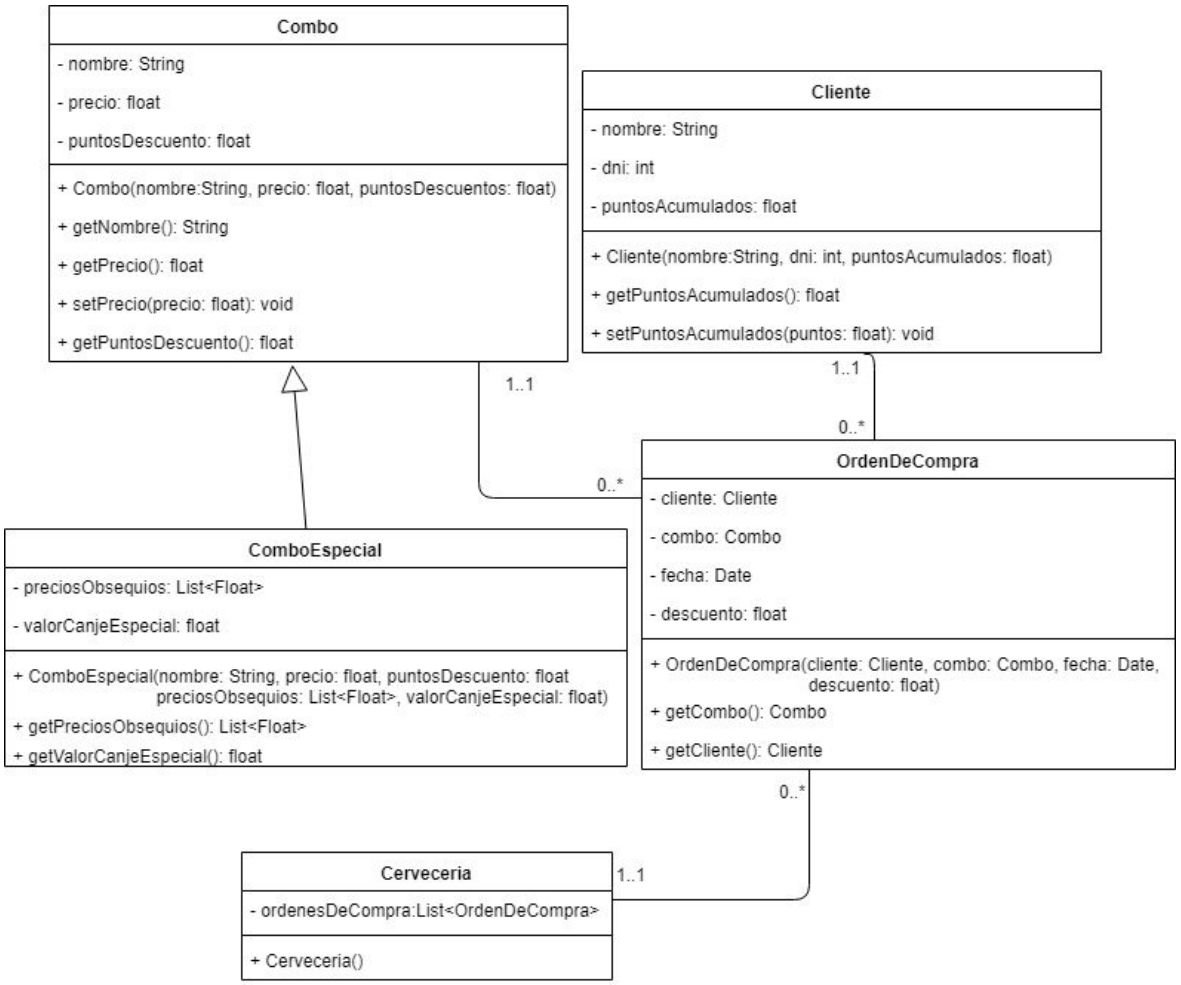
Examen 111Mil – Cervecería

Kevin está cumpliendo su sueño, y el de sus amigos, que es abrir una cervecería en el shopping de Adrogué. Ellos pensaron en ofrecer diferentes combos de cervezas inspirados en los superhéroes de Marvel. Específicamente, un combo estará compuesto por 2 pintas de cerveza artesanal y una pizza gourmet. Además, habrá combos especiales donde también se incorporarán obsequios inspirados en la liga de superhéroes. Existe también una tarjeta del Club de Cerveceros en la cual el cliente podrá acumular puntos con sus compras y obtener importantes descuentos. Como Kevin se enteró que cursamos el programa 111Mil, nos contactó para que lo ayudemos con el desarrollo de un sistema que le permita administrar la cervecería.

Ejercicio 1. Implementar desde el diagrama de clases

Kevin realizó un diagrama UML preliminar del sistema y nos pidió que implementemos la clase *ComboEspecial* según el diagrama.



Posible solución

```
public class ComboEspecial extends Combo{
	private List<Float> preciosObsequios;
	private float valorCanjeEspecial;


public ComboEspecial(String nombre, float precio, float puntosDescuento,
	List<Float> preciosObsequios,float valorCanjeEspecial) {
```

```
        super(nombre, precio, puntosDescuento);
        this.preciosObsequios=preciosObsequios;
        this.valorCanjeEspecial=valorCanjeEspecial;
    }

    public List<Float> getPreciosObsequios() {
        return preciosObsequios;
    }

    public float getValorCanjeEspecial() {
        return valorCanjeEspecial;
    }
```

Ejercicio 2. Implementar un método a partir de un enunciado

Kevin comenzó a trabajar en el mecanismo que permite saber cómo se realiza el canje de puntos acumulados de la tarjeta de Club de Cerveceros de un determinado cliente. En el caso de los combos "comunes", el descuento del precio del combo se realizará cuando el puntaje acumulado del cliente sea mayor o igual a los puntos necesarios del combo. Para ello Kevin implementó el siguiente método en la clase Combo, el cual devuelve *true* si puede canjear y *false* en caso contrario.
