

Guía de Ejercicios N°7 – Curso de Introducción a Java

Ejercicio 1 Al programa del carrito de la Clase 5, agregue un parámetro que indique si hay un descuento fijo o % y ponga el valor del mismo.

Clase "Descuento"

```
1 package modelo;
2
3
4 public abstract class Descuento {
5
6     private double valor;
7
8     public double getValor() {
9         return valor;
10    }
11
12    public void setValor(double valor) {
13        this.valor = valor;
14    }
15
16    public abstract double valorFinal(double valorInicial);
17
18 }
```

Clase "DescuentoFijo"

```
1 package modelo;
2
3
4 public class DescuentoFijo extends Descuento {
5
6     public DescuentoFijo(double valor) {
7         this.setValor(valor);
8     }
9
10    public DescuentoFijo() {
11        this.setValor(0);
12    }
13
14    @Override
15    public double valorFinal(double valorInicial) {
16        return valorInicial - this.getValor();
17    }
18
19 }
```

Clase "DescuentoPorcentaje"

```
1 package modelo;
2
3
4 public class DescuentoPorcentaje extends Descuento {
5
6     public DescuentoPorcentaje(double valor) {
7         this.setValor(valor);
8     }
9
10    public DescuentoPorcentaje() {
11        this.setValor(0);
12    }
13
14    @Override
15    public double valorFinal(double valorInicial){
16        return valorInicial - (valorInicial * this.getValor());
17    }
18
19 }
```

Clase "Carrito" con el método "calcularTotal" modificado

```
1 package modelo;
2
3 import java.time.LocalDate;
4
5
6 public class Carrito {
7
8     private int numeroCarrito;
9     private ItemCarrito[] itemsCarrito = new ItemCarrito[3];
10    private LocalDate fechaCompra;
11
12    public Carrito() {
13    }
14
15    public Carrito(int numeroCarrito) {
16        this.numeroCarrito = numeroCarrito;
17        this.fechaCompra = LocalDate.now();
18    }
19
20    public double calcularTotal(Descuento descuento){
21        double total = 0;
22        for (int i = 0; i<3; i++){
23            total += this.itemsCarrito[i].calcularSubTotal();
24        }
25        return descuento.valorFinal(total);
26    }
27 }
```

Clase Principal "PruebaCarrito"

```
17 public class PruebaCarrito {
18
19     public static void main(String[] args) {
20
21         // Inicialización de variables
22         char tipoDescuento = 'f';
23         double valorFijo = 20;
24         double valorPorcentual = 0.2;
25         String archivo = "C:\\Users\\rochi\\OneDrive\\Documentos"
26             + "\\NetBeansProjects\\DesarrolladorJavaClase7"
27             + "\\src\\desarrolladorjavaclase7\\productos.txt";
28         String codigo;
29         String nombre;
30         int cantidad;
31         double precio;
32         Carrito carrito = new Carrito(1);
33         int i = 0;
34
35         // Lectura de archivo y creación de objetos
36         try {
37             for (String linea : Files.readAllLines(Paths.get(archivo))) {
38
39                 codigo = linea.split(",")[0];
40                 cantidad = Integer.parseInt(linea.split(",")[1]);
41                 precio = Double.parseDouble(linea.split(",")[2]);
42                 nombre = linea.split(",")[3];
43
44                 Producto producto = new Producto(nombre, codigo, precio);
45                 ItemCarrito itemCarrito = new ItemCarrito(producto, cantidad);
46                 carrito.getItemsCarrito()[i] = itemCarrito;
47
48                 i++;
49             }
50         } catch (IOException ex) {
51             Logger.getLogger(PruebaCarrito.class.getName()).log(Level.SEVERE, null, ex);
52         }
53
54         //Salida
55         System.out.println("Lista de Items: ");
56         System.out.println("Codigo Cantidad Precio Descripcion");
57         for (ItemCarrito item : carrito.getItemsCarrito()){
58             System.out.print(item.getProducto().getCodigo() + "      "
59                 + item.getCantidad() + "      "
60                 + item.getProducto().getPrecio() + "      "
61                 + item.getProducto().getNombre() + "\n");
62         }
63
64         // Ingreso por teclado del tipo de descuento
65         Scanner teclado = new Scanner(System.in);
66         System.out.println("Ingrese tipo de descuento (fijo, porcentual o ninguno): f/p/n ");
67         tipoDescuento = teclado.next().charAt(0);
68     }
```

```

69 // Inicializo un descuento de cualquiera de las subclases
70 // usando su constructor vacío para que el valor sea 0 por defecto
71 Descuento descuento = new DescuentoFijo();
72
73 switch (tipoDescuento){
74     case 'f':
75         descuento = new DescuentoFijo(valorFijo);
76         System.out.println("Total con un descuento de $" + descuento.getValor()
77             + " es: $" + carrito.calcularTotal(descuento));
78         break;
79     case 'p':
80         descuento = new DescuentoPorcentaje(valorPorcentual);
81         System.out.println("Total con un descuento del " + descuento.getValor()*100
82             + "% es: $" + carrito.calcularTotal(descuento));
83         break;
84     default:
85         System.out.println("Total bruto: $" + carrito.calcularTotal(descuento));
86         break;
87 }
88 }
89 }
90 }

