UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRUPO BC.05

TRABAJO TEÓRICO PROBLEMA 2

Trabajo realizado por:

Enrique Reyes Gonzalez

Asignatura: Ingeniería del Software II

Grupo: BC.05

Titulación: Grado en Ingeniería Informática

1) Escribir, al menos el pseudocódigo correspondiente al método o a los métodos identificados

```java
package Implementación;
import java.util.Scanner;

public class triangulo {
        public static Scanner scanner = new Scanner(System.in);
        private int lado1;
        private int lado2;
        private int lado3;

          public triangulo(int a, int b, int c) {
            this.lado1 = a;
            this.lado2 = b;
            this.lado3 = c;
          }


        public boolean Equilatero() {
            return lado1 == lado2 && lado2 == lado3;
          }

          public boolean Isosceles() {
            return lado1 == lado2 || lado1 == lado3 || lado2 == lado3;
          }

          public boolean Escaleno() {
            return lado1 != lado2 && lado2 != lado3 && lado1 != lado3;
          }

          public boolean Rectángulo() {
            return lado1*lado1 + lado2*lado2 == lado3*lado3 ||
                   lado1*lado1 + lado3*lado3 == lado2*lado2 ||
                   lado2*lado2 + lado3*lado3 == lado1*lado1;
          }

          public boolean Acutángulo() {
            return lado1 < lado2 + lado3 && lado2 < lado1 + lado3 && lado3 < lado1 + lado2;
          }

          public boolean Obtusangulo() {
            return !Acutángulo();
          }
```

```java
           public static int controlLados() {
                   int lado=-1;
                   while(lado <= 0) {
               System.out.print("Ingrese el lado: ");
               try {
                   lado = scanner.nextInt();
               } catch (Exception e) {
                   System.out.println("Por favor no introduzca caracteres, únicamente valores enteros positivos");
                   scanner.nextLine();

               }

         }
                   return lado;
           }

           public static void main(String[] args) {
                 System.out.println("Bienvenido al programa, por favor introduzca 3 lados y se determinará que tipo de triangulo es");
             System.out.println("En caso de que introduzca un valor negativo se le volverá a pedir el mismo número");

             int a = triangulo.controlLados();
             int b = triangulo.controlLados();
             int c = triangulo.controlLados();

             triangulo triangulo = new triangulo(a, b, c);
             if (triangulo.Equilatero()) {
               System.out.println("Es un triángulo equilátero");
             }
             if (triangulo.Isosceles()) {
               System.out.println("Es un triángulo isósceles");
             }
             if (triangulo.Escaleno()) {
               System.out.println("Es un triángulo escaleno");
             }
             if (triangulo.Rectángulo()) {
               System.out.println("Es un triángulo rectángulo");
             }
             if (triangulo.Acutángulo()) {
               System.out.println("Es un triángulo acútangulo");
             }
             if (triangulo.Obtusangulo()) {
               System.out.println("Es un triángulo obtusángulo");
             }
           }
         }
```

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

3) Identificar los valores de pruebas para cada una de las variables anteriores usando las tres técnicas vistas en teoría, especificando para cada una cual es la que ha sido usada.

Parámetros	Valores objetivos	Clases de equivalencia	Valor limite (variante pesada)	Conjetura de errores	Valores totales
lado	(0, ∞)	(-∞,0](0,∞)	-1,0,1	-1000,1000	-1000, -1,0,1,1000
lado1	(0, ∞)	(-∞,0](0,∞)	-1,0,1	-1000,1000	-1000, -1,0,1,1000
lado2	(0, ∞)	(-∞,0](0,∞)	-1,0,1	-1000,1000	-1000, -1,0,1,1000
lado3	(0, ∞)	(-∞,0](0,∞)	-1,0,1	-1000,1000	-1000, -1,0,1,1000

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

Cada parámetro tiene 5 valores totales y hay 4 parámetros por tanto el número máximo de posibles casos de prueba será 5^4=625

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

Para cumplir each use tenemos que fijarnos que parámetro tiene mas valores totales, en este caso todos tienen los mismos; 5. Esto quiere decir que habrá 5 casos de prueba para poder realizar each use, estos son:

{-1000,-1000,-1000,-1000,-1000}

{-1,-1,-1,-1,-1}

{0,0,0,0,0}

{1,1,1,1,1}

{1000,1000,1000,1000,1000}

6) Defina conjuntos de pruebas para alcanzar cobertura pairwise usando el algoritmo explicado en clase. Se pueden comprobar los resultados con el programa PICT2

El pairwaise se define como todos los pares los casos de prueba deben visitar, al menos una vez, todos los pares de valores de dos parámetros cualesquiera

En este caso vamos a seleccionar los parámetros lado y lado1 para hacer todas las combinaciones posibles entre ellos dos.

lado	lado1
-1000	-1000
-1000	-1
-1000	0
-1000	1
-1000	1000
-1	-1
-1	0
-1	1
-1	1000
-1	-1000
0	0
0	1
0	1000
0	-1000
0	-1
1	1
1	1000
1	-1000
1	-1
1	0
1000	1000
1000	-1000
1000	-1
1000	0
1000	1

7) Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura de decisiones

A= (lado1 == lado2)

B=(lado2 == lado3)

C=(lado1 == lado3)

D=(lado1*lado1 + lado2*lado2 == lado3*lado3)