Tenga en cuenta que la variable *puntosAcumulados* siempre tendrá los puntos acumulados y actualizados al momento de realizar el canje de puntos de los clientes.

```
public boolean esPosibleCanjearCombo(float puntosAcumulados){
        return puntosAcumulados >= this.getPuntosDescuento();
}
```

En el caso de los combos especiales, el descuento al cliente se efectivizará si, además de que el puntaje acumulado del cliente sea mayor o igual a los puntos necesarios del combo, la suma de los precios de los obsequios superan un *valorCanjeEspecial*. Kevin nos pidió que sobre-escribamos el método *esPosibleCanjearCombo* en la clase *ComboEspecial* para realizar el descuento al cliente.

Por ejemplo, si un cliente tiene acumulados 1000 puntos, y quiere canjear un combo especial cuyos puntos necesarios son 500 y, los precios de obsequios del combo son [100, 150, 200] con el *valorCanjeEspecial* igual a 400, entonces la función *esPosibleCanjearCombo* retorna true dado que el puntaje acumulado del cliente (1000) es mayor o igual a los puntos necesarios del combo (500) y la suma de los precios de los obsequios (450) es mayor al *valorCanjeEspecial* (400).

Posible solución

```
    @Override
    public boolean esPosibleCanjearCombo(float puntosAcumulados) {
        float sumaPreciosObsequios=0;
        for(int i=0; i<this.preciosObsequios.size();i++){
            sumaPreciosObsequios+=preciosObsequios.get(i);
        }
        return super.canjearPuntosCombo(puntosAcumulados)
                    && sumaPreciosObsequios> this.valorCanjeEspecial;
    }
```

Ejercicio 3. Implementar un método a partir de un enunciado

Programar en Java la funcionalidad para obtener la cantidad de combos vendidos cuyos nombres sean igual al nombre de combo recibido por parámetro. Por ejemplo, suponga que existen 3 órdenes de compra, cuyos respectivos combos tienen los siguientes nombres: “Thor”, “Gamora” y “Thor”. Si el nombre de combo recibido por parámetros es “Thor”, entonces la función retorna 2.