```

Ejecuciones

Output - DesarrolladorJavaClase7 (run) X

```

run:
Lista de Items:
Codigo Cantidad Precio Descripcion
a      1      40.0  jabón en polvo
b      3      10.0  esponjas
c      2     100.0  chocolates
Ingrese tipo de descuento (fijo, porcentual o ninguno): f/p/n
n
Total bruto: $270.0
BUILD SUCCESSFUL (total time: 3 seconds)

```

Output - DesarrolladorJavaClase7 (run) X

```

run:
Lista de Items:
Codigo Cantidad Precio Descripcion
a      1      40.0  jabón en polvo
b      3      10.0  esponjas
c      2     100.0  chocolates
Ingrese tipo de descuento (fijo, porcentual o ninguno): f/p/n
f
Total con un descuento de $20.0 es: $250.0
BUILD SUCCESSFUL (total time: 2 seconds)

```

```
Output - DesarrolladorJavaClase7 (run) X
run:
Lista de Items:
Codigo Cantidad Precio Descripcion
a      1      40.0  jabón en polvo
b      3      10.0  esponjas
c      2     100.0  chocolates
Ingrese tipo de descuento (fijo, porcentual o ninguno): f/p/n
p
Total con un descuento del 20.0% es: $216.0
BUILD SUCCESSFUL (total time: 3 seconds)
```

Ejercicio 2 Agregue un nuevo descuento %, pero que tenga un tope Fijo en una nueva clase DescuentoPorcentajeConTope.

Clase "DescuentoPorcentajeConTope"

```
1 package modelo;
2
3 public class DescuentoPorcentajeConTope extends Descuento {
4
5     double tope;
6
7     public DescuentoPorcentajeConTope(double valor, double tope) {
8         this.setValor(valor);
9         this.tope = tope;
10    }
11
12    public DescuentoPorcentajeConTope() {
13        this.setValor(0);
14        this.tope = 0;
15    }
16
17    @Override
18    public double valorFinal(double valorInicial){
19
20        double desc = valorInicial * this.getValor();
21
22        if(desc < tope){
23            return valorInicial - desc;
24        }
25        else {
26            return valorInicial - tope;
27        }
28    }
29
30    public double getTope() {
31        return tope;
32    }
33
34    public void setTope(double tope) {
35        this.tope = tope;
36    }
37 }
```