E=(lado1*lado1 + lado3*lado3 == lado2*lado2)

F=(lado2*lado2 + lado3*lado3 == lado1*lado1)

G=(lado1 < lado2 + lado3)

H=(lado2 < lado1 + lado3)

I=(lado3 < lado1 + lado2)

J= (lado <= 0)

```
public boolean Equilatero() {
        return lado1 == lado2 && lado2 == lado3;
      }
```

A	**B**	**A AND B**
0	0	0
0	1	0
1	0	0
1	1	1

```
public boolean Isosceles() {
        return lado1 == lado2 || lado1 == lado3 || lado2 == lado3;
      }
```

A	**C**	**B**	**A OR B OR C**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

```
public boolean Escaleno() {
```

```
        return lado1 != lado2 && lado2 != lado3 && lado1 != lado3;
    }
```

A	B	C	A’ AND B’ AND C’
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

```
public boolean Rectángulo() {
        return lado1*lado1 + lado2*lado2 == lado3*lado3 ||
            lado1*lado1 + lado3*lado3 == lado2*lado2 ||
            lado2*lado2 + lado3*lado3 == lado1*lado1;
    }
```

D	E	F	D OR E OR F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

```
public boolean Acutángulo() {
        return lado1 < lado2 + lado3 && lado2 < lado1 + lado3 && lado3 < lado1 + lado2;
    }
```

G	H	I	G AND H AND I
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

```
public boolean Obtusangulo() {
```

return !Acutángulo();

}

G	H	I	(G AND H AND I)’
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

while(lado <= 0)

I	I
0	0
1	1

{lado,lado1,lado2,lado3}

Simplificando todos los casos de prueba posibles podemos a escoger los siguientes para tener una cobertura completa

{1,1,1,2]

[-1,1,2,3}

{0,1,2,2}

{1,2,2,2}

{-1,3,2,5}

{1,4,2,2}

{1,8000,0,0}

8) Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura MC/DC.

public boolean Equilatero() {

return lado1 == lado2 && lado2 == lado3;

}

A	B	A AND B	
0	0	0	A,B
0	1	0	B
1	0	0	A

1	1	1	A,B

public boolean Isosceles() {

return lado1 == lado2 || lado1 == lado3 || lado2 == lado3;

}

A	C	B	A OR B OR C	
0	0	0	0	A,B,C
0	0	1	1	B
0	1	0	1	C
0	1	1	1	B,C
1	0	0	1	A
1	0	1	1	A,B
1	1	0	1	A,C
1	1	1	1	A,C,B

public boolean Escaleno() {

return lado1 != lado2 && lado2 != lado3 && lado1 != lado3;

}

A	B	C	A' AND B' AND C'	
0	0	0	1	A,B,C
0	0	1	0	C
0	1	0	0	B
0	1	1	0	B,C
1	0	0	0	A
1	0	1	0	A,C
1	1	0	0	A,B
1	1	1	0	A,B,C

public boolean Rectángulo() {

return lado1*lado1 + lado2*lado2 == lado3*lado3 ||

lado1*lado1 + lado3*lado3 == lado2*lado2 ||

lado2*lado2 + lado3*lado3 == lado1*lado1;

}

D	E	F	D OR E OR F	
0	0	0	0	D,E,F
0	0	1	1	F
0	1	0	1	E
0	1	1	1	E,F
1	0	0	1	D
1	0	1	1	D,F
1	1	0	1	D,E

1	1	1	1	D,E,F

public boolean Acutángulo() {

return lado1 < lado2 + lado3 && lado2 < lado1 + lado3 && lado3 < lado1 + lado2;

}

G	H	I	G AND H AND I	
0	0	0	0	G,H,I
0	0	1	0	I
0	1	0	0	H
0	1	1	0	I
1	0	0	0	G
1	0	1	0	G,I
1	1	0	0	G,H
1	1	1	1	G,H,I

public boolean Obtusangulo() {

return !Acutángulo();

}

G	H	I	(G AND H AND I)'	
0	0	0	1	G,H,I
0	0	1	1	I
0	1	0	1	H
0	1	1	1	H,I
1	0	0	1	G
1	0	1	1	G,I
1	1	0	1	G,H
1	1	1	0	G,H,I

while(lado <= 0)

I	I	
0	0	I
1	1	I

{lado,lado1,lado2,lado3}

Simplificando todos los casos de prueba posibles podemos a escoger los siguientes para tener una cobertura completa

{1,1,1,2]

[-1,4,2,3}

{0,1,6,4}

{1,4,2,600}

{-1,4,2,4}

{1,7,2,2}

{1,0,0,0}

{1,3,7,3}

{1,2,5,4}

{1,5,3,3}

9) Comente los resultados del número de los casos de pruebas conseguidos en los apartados 4, 5 y 6 ¿qué podría decirse algo de la cobertura alcanzada?

En el apartado 4 que son todos los posibles casos de prueba se obtendrá una cobertura del 100% ya que son todas las posibilidades que puede existir en el sistema.

Del caso 5 no alcanzaremos una cobertura tan satisfactoria como la del apartado 4 ya que habrá tantos casos de uso como tanto haya en la variable que más posibles valores tenga.

Del ejercicio 6 podemos concluir que la cobertura ser muy pobre ya que solo cogeremos todas las posibilidades, pero entre dos variables cualquiera del sistema y al no coger todas, la cobertura será ínfima.