Implemente los métodos que considere necesarios indicando para cada uno de ellos a qué clase corresponden.

Posible solución

Clase Cerveceria

```
public int obtenerCombosPorNombre(String nombreCombo){
      int cantidadCombos=0;

      for (OrdenDeCompra orden: ordenesDeCompra) {
            if(orden.getNombreCombo().equals(nombreCombo))
             cantidadCombos++;
        }

        return cantidadCombos;
    }
```

Clase OrdenDeCompra

```
public String getNombreCombo() {
                return this.combo.getNombre();
       }
```

Ejercicio 4. Seguimiento de código

Kevin programó el siguiente método en la clase Cerveceria pero no le puso un nombre representativo (ni tampoco a algunas variables).

Clase OrdenDeCompra

```
public float metodoSinNombreRepresentativo() {

       if (this.combo.esPosibleCanjearCombo(this.cliente.getPuntosAcumulados())) {

              float variableSinNombreRepresentativo1=this.combo.getPrecio()
                     -(this.getDescuento()*this.combo.getPrecio()/100);
              this.combo.setPrecio(variableSinNombreRepresentativo1);

              float variableSinNombreRepresentativo2=this.cliente.getPuntosAcumulados()
                     - this.combo.getPuntosDescuento();
              this.cliente.setPuntosAcumulados(variableSinNombreRepresentativo2);

  }
       return this.combo.getPrecio();
}
```

¿Qué imprimirá el programa al ejecutar el siguiente código?

```
Combo c1 = new Combo("Capitán América",300,125);
Combo c2 = new Combo("Gamora",199,50);
Combo c3 = new Combo("Thor", 250,350);

Cliente cliente = new Cliente("Juan Pérez", 23567892,500);

OrdenDeCompra o1 = new OrdenDeCompra(cliente, c1, new Date(),10);
OrdenDeCompra o2 = new OrdenDeCompra(cliente, c2, new Date(),20);
OrdenDeCompra o3 = new OrdenDeCompra(cliente, c3, new Date(),5);

System.out.println(o1.metodoSinNombreRepresentativo());
System.out.println(o2.metodoSinNombreRepresentativo());
System.out.println(o3.metodoSinNombreRepresentativo());
```

Solución

```
270.0
159.2
250.0
```

Ejercicio 5. Consulta SQL

Dado el diagrama de entidad-relación parcial, escriba la consulta SQL que liste para cada nombre cliente la cantidad de combos “Avispa” a los que se les ha aplicado un descuento superior o igual al 10%. Liste los resultados por nombre de cliente en orden alfabético.

Además, dadas las siguientes tuplas de ejemplo, determinar el resultado de la consulta.

Cliente		
32800115	Tatiana	200
30256456	Mariana	153.25
25456965	Anastasia	90.75

Combo			
1	Thor	150	120
2	Iron Man	250	500
3	Capitan America	190	150
4	Gamora	200	300
5	Hulk	350	400
6	Viuda Negra	210	175
7	Ant-Man	255	185
8	Pantera Negra	195	100
9	Avispa	500	950

OrdenCompra				
1	32800115	3	2019-01-30	4
2	32800115	3	2019-01-31	15
3	30256456	3	2019-03-30	20
4	25456965	5	2019-04-15	20
5	30256456	9	2019-05-09	10

Posible solución

```
SELECT  cli.nombre, COUNT(oc.IdOrden) FROM  Cliente cli

INNER JOIN OrdenDeCompra oc ON (cli.DNI=oc.Cliente_DNI)

INNER JOIN Combo c ON (oc.Combo_idCombo=c.idCombo)

WHERE

c.nombre= "Avispa" AND oc.descuento >= 10

GROUP BY cli.nombre

ORDER BY cli.nombre
```

Respuesta

Mariana 1