Clase Principal "PruebaCarrito" (con la nueva opción)

```

65 // Ingreso por teclado del tipo de descuento
66 Scanner teclado = new Scanner(System.in);
67 System.out.println("Ingrese tipo de descuento (fijo, porcentual,"
68                   + " porc c/tope o ninguno): f/p/t/n ");
69 tipoDescuento = teclado.next().charAt(0);
70
71 // Inicializo un descuento de cualquiera de las subclases
72 // usando su constructor vacío para que el valor sea 0 por defecto
73 Descuento descuento = new DescuentoFijo();
74
75 switch (tipoDescuento){
76     case 'f':
77         descuento = new DescuentoFijo(valorFijo);
78         System.out.println("Total con un descuento de $" + descuento.getValor()
79                             + " es: $" + carrito.calcularTotal(descuento));
80         break;
81     case 'p':
82         descuento = new DescuentoPorcentaje(valorPorcentual);
83         System.out.println("Total con un descuento del " + descuento.getValor()*100
84                             + "% es: $" + carrito.calcularTotal(descuento));
85         break;
86     case 't':
87         DescuentoPorcentajeConTope descT = new DescuentoPorcentajeConTope(valorPorcentual, 50);
88         System.out.println("Total con un descuento del " + descT.getValor()*100
89                             + "%(hasta $" + descT.getTope()+") es: $" + carrito.calcularTotal(descT));
90         break;
91     default:
92         System.out.println("Total bruto: $" + carrito.calcularTotal(descuento));
93         break;
94 }
95 }
96 }
97 }

```

Ejecución

Output - DesarrolladorJavaClase7 (run) X

```

run:
Lista de Items:
Codigo Cantidad Precio Descripcion
a          1      40.0   jabón en polvo
b          3      10.0   esponjas
c          2     100.0   chocolates
Ingrese tipo de descuento (fijo, porcentual, porc c/tope o ninguno): f/p/t/n
t
Total con un descuento del 20.0%(hasta $50.0) es: $220.0
BUILD SUCCESSFUL (total time: 3 seconds)

```

Ejercicio 3 Cree 2 o 3 Tests para el método "precio" del carrito, y para los 3 descuentos DescuentoFijo, DescuentoPorcentaje y DescuentoPorcentajeConTope

Tests para DescuentoFijo

```

7  public class DescuentoFijoTest {
8
9      public DescuentoFijoTest() {
10     }
11
12     /**
13     * Test of valorFinal method, of class DescuentoFijo.
14     */
15     @Test
16     public void testValorFinal() {
17         System.out.println("valorFinal con DescuentoFijo");
18         double valorInicial = 100;
19         DescuentoFijo instance = new DescuentoFijo(20);
20         double expectedResult = 80;
21         boolean result = instance.valorFinal(valorInicial) == expectedResult;
22         assertTrue(result);
23     }
24
25     /**
26     * Test of valorFinal method, of class DescuentoFijo.
27     */
28     @Test
29     public void testValorFinalDescuentoNulo() {
30         System.out.println("valorFinal con DescuentoFijo vacío");
31         double valorInicial = 100;
32         DescuentoFijo instance = new DescuentoFijo();
33         double expectedResult = 100;
34         boolean result = instance.valorFinal(valorInicial) == expectedResult;
35         assertTrue(result);
36     }
37
38 }

```

Test Results X

modelo.DescuentoFijoTest X

▶▶ Tests passed: 100,00 %

▶ Both tests passed. (0,089 s)

✓

valorFinal con DescuentoFijo vacío

valorFinal con DescuentoFijo

Tests para DescuentoPorcentaje

```

7 public class DescuentoPorcentajeTest {
8
9     public DescuentoPorcentajeTest() {
10    }
11
12    /**
13     * Test of valorFinal method, of class DescuentoPorcentaje.
14     */
15    @Test
16    public void testValorFinal() {
17        System.out.println("valorFinal con DescuentoPorcentaje");
18        double valorInicial = 100;
19        DescuentoPorcentaje instance = new DescuentoPorcentaje(0.15);
20        double expectedResult = 85;
21        boolean result = instance.valorFinal(valorInicial) == expectedResult;
22        assertTrue(result);
23    }
24
25    /**
26     * Test of valorFinal method, of class DescuentoPorcentaje.
27     */
28    @Test
29    public void testValorFinalDescuentoNulo() {
30        System.out.println("valorFinal con DescuentoPorcentaje vacío");
31        double valorInicial = 100;
32        DescuentoPorcentaje instance = new DescuentoPorcentaje();
33        double expectedResult = 100;
34        boolean result = instance.valorFinal(valorInicial) == expectedResult;
35        assertTrue(result);
36    }
37
38 }

