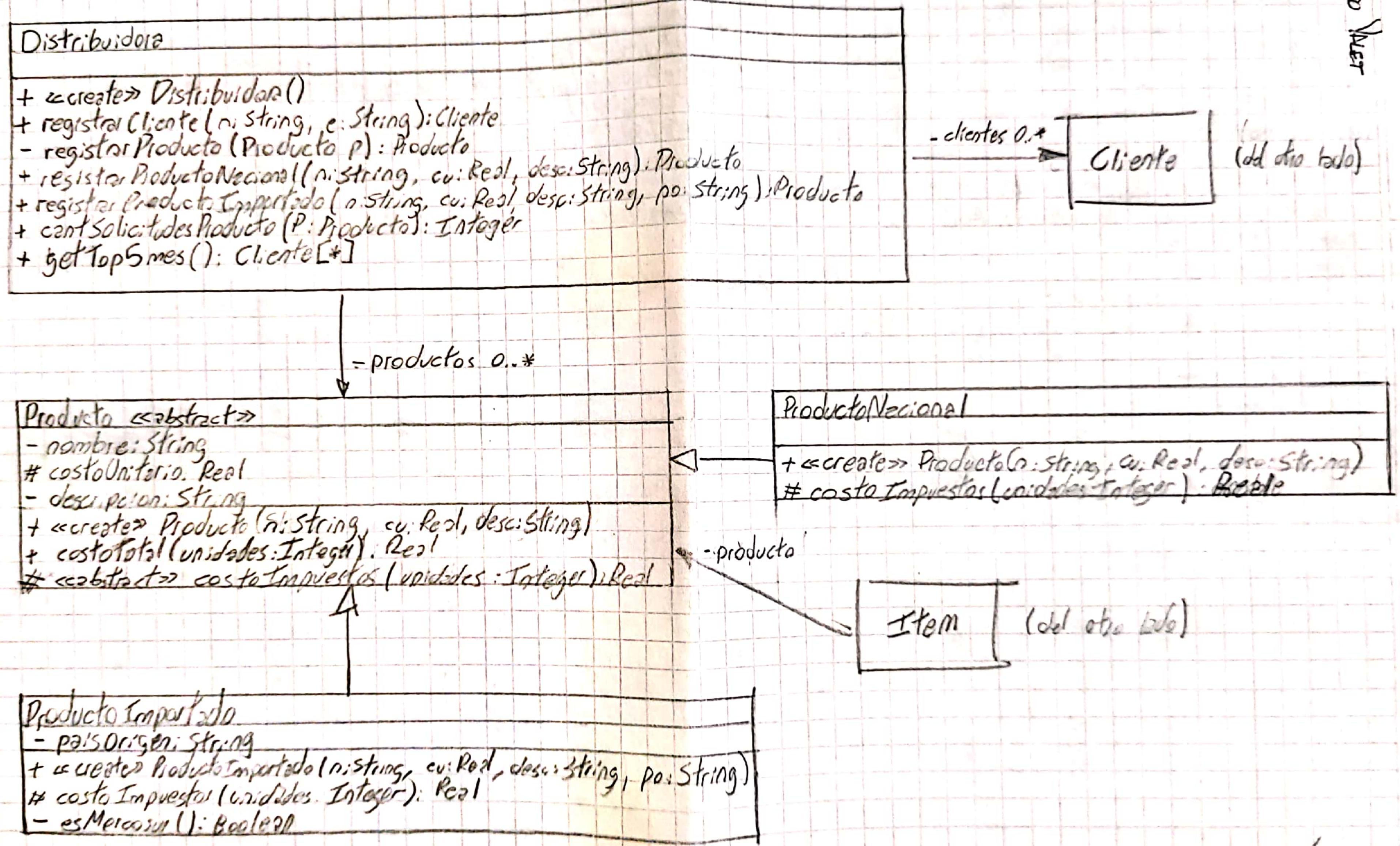
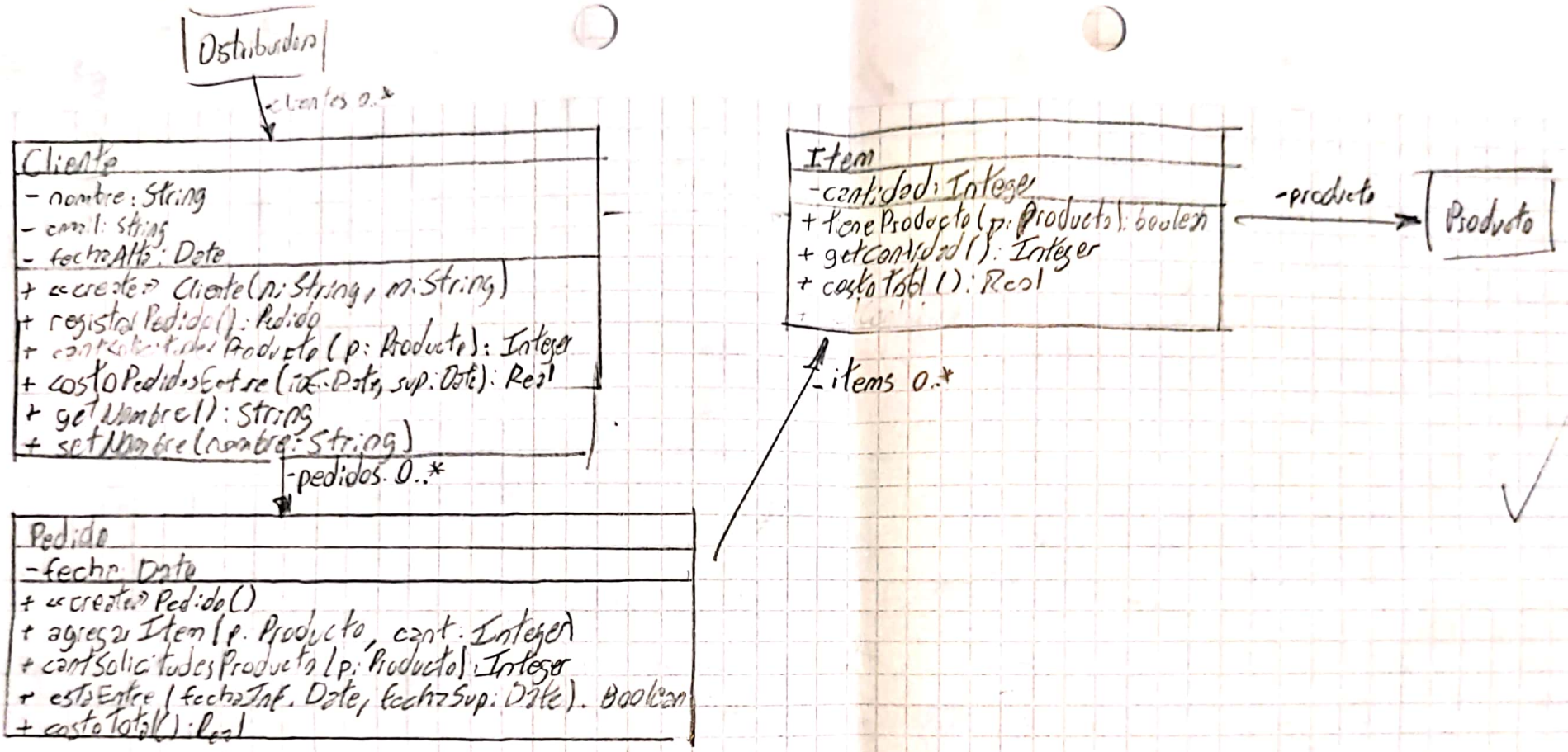
APROBADO

