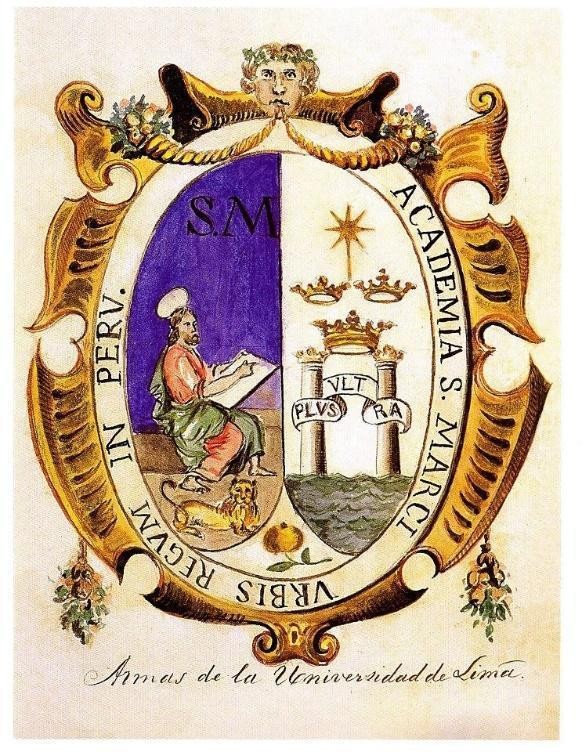
# UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

(Universidad del Perú, DECANA DE AMÉRICA)

# FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMATICA

**ESCUELA DE INGENIERÍA DE SOFTWARE**



**PATRONES DE DISEÑO**

# PRÁCTICA 01

**FATIMA MARUSIA PEVES LARA**

**SEMESTRE 2021-2**

**Patrones de Diseño Práctica 01**

**Principios de diseño de clases SRP, OCP y LSP**

1. Realice el diagrama de clases de la versión 1.0 de la tienda virtual y el diagrama de clases de la versión 1.1. Haga una comparación de ambos diagramas.

Diagrama de clases v1.0

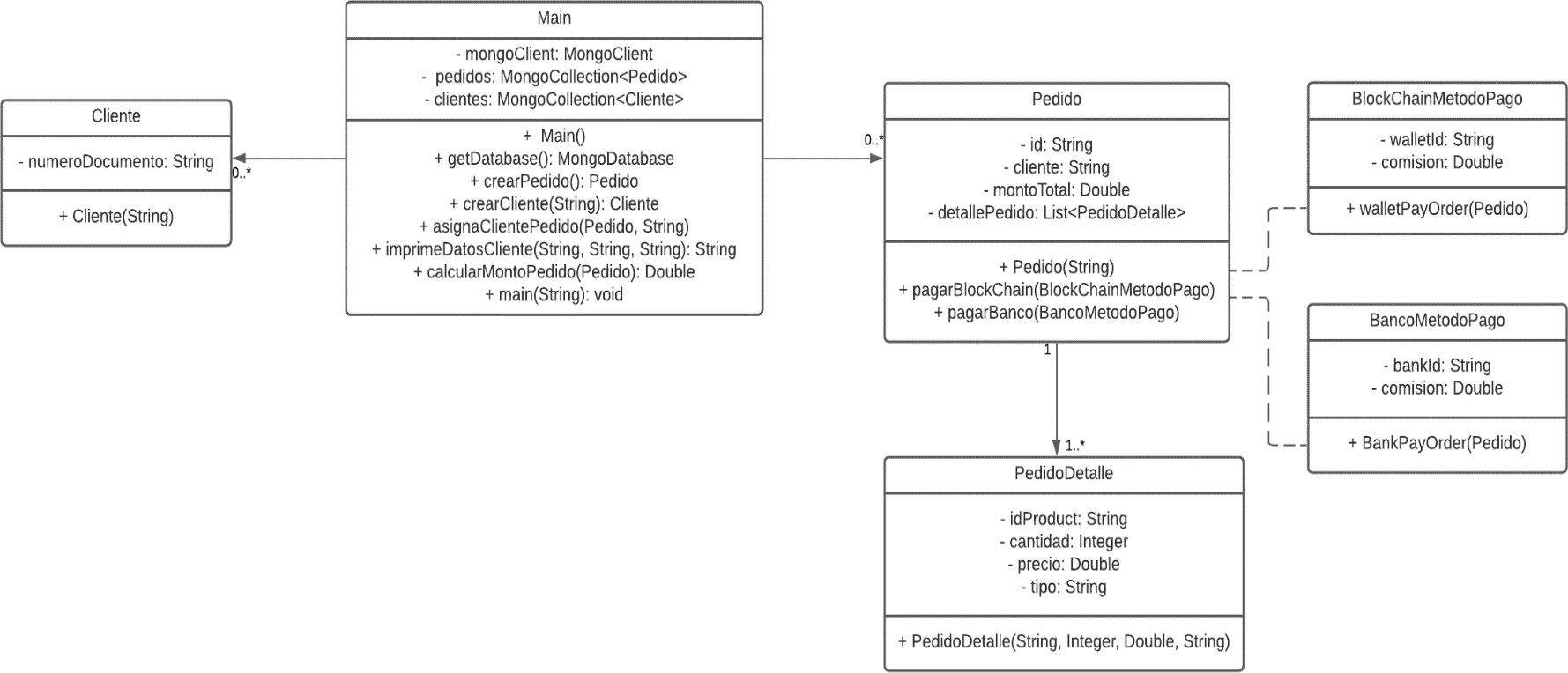
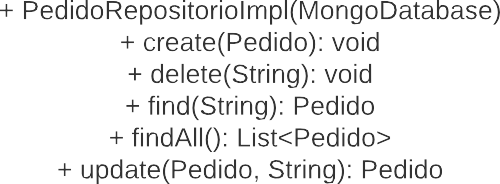
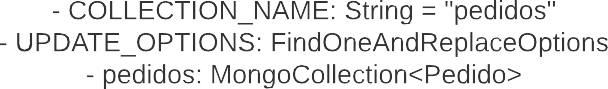
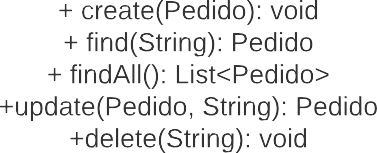
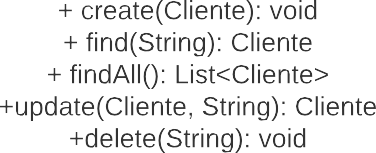
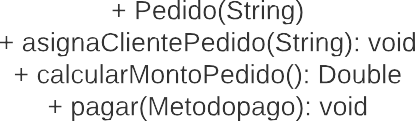
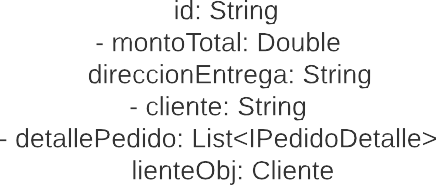
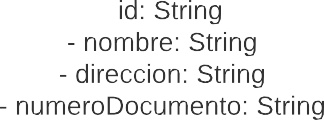
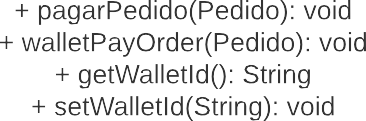
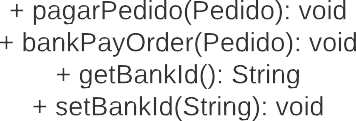
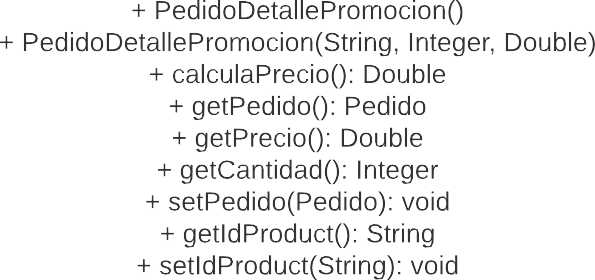
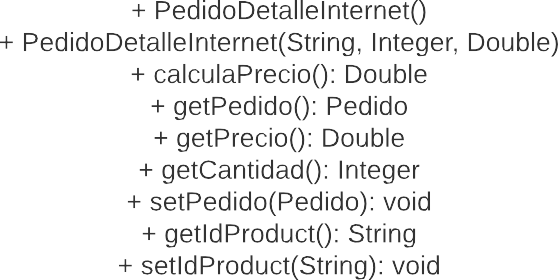
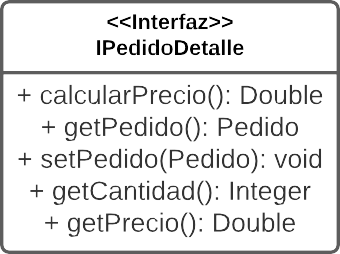
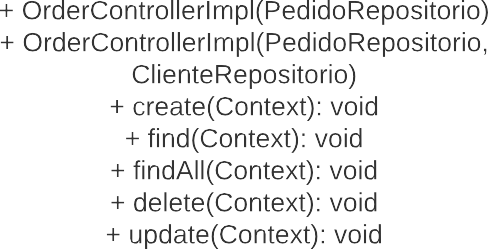
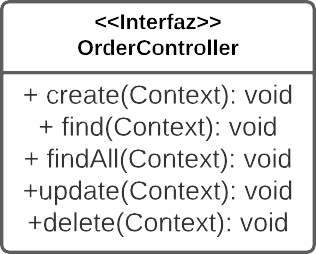
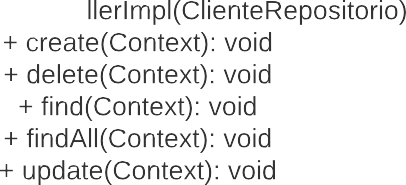
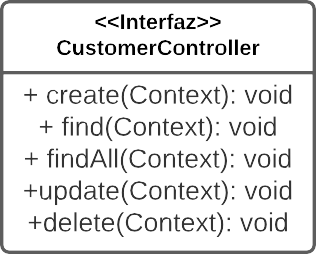


Diagrama de clases v1.1



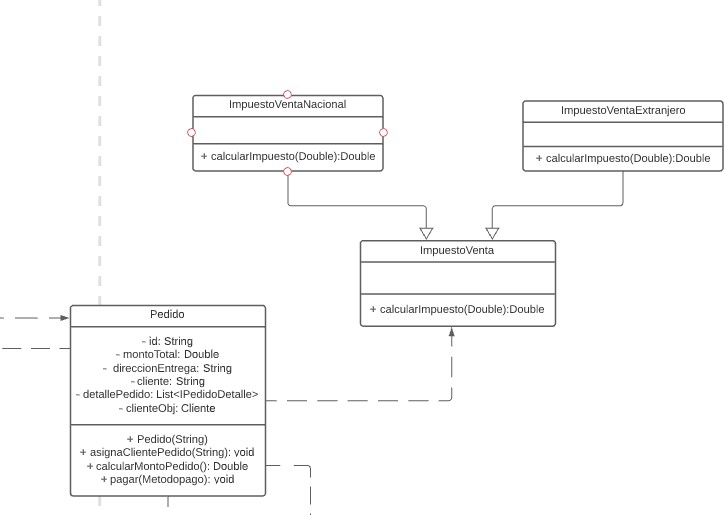
Como podemos observar en el diagrama de clases de la v1.0, la clase Main, es una especie de clase ‘Dios’, pues hace de todo y por ello tan solo contamos con 6 clases para esta versión. Más cuando aplicamos los principios de diseño, y dividimos por funciones, creando nuevas clases y extendiendo otras, nos da como resultado 14 nuevas clases más 5 interfaces. Puede verse muy extenso, pero todo ello es con el

propósito de mejorar la organización del código y hacer más fácil su mantenimiento.

1. Agregue al código los métodos para el cálculo de impuestos de la venta, considere el cálculo de impuesto nacional y extranjero. ¿Qué principio aplicaría?

Aplicaremos el principio de Responsabilidad Única, por lo que se creará una clase padre ‘Impuesto Venta’ de donde heredarán las clases para calcular el impuesto nacional y el impuesto extranjero.