```

Test Results X

modelo.DescuentoPorcentajeTest X

▶▶	Tests passed: 100,00 %	valorFinal con DescuentoPorcentaje vacío
▶▶	Both tests passed. (0,106 s)	valorFinal con DescuentoPorcentaje
✓		

Tests para DescuentoPorcentajeConTope

```

7 public class DescuentoPorcentajeConTopeTest {
8
9     public DescuentoPorcentajeConTopeTest() {
10    }
11    /**
12     * Test of valorFinal method, of class DescuentoPorcentajeConTope.
13     */
14    @Test
15    public void testValorFinal() {
16        System.out.println("valorFinal con descuento porcentual < al tope ");
17        double valorInicial = 100;
18        DescuentoPorcentajeConTope instance = new DescuentoPorcentajeConTope(0.15,20);
19        double expResult = 85;
20        boolean result = instance.valorFinal(valorInicial) == expResult;
21        assertTrue(result);
22    }
23    /**
24     * Test of valorFinal method, of class DescuentoPorcentajeConTope.
25     */
26    @Test
27    public void testValorFinalTope() {
28        System.out.println("valorFinal con descuento porcentual == al tope ");
29        double valorInicial = 100;
30        DescuentoPorcentajeConTope instance = new DescuentoPorcentajeConTope(0.2,20);
31        double expResult = 80;
32        boolean result = instance.valorFinal(valorInicial) == expResult;
33        assertTrue(result);
34    }
35    /**
36     * Test of valorFinal method, of class DescuentoPorcentajeConTope.
37     */
38    @Test
39    public void testValorFinalMayorTope() {
40        System.out.println("valorFinal con descuento porcentual > al tope ");
41        double valorInicial = 100;
42        DescuentoPorcentajeConTope instance = new DescuentoPorcentajeConTope(0.3,20);
43        double expResult = 80;
44        boolean result = instance.valorFinal(valorInicial) == expResult;
45        assertTrue(result);
46    }
47    }

```

Test Results X

modelo.DescuentoPorcentajeConTopeTest X



Tests passed: 100,00 %



All 3 tests passed. (0,094 s)



valorFinal con descuento porcentual > al tope
 valorFinal con descuento porcentual < al tope
 valorFinal con descuento porcentual == al tope

Tests para el método "CalcularTotal" de la clase "Carrito"

```

7  public class CarritoTest {
8
9      Carrito carrito;
10     Producto producto;
11     ItemCarrito item1;
12     ItemCarrito item2;
13     ItemCarrito item3;
14     Descuento descuento;
15
16     public CarritoTest() {
17     }
18
19     @Before
20     public void setUp() {
21         carrito = new Carrito(1);
22         producto = new Producto("Fideos", "a", 150);
23         item1 = new ItemCarrito(producto, 2);
24         item2 = new ItemCarrito(producto, 4);
25         item3 = new ItemCarrito(producto, 1);
26         descuento = new DescuentoFijo();
27     }
28
29     /**
30      * Test of calcularTotal method, of class Carrito.
31      */
32     @Test
33     public void testCalcularTotal() {
34         System.out.println("calcularTotal sin descuento");
35         carrito.setItemsCarrito(new ItemCarrito[]{item1, item2, item3});
36         double expectedResult = 1050;
37         boolean result = carrito.calcularTotal(descuento) == expectedResult;
38         assertTrue(result);
39     }
40
41     /**
42      * Test of calcularTotal method, of class Carrito.
43      */
44     @Test
45     public void testCalcularTotalOtraCantidad() {
46         System.out.println("calcularTotal con otra cantidad de productos");
47         carrito.setItemsCarrito(new ItemCarrito[]{item1, item2, item2});
48         double expectedResult = 1500;
49         boolean result = carrito.calcularTotal(descuento) == expectedResult;
50         assertTrue(result);
51     }

```

Test Results X

modelo.CarritoTest X

Tests passed: 100,00 %

Both tests passed. (0,092 s)

calcularTotal con otra cantidad de productos
calcularTotal sin descuento