CORRIGIÓ: LEANDRO VALET

Silenzi Manuel 19171/7





Silenzi Manuel 19171/7

FECHA

```
public class Distribuidora {
    private List<Cliente> clientes;
    private List<Producto> productos;

    public Distribuidora() {
        this.clientes = new ArrayList<>();
        this.productos = new ArrayList<>();
    }

                                        NOMBRE      EMAIL
    public Cliente registrarCliente(String n, String e) {
        Cliente cli = new Cliente(n, e);
        this.clientes.add(cli);
        return cli;
    }

    private Producto registrarProducto(Producto p) {
        this.productos.add(p);
        return p;
    }

    public Producto registrarProductoNacional(String n, double cu, String desc) {
        return this.registrarProducto(new ProductoNacional(n, cu, desc));
    }

    public Producto registrarProductoImportado(String n, double cu, String desc,
                                               String po) {
        return this.registrarProducto(new ProductoImportado(n, cu, desc, po));
    }

    public int cantSolicitudesProducto(Producto p);
        return this.clientes.stream()
                .mapToInt(c -> c.cantSolicitudesProducto(p))
                .sum();
    }

    public List<Cliente> getTop5mes() {
        LocalDate act = LocalDate.now();
        LocalDate ant = LocalDate.of(act.getYear(), act.getMonth(), act.getDay()-30);
        return this.clientes.stream()
                .sort((c1, c2) -> c2.costoPedidosEntre(ant, act) -
                                  c1.costoPedidosEnte(ant, act))
                .limit(5)
                .collect(Collectors.toList());
    }
}
```

VERSIÓN SIMPLE:
LocalDate.now().minusDays(30)

