*fechaExportacion: date, nombreEmpresa: string | Pedido

Empresa

+darDeAltaPedido(destino: string, fechaDeExportacion: Date, nombreEmpresa: string): Pedido
+facturarMayorCostoFinal(fechaInicio: Date, fechaFin: Date): Factura
+facturar(fechaExportacion: Date, costoBasico: real, pedido: Pedido): Factura

1 listadoPedidos *

1 listadoFacturas *

Pedido

-destino: String
-fechaExportacion: Date
-nombreEmpresa: String
+<<create>>createPedido(destino: String, fechaExportacion: Date, nombreEmpresa: String)
+agregarBien(descripcion: String, unidades: integer, pesoUnitario: real, valorUnitario: real)
+agregarServicio(descripcion: String, costoMateriales: real, costoManoObra: real)

Factura

-fechaFacturacion: Date
-fechaExportacion: Date
-costoBasico: real
-costoExportacion: real
-costoFinal: real
+<<create>>createFactura(fechaExportacion: Date, costoBasico: real, pedido: Pedido): factura
+estarEnPeriodo(periodo: DateLapse)

1 items *

«abstract» Item

-descripcion: String
«abstract»+«create»createItem(descripcion: String)
«abstract»+calcularCostoBásico(): real

Bien

-unidades: integer
-pesoUnitario: real
-valorUnitario: real
+«create»createBien(unidades: integer, pesoUnitario: real, valorUnitario: real): Bien
+calcularCostoBasico(): real

Servicio

-costoMateriales: real
-costoManoObra: real
+«create»createServicio(costoMateriales: real, costoManoObra: real): servicio
+calcularCostoBasico(): real

```
public class Empresa {
    Private List<Pedido> listadoPedidos;
    Private List<Factura> listadoFacturas;

    Public Empresa() {
        this.listadoPedidos = new ArrayList<>();
        this.listadoFacturas = new ArrayList<>();
    }

    Public Pedido darDeAltaPedido(String destino, LocalDate fechaDeExportacion,
    String nombreEmpresa) {
        Pedido agregar = new Pedido(destino, fechaDe
        Exportacion, nombreEmpresa);
        this.listadoPedidos.add(agregar);
        return agregar;
    }

    Public Factura facturar(LocalDate fecha
    Exportacion, double costoBasico, Pedido
    Facturarse){
        Factura agregar = new Factura(fecha
        Exportacion, costoBasico, facturarse);
        this.listadoFacturas.add(agregar);
        return agregar;
    }

    Public Factura facturaMayorCostoFinal(
    LocalDate fechaInicio, LocalDate, fecha
    Fin){
        DateLapse periodo = new DateLapse(fecha
        Inicio, fechaFin);
        List<Factura> filtrado = listadoFacturas.
        stream.filter(check -> check.estaEnPeriodo
        (periodo)).collect(Collectors.toList);
        return filtrado.stream.max((fac1, fac2) ->
        Double.compare(fac1.getFinal,
        fac2.getFinal())).orElse(null);
    }
}
```

8/8

```
Public class ExamenTest {
    Private [illegible] emp;
    [illegible] private Pedido pedLleno;
    Private Pedido pedVacio;
    Private Factura test;
@BeforeEach
Public Void setUp(){
    [illegible]
    PedidoVacio = new Pedido("a", LocalDate
    .now(), "Aj");
    [illegible] test = new Factura(PedidoVacio.
    getFechaExportacion, PedidoVacio.
    Total);
    Pedido con Items = new Pedido("b",
    LocalDate.now(), "z");
    PedidoconItems
```

Yoequis

4/8

```
Public abstrac class Item {
    protected String descripcion;

    public Item (String descripcion){
        this.descripcion = descripcion;
    }

    Public abstrac double calcularCostoBasico();
}
```

```
Public class Bien extends Iten {
    Private int unidades;
    Private double pesoUnitario;
    Private double valorUnitario;

    Public Bien (String descripcion, int unidades, double pesoUnitario, double
    valorUnitario) {
        super (descripcion);
        this.unidades = unidades;
        this.pesoUnitario = pesoUnitario;
        this.valorUnitario = valorUnitario;
    }

    Public double calcularCostaBasico () {
        int output = unidades * valorUnitario;
        if ((pesoUnitario * unidades) > 1000) {
            return output * 1,1;
        } else {
            return output;
        }
}
```

Ejercicio 6/8

```
Public class Servicio extends Item {
    Private double CostoMateriales;
    private double costoManoObra;

    Public Servicio(String descripcion,
    double CostoMateriales, double
    CostoManoObra) {
        Super(descripcion);
        this.costoMateriales = costoMateri
        es;
        this.costoManoObra=costoManoObra;
    }

    Public double calcularCostoBasico() {
        return costoManoObra+costoMateriales;
    }
}
```

```
Public class Factura {
    Private LocalDate fechafacturacion;
    private LocalDate fechaExportacion;
    private ~~costoBasica~~double costoBasico;
    private double costoExportacion;
    Private double costoFinal;
    Private Pedido facturado;

    Public factura (LocalDate fechaExportacio, double
    costoBasico, Pedido pedido){
        this.fechaFacturacion=LocalDate.
        now();
        this.fechaExportacion= fechaExporta
        cion;
        this.costoBasico = costoBasico;
        This.costoExportacion=costoBasico *
        0,05;
        this.costoFinal = costoBasico+costo
        Exportacion;
        this.facturado = pedido;
    }

    Public boolean estaEnPeriodo (DateElapse
    Periodo){
        return periodo.includesDate(
        fechaFacturacion);

    Public double getFinal (){
        return costoFinal;
    }
}
```

```
Public class Pedido {
    Private String destino;
    Private LocalDate fechaExportacion;
    Private String nombreEmpresa;
    Private List<Item> items;
    Public Pedido(String destino, LocalDate
    fechaExportacion, String nombreEmpresa)
    {
     this.destino = destino;
     this.FechaExportacion = fechaExportacion;
     this.nombreEmpresa = nombreEmpresa;
     this.items = new ArrayList<>();
    }

    Public void agregarBien(String descripcion,
    int unidades, double pesoUnitario, double
    valorUnitario) {
        Bien agregar = new Bien(descripcion,
        unidades, pesoUnitario, valorUnitario);
        this.items.add(agregar);
    }

    Public void agregarServicio(String descripcion,
    double costoMateriales, double costoManoObra)
    {
     Servicio agregar = new Servicio(descripcion,
     costoMateriales, costoManoObra)
     this.items.add(agregar);
    }
    Public double total(){
     return this.items.stream.mapToDouble(
     i -> i.calcularCostoBasico).sum();
    }
```