Pedido De Exportacion
- destino: String
- fecha Exportacion: LocalDate
- nombre Empresa: String
+ agregar Producto (nuevo: Producto)
+ get Factura (): Factura

pedidos
0..

Factura
- fecha Factura: LocalDate
- fecha Exportacion: LocalDate
- costo Basico: real
- costo Exportacion: real
- costo total: real
+ esta Comprando Unico (inicio: LocalDate, fin: LocalDate)
: boolean

facturas
0..*

Esquema Simplificado De Exportacion
+ ... Pedido (destino: String, fecha Exportacion: LocalDate, nombre Empresa: String, productos <Producto>): Pedido De Exportacion
+ agregar Bien (pedido: Pedido Exportacion, descripcion: String, unidades: integer, peso Unitario: real, volumen Unitario: real)
+ agregar Servicio (pedido: Pedido Exportacion, descripcion: String, costo Materiales: real, costo Mano Obra: real)
+ obtener Factura Mas Cara (inicio: LocalDate, fin: LocalDate): Factura

productos
0..*

Bien
- cant Unidades: integer
- peso Unitario: real
- vol Unitario: real

** Producto**
- descripcion: String
+ get Costo Basico (): real

Servicio
- costo Mano: real
- costo Materiales: real

```java
public class EsquemaSimplificadoDeExportacion {
    private List <PedidoDeExportacion> pedidos;
    private List< Factura > Facturas;
public EsquemaSimplificadoDeExportacion () {
    this.pedidos= new ArrayList <PedidoDeExportacion> ();
    this.facturas = new ArrayList <Factura> ();
}

public PedidoDeExportacion dorAltoPedido (... String destino, LocalDate fechaExportacion, String nombreEmpresa,
    List < Producto > productos ) {
    PedidoDeExportacion nuevo = new PedidoDeExportacion (destino, fechaExportacion, nombreEmpresa, productos);
    this.pedidos. add ( nuevo );
    return nuevo;
}

public void agregarBien ( PedidoDeExportacion pedido, String descripcion, int unidades, double pesoUnitario,
    double valorUnitario ) {
    Producto nuevo = new Bien(descripcion, pesoUnitario, valorUnitario, unidades);
    pedido. agregarProducto(nuevo);
}

public void agregarServicio ( PedidoDeExportacion pedido, String descripcion, double costoMaterial, double costoManoObra ) {
    Producto nuevo = new Servicio (descripcion, costoManoObra, costoMaterial );
    pedido. agregarProducto (nuevo);
}

public Factura getFactura ( PedidoDeExportacion pedido );
    Factura nuevo = pedido. getFactura ();
    this.facturas. add ( nuevo );
    return nuevo;
}

public Factura obtenerFacturaMasCara ( LocalDate inicio, LocalDate fin ) {

    List <Factura> facturasFiltradas= this.facturas.stream(). filter (facturas -> facturas.esta Comprendido (
        inicio, fin )) . collect (Collectors. toList());

    Factura max = facturasFiltradas.stream(). max (Factura f1, Factura f2 -> Double.compare (f1.getCostoTotal()
        f2. getCostoTotal()) ). get();
    return max;
}

}
```

```java
public class Factura () {
    private LocalDate fechaFacturacion, fechaExportacion;
    private double costoBasico, costoExportacion, costoFinal;
    private List<Producto> productos;

    public Factura( LocalDate fechaExportacion, double costoBasico, double costoExportacion, double costoFinal,
    List <producto> productos) {
    this. fechaExportacion= fechaExportacion;
    this. fechaFacturacion= LocalDