VI. CODIGO REFACTORIZADO CON CORRECCIONES DE BAD SMELLS

ROJO – representa el código eliminado

VERDE – representa el código agregado

10 src/Principal.java View

```
@@ -58,15 +58,7 @@ public static void main(String[] args) {
                Triangulo trianguloIsoscelesObtusangulo = new Triangulo(p1, p2, p3);
                trianguloIsoscelesObtusangulo.mostrarTriangulo();
-
-               // Escaleno/acutangulo
-               /*p1 = new Punto(2.0, -9.0);
-               p2 = new Punto(-10.0, -17.0);
-               p3 = new Punto(-18.0, 17.0);
-
-               Triangulo trianguloEscalenoAcutangulo = new Triangulo(p1, p2, p3);
-               trianguloEscalenoAcutangulo.mostrarTriangulo();*/
-
+
                // Escaleno/obtusangulo
                p1 = new Punto(-9.0, 2.0);
                p2 = new Punto(-10.0, -17.0);
```

6 src/code/Punto.java View

```
@@ -12,11 +12,7 @@ public Punto(double x,double y) {
        public double calcularDistancia(Punto punto) {
                return Math.rint(Math.sqrt(Math.pow(punto.x - this.x,2) + Math.pow(punto.y - this.y, 2))*100)/100;
        }
-
-       public Punto devolverPunto() {
-               return this;
-       }
-
+
        public String mostrarPunto() {
                return ("x:" + this.x + " y:" + this.y);
        }
```

6 src/code/Punto.java View

```
@@ -12,11 +12,7 @@ public Punto(double x,double y) {
        public double calcularDistancia(Punto punto) {
                return Math.rint(Math.sqrt(Math.pow(punto.x - this.x,2) + Math.pow(punto.y - this.y, 2))*100)/100;
        }
-
-       public Punto devolverPunto() {
-               return this;
-       }
-
+
        public String mostrarPunto() {
                return ("x:" + this.x + " y:" + this.y);
        }
```

20 src/code/Triangulo.java View

```
@@ -37,9 +37,9 @@ public Triangulo(Punto puntoA, Punto puntoB, Punto puntoC) {
37 37                Double ladoB = lados.get(1);
38 38                Double ladoC = lados.get(2);
39 39
40    -              Double anguloCAB = calcularAnguloA(ladoA, ladoB, ladoC);
41    -              Double anguloABC = calcularAnguloA(ladoB, ladoC, ladoA);
42    -              Double anguloBCA = calcularAnguloA(ladoC, ladoA, ladoB);
   40 +              Double anguloCAB = calcularAngulo(ladoA, ladoB, ladoC);
   41 +              Double anguloABC = calcularAngulo(ladoB, ladoC, ladoA);
   42 +              Double anguloBCA = calcularAngulo(ladoC, ladoA, ladoB);
43 43
44 44                angulos.add(anguloCAB);
45 45                angulos.add(anguloABC);
@@ -48,11 +48,12 @@ public Triangulo(Punto puntoA, Punto puntoB, Punto puntoC) {
48 48                return angulos;
49 49        }
50 50
```

20 src/code/Triangulo.java View

```
@@ -37,9 +37,9 @@ public Triangulo(Punto puntoA, Punto puntoB, Punto puntoC) {
50 50
51    -       private Double calcularAnguloA(Double ladoA, Double ladoB, Double ladoC) {
52    -               Double numerador = Math.pow(ladoA, 2) - Math.pow(ladoB, 2) - Math.pow(ladoC, 2);
53    -               Double denominador = -2 * ladoB * ladoC;
54    -               Double cosenoA = Math.acos(numerador/denominador);
55    -               return Math.rint(Math.toDegrees(cosenoA)*100)/100;
   51 +       private Double calcularAngulo(Double ladoCatetoOpuestoA, Double ladoCatetoAdyacenteB, Double ladoCatetoAdyacenteC) {
   52 +               Double numerador = Math.pow(ladoCatetoOpuestoA, 2) - Math.pow(ladoCatetoAdyacenteB, 2) -
   53 +                               Math.pow(ladoCatetoAdyacenteC, 2);
   54 +               Double denominador = -2 * ladoCatetoAdyacenteB * ladoCatetoAdyacenteC;
   55 +               Double arcosenoA = Math.acos(numerador/denominador);
   56 +               return Math.rint(Math.toDegrees(arcosenoA)*100)/100;
56 57        }
```

20 src/code/Triangulo.java

```
@@ -37,9 +37,9 @@ public Triangulo(Punto puntoA, Punto puntoB, Punto puntoC) {
58 59        public ClasificacionPorLado retornarClasificacionPorLado(ArrayList<Double> lados) {
@@ -71,8 +72,7 @@ public ClasificacionPorLado retornarClasificacionPorLado(ArrayList<Double> lados
71 72                                return ClasificacionPorLado.ESCALENO;
72 73                        case 1:
73 74                                return ClasificacionPorLado.ISOSCELES;
74    -                       case 2:
75    -                               return ClasificacionPorLado.EQUILATERO;
   75 +                       case 2:
76 76                        case 3:
77 77                                return ClasificacionPorLado.EQUILATERO;
78 78                        default:
```