Asumo que funciona, no se

NOTA

```
public class Cliente {
    private String nombre, email;    private List<Pedido> pedidos;
    private LocalDate fechaAlta;

    public Cliente (String n, String e) {
        this.nombre = n;
        this.email = e;
        this.fechaAlta = LocalDate.now();
        this.pedidos = new ArrayList<>();
    }

    public Pedido registrarPedido() {
        Pedido p = new Pedido()
        this.pedidos.add(p);
        return p;
    }

    public int getCantSolicitudesProducto(Producto p) {
        return this.pedidos.stream()
                    .mapToInt(pedido -> pedido.getCantSolicitudesProducto(p))
                    .sum();
    }

    public double costoPedidosEntre(LocalDate inf, LocalDate sup) {
        return this.pedidos.stream()
                    .filter(p -> p.estaEntre(inf, sup))
                    .mapToDouble(Pedido::costoTotal)
                    .sum();
    }

    public String setNombre() {
        return this.nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

LA VERSIÓN LAMBDA ES MÁS BONITA

Silenzi Manuel 191717

```
public class Pedido {
    private LocalDate fecha;
    private List<Item> items;

    public Pedido () {
        this.fecha = LocalDate.now();
        this.items = new ArrayList<>();
    }

    public void agregarItem(Producto p, int cant) {
        items.add(new Item(p, cant));
    }

    public int getCantSolicitudesProducto(Producto p) {
        return this.items.stream()
                .filter(i -> i.tieneProducto(p))
                .mapToInt(i -> i.getCantidad())
                .sum();
    }

    public boolean estaEntre(LocalDate fechaInf, LocalDate fechaSup){
        // Devuelve true si la fecha del pedido esta entre la fechaInf y
        // fechaSup, false en caso contrario
        return !this.fecha.isBetween(fechaInf, fechaSup)
    }

    public double costoTotal() {
        return this.items.stream()
                .mapToDouble(Item::costoTotal)
                .sum();
    }
}
```

SE IMPLEMENTA CON:
.isAFTER(DATE)
.isBEFORE(DATE)

```
public class Item {
    private int cantidad;
    private Producto producto;

    public boolean tieneProducto(Producto p) {
        return this.producto.equals(p);
    }

    public int getCantidad() {
        return this.cantidad;
    }

    public double costoTotal()
        return this.producto.costoTotal(this.cantidad) * this.cantidad;
    }
}
```

FALTÓ CONSTRUCTOR

Silenzi Manuel 19171/7

```
public abstract class Producto {
    private String nombre;
protected ~~private~~ double costoUnitario;
    private String descripcion;

    public Producto(String n, double cu, String desc) {
        this.nombre = n;
        this.costoUnitario = cu;
        this.descripcion = desc;
    }

    public double costoTotal(int unidades){
        return this.costoUnitario + this.costoImpuestos(unidades);
    }

    protected abstract double costoImpuestos(int unidades);
}

public class ProductoNacional extends Producto {
    public ProductoNacional(String n, double cu, String desc){
        super(n, cu, desc);
    }

    @override
    protected double costoImpuestos(int unidades){
        if (unidades > 10) return 0;
        return this.costoUnitario * 0,05;
    }
}

public class ProductoImportado extends Producto {
    private String paisOrigen;

    public ProductoImportado(String n, double cu, String desc, String po) {
        super(n, cu, desc);
        this.paisOrigen = po;
    }

    @override
    protected double costoImpuestos(int unidades){
        if (this.enMercosur() && unidades > 50){
            return this.costoUnitario * 0,08;
        }
        return this.costoUnitario * 0,21;
    }

    private boolean enMercosur(){
        String[] paises = {"Brasil", "Paraguay", "Uruguay"};
        return paises.contains(this.paisOrigen);
    }
```

APLICA PARA LISTAS Y NO ARRAYS

NOTA

Silenzi Manuel 19171/7

Cliente :: costoPedidosEntre:
- no hay pedidos
- ningún pedido está dentro del periodo
- al menos un pedido se encuentra en el periodo ✓

Pedido:: esteEntre
- creo que no es necesario testearlo ya que sólo utilizo un método provisto por el lenguaje, pero si lo testeara verificaría los casos límite: está en el límite inferior y está en el límite superior, en caso que esté fuera de rango

CUÁL SERÍA EL TEST?

Pedido:: costoTotal
- no hay ítems
- hay al menos un ítem ✓

Producto:: costoTotal
- un producto nacional con 10 unidades ✓
- " " 11 unidades
- un producto importado del mercosur con 50 unidades ✓
- " " más de 50 unidades
- un producto importado que no es del mercosur ✓

NOTA

```
Distribuidora distribuidora = new Distribuidora();
distribuidora.registrarProductoNacional("Torgito", 250.0, "Alestar");
Producto alestar =
Cliente pepe = new Cliente("Pepi", "p@mail.com");
Pedido p = pepe.registrarPedido();
p.agregarItem(alestar, 5);
```