ET.02.03 Segundo Problema – Grupo C01

Realizado por Guillermo Santos y Benedicto Tapetado.

1) Pseudocódigo del problema

```
public static Triangulo saberTipo(String[] ladosString,String[] angulosString){
    double[] lados=new double[3];
    double[]angulos=new double[3];
    boolean esRectangulo=false;
    boolean esObtuso=false;
    Triangulo t= null;
    for(int i=0;i<3;i++) {
        try {
            lados[i]=Double.parseDouble(ladosString[i]);
            angulos[i]=Double.parseDouble(angulosString[i]);
        }catch(NumberFormatException e) {
            return null;
        }
            if (lados[i]<=0 || angulos[i]<=0) {
                return null;
            }
    }
    double[] ladosAux =lados;
    Arrays.sort(ladosAux);
    if(ladosAux[2]>=(ladosAux[0]+ladosAux[1])) {
        System.out.println("Dados los lados no puede ser un triángulo");
    }else {
        double sumaAngulos = angulos[0]+angulos[1]+angulos[2];

        if(sumaAngulos!=180) {
            System.out.println("Dados los ángulos no puede ser un triángulo");
        }else {
            int ladoscoinciden=0;
            int anguloscoinciden=0;
            for (int i=0;i<2;i++) {

                if(lados[i]==lados[i+1])
                    ++ladoscoinciden;

                if(angulos[i]==angulos[i+1])
                    ++anguloscoinciden;
```

```
            }
            if (ladoscoinciden != anguloscoinciden) {
                System.out.println("Dados los lados y ngulos no es un tringulo posible");
            }else {
                t = new Triangulo(lados[0], lados[1], lados[2], angulos[0], angulos[1], angulos[2]);
                switch(ladoscoinciden) {
                case 0:
                    System.out.print("El tringulo es escaleno y");
                    break;
                case 1:
                    System.out.print("El tringulo es issceles y");
                    break;
                case 2:
                    System.out.print("El tringulo es equiltero y");
                    break;
                }
                for (int i=0;i<3;i++) {
                    if(angulos[i]==90) {
                        esRectangulo=true;
                    }
                    if(angulos[i]>90) {
                        esObtuso=true;
                    }
                }
                if(esRectangulo) {
                    System.out.println(" rectngulo");
                }else if(esObtuso){
                    System.out.println(" obtusngulo");
                }else
                    System.out.println(" acutngulo");
            }
        }
    }
    return t;
}
```

2) Identificación de las variables que se deben tener en cuenta para probar el método.

Las variables a tener en cuenta serán aquellas cuyos valores llegan directamente desde la entrada, como los lados del triángulo (llamados lado1, lado2 y lado3) y sus ángulos (angulo1, angulo2 y angulo3). Además, mediante los valores de entrada, obtenemos los valores de las variables de control como “sumaAngulos”,“ladosCoinciden”, “angulosCoinciden”, “esRectangulo” y “esObtuso”, que nos ayudarán a resolver diferentes decisiones dentro del código.

3) Valores de pruebas para las variables anteriores usando las técnicas vistas en teoría.

- Clases de equivalencia:
  - Lado1, lado2 y lado 3: (-inf,0] U (0, inf)
  - Ángulo1, ángulo2 y ángulo3: (-inf,0] U (0,180]
- Valores límite (variante pesada):
  - Lado1, lado2 y lado 3: -1,0,1
  - Ángulo1, ángulo2 y ángulo3: -1,0,1,179,180,181.
- Conjetura:
  - Lado1, lado2 y lado 3: -150000,2,2.5,200,150000.
  - Ángulo1, ángulo2 y ángulo3: -150000,0.3,0.5,0.7,45,60,90,150000.

*Añadimos en “conjetura” también valores para cubrir todas las condiciones/decisiones dentro del código, no únicamente valores que intuyamos que van a tener un comportamiento erróneo.

4) Número máximo posible de casos de pruebas que se podrían generar a partir de los valores de pruebas (combinatoria).

El número máximo de casos de prueba será el producto cartesiano de todos los valores posibles que pueda tomar cada variable. Los valores que pueden tomar los lados son un total de 8 y los de los ángulos 13, es decir, 8*8*8*13*13*13 = 1.124.864 casos de prueba totales.

5) Definimos un conjunto de casos de pruebas para cumplir con each use (cada valor una vez).

Usamos cada valor interesante al menos en un caso de prueba, obteniendo así tantos casos de prueba como valores tenga la variable con más parámetros. Utilizaremos la estrategia de combinación “each choice” vista en teoría para lograr esta cobertura.

