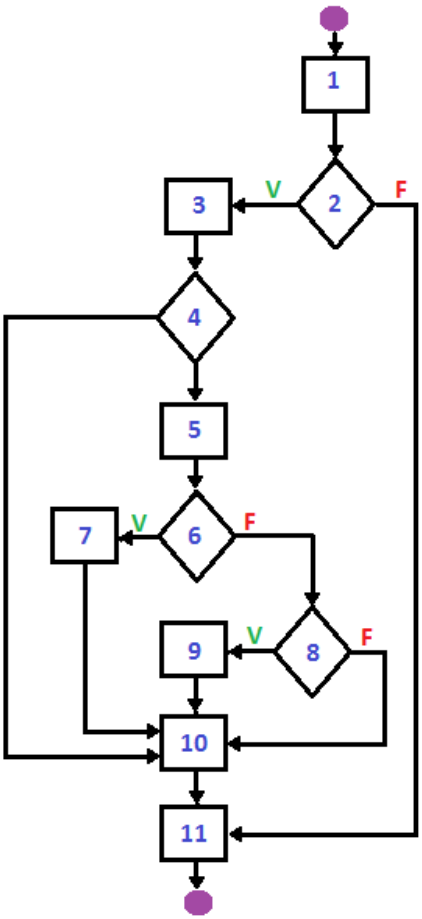
- `statementCoverage` que realiza el número mínimo de tests para alcanzar la cobertura de sentencias
- `decisionCoverage` que realiza el número mínimo de tests para alcanzar la cobertura de decisiones
- `conditionCoverage` que realiza el número mínimo de tests para alcanzar la cobertura de condiciones

- `decisionConditionCoverage` que realiza el número mínimo de tests para alcanzar la cobertura de decisiones y condiciones (lo que EclEmma llama `Branch Counters`)

Complejidad Ciclomática

Dibuja el grafo de control de flujo con nodos de decisión correspondiente a la función `minMax`. Puedes utilizar la herramienta que desees para crear la figura del grafo (PowerPoint, etc.). Además del grafo, la figura debe incluir el valor de la complejidad ciclomática calculada aplicando las 3 fórmulas vistas en clase; así como los caminos que forman la base. No es necesario que entregues el grafo en el formato en que lo hayas creado; pero sí debes entregar una versión en pdf o en algún formato de imagen estándar (jpg, etc.).

Crea una clase de prueba `MinMaxCiclomatica` e incluye un método de prueba (`@Test`) por cada camino de la base. Comprueba que si se ejecutan todos los test de `MinMaxCiclomatica` se alcanza la cobertura ciclomática.



Cálculo de la complejidad ciclomática:

A=Aristas=14
N=Nodos=11
C=A-N+2=14-11+2=5

R=Regiones cerradas=4
C=R+1=4+1=5

P=Nodos Predicados=4 ={2,4,6,8}
C= P+1 = 4+1 =5

Caminos base:
C1: 1, 2, 11
C2: 1, 2, 3, 4, 10, 11
C3: 1, 2, 3, 4, 5, 6, 7, 10, 11
C4: 1, 2, 3, 4, 5, 6, 8, 9, 10, 11
C5: 1, 2, 3, 4, 5, 6, 8, 10, 11

Entrega en GitHub

Una vez concluidas las pruebas, crea un repositorio en GitHub y sube los siguientes ficheros:

- *MinMaxTestCajaNegra.java*
- *MinMaxTestCajaBlanca.java*
- *MinMaxCiclomatica.java*
- *pdf o imagen del grafo de la complejidad ciclomática*

El repositorio creado es:
https://github.com/lorenard/Testing.git