En el diagrama v1.1



Clase ImpuestoVenta

package tienda.models; public class ImpuestoVenta {

public ImpuestoVenta(){

}

public Double calcularImpuesto(Double total){ return null;

}

}

Clase ImpuestoVentaNacional

package tienda.models;

public class ImpuestoVentaNacional extends ImpuestoVenta{ public ImpuestoVentaNacional(){

}

public Double calcularImpuesto(Double total){ Double impuesto;

impuesto = 0.18 \* total;

return impuesto;

}

}

Clase ImpuestoVentaExtranjero

package tienda.models;

/\*\*

\* ImpuestoVentaExtranjero

\*/

public class ImpuestoVentaExtranjero extends ImpuestoVenta{ public ImpuestoVentaExtranjero(){

}

public Double calcularImpuesto(Double total){

Double impuesto; impuesto = 0.25 \* total;

return impuesto;

}

}

Clase Pedido

public Double calcularMontoPedido() {

List<IPedidoDetalle> detallePedido = this.getDetallePedido(); ImpuestoVenta imp = new ImpuestoVenta();

Double total = 0.0;

for (IPedidoDetalle item : detallePedido) { total += item.getPrecio();

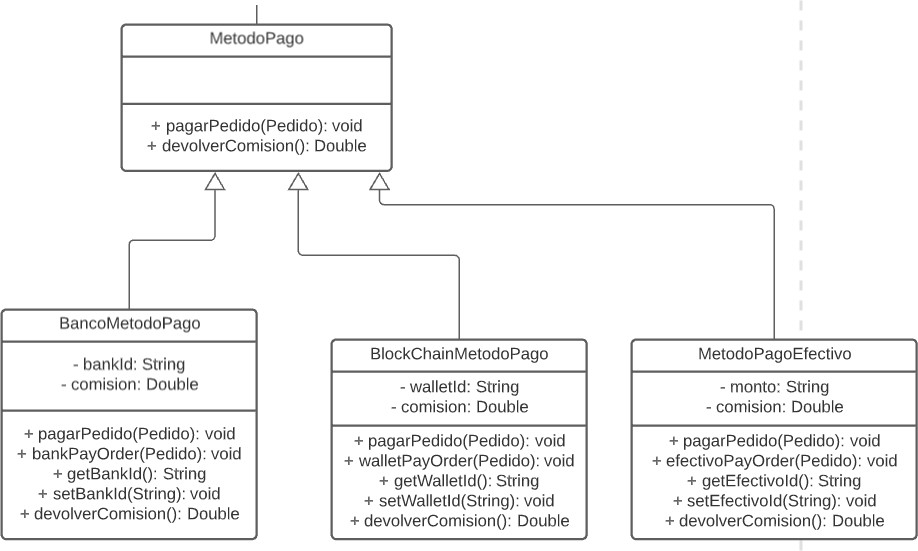
}

total = total + imp.calcularImpuesto(total); return total;

}

1. Implemente las clases de Producto, Categoría y Familia, donde una familia tiene muchas categorías y una categoría tiene muchos productos. Implemente un método de impresión de los datos del Producto respetando el principio de abierto cerrado y de sustitución de Liskov.
2. Desarrolle un nuevo Método de Pago, pero considere agregar también el método devolverComision() para todos los métodos. Realice el diagrama de clases.

En el diagrama v1.1



Clase YapeMetodoPago

package tienda.models;

public class YapeMetodoPago extends MetodoPago { private String cuentaYape;

private Double comision;

@Override

public void pagarPedido(Pedido order){ yapePayOrder(order);

}

public void yapePayOrder(Pedido order){ comision = order.getMontoTotal() \* 0.05;

System.out.println("Procesando el pago con Yape: "+ getCuentaYape()+"

| total: "+order.getMontoTotal() + " comision: " +comision);

}

public String getCuentaYape() { if(this.cuentaYape == null){

setCuentaYape("999888123");

}

return cuentaYape;

}

public void setCuentaYape(String ctaYape){ cuentaYape= ctaYape;

}

public Double devolverComision(){ return comision;

}

}

1. Agregue una interfaz gráfica para crear Clientes. La interfaz puede ser en web o escritorio, según crea conveniente.