Test suite 1:

- **CP1**: (-150000,-150000,-150000,-150000,-150000,-150000)
- **CP2**: (-1,-1,-1,-1,-1,-1)
- **CP3**: (0,0,0,0,0,0)
- **CP4**: (1,1,1,0.3,0.3,0.3)
- **CP5**: (2,2,2,0.5,0.5,0.5)
- **CP6**: (200,200,200,0.7,0.7,0.7)
- **CP7**: (150000,150000,150000,179,179,179)
- **CP8**: (1,2,200,180,180,180)
- **CP9**: (2,2,-1,181,181,181)
- **CP10**: (0,-150000,1,150000,150000,150000)
- **CP11**: (0,0,0,45,45,45)
- **CP12**: (1,1,2,60,60,60)
- **CP13**: (2,200,2,90,90,90)
- **CP13**: (2,2.5,2,90,90,90)

6) Definimos conjuntos de pruebas para alcanzar cobertura pairwise.

Hemos creado un proyecto de Excel en el que tendremos tres hojas. La primera está enfocada a alcanzar la cobertura pairwise en la que contamos con 169 casos de prueba.

Para acceder al proyecto haga clic aquí.

Hemos utilizado la siguiente herramienta online para generar los casos de prueba de forma exacta: https://pairwise.teremokgames.com/

7) Proponemos conjunto de casos de prueba para alcanzar cobertura de decisiones.

Tras realizar un estudio de decisiones, hemos alcanzado los siguientes casos de prueba:

Parámetros	Cp1	Cp2	Cp3	Cp4	Cp5	Cp6	Cp7	Cp8	Cp9
Lado 1	-150000	1	1	1	2	2	2	1	2
Lado 2	-1	2	1	2	1	2	2	1	2
Lado 3	0	2.5	1	1	2.5	2.5	2.5	1	2.5
Ángulo 1	-150000	179	179	180	60	0.3	0.5	60	90
Ángulo 2	-1	0.7	180	181	60	0.7	0.5	60	45
Ángulo 3	0	0.3	181	150000	60	179	179	60	45

Si se desea ver el proceso manual de generación de estos casos de prueba se debe acceder a la hoja llamada “7) Cobertura de decisiones” del proyecto de Excel.

8) Proponemos conjunto de casos de prueba para alcanzar cobertura MC/DC.

En este caso hemos evaluado el código de forma que cada posible valor utilizado por una condición determine la salida de la decisión al menos una vez. De esta forma obtenemos 21 casos de prueba.

Parámetros	Cp1	Cp2	Cp3	Cp4	Cp5	Cp6	Cp7	Cp8	Cp9	Cp10	Cp11
Lado 1	-150000	-150000	1	1	1	2	2	2	1	1	2
Lado 2	-1	-1	2	1	2	1	2	2	2	1	2
Lado 3	0	0	200	1	1	2.5	2.5	2.5	2.5	1	2.5
Ángulo 1	-150000	179	-150000	179	180	60	0.3	0.5	179	60	90
Ángulo 2	-1	0.7	-1	180	181	60	0.7	0.5	0.7	60	45
Ángulo 3	0	0.3	0	181	150000	60	179	179	0.3	60	45

Si se desea ver el proceso manual de generación de estos casos de prueba acceda a la hoja llamada “8) Cobertura MC|DC” del proyecto de Excel.

9) ¿Qué podemos decir acerca de la cobertura alcanzada por combinatoria, each use y pairwise?

La diferencia entre la cantidad de casos de uso obtenida por el uso de cada técnica es sustancial, pero también la cobertura que alcanzamos con cada una de ellas.

En nuestra opinión es innecesario y prácticamente imposible usar la técnica combinatoria, ya que se generan excesivos casos de uso (1.124.864) que realmente prueban varias veces las mismas partes del código. Sin duda esta sería la técnica menos eficiente aun dándonos una cobertura máxima.

En caso de la cobertura each use creemos que se queda algo justa, ya que al no tener en cuenta las decisiones puede llegar a tener un porcentaje de cobertura realmente bajo.

Para la cobertura pairwise creemos que ésta sí puede tener una cobertura mayor, eso sí, tendríamos 169 casos de prueba y lo más probable es que gran parte de ellos estén cubriendo la misma parte de código que otros casos de prueba (tal y como pasaba con la técnica combinatoria). Por otra parte, sí asegura que cada valor de cada variable se combina al menos una vez con cada otro valor de cada otra variable.

La cobertura que nos ofrece pairwise se encuentra entre la de each use y combinatoria, siendo pairwise la más atractiva ya que ni tiene tan baja cobertura como each use ni tantos casos de usos posibles como la opción de combinatoria (se encuentra en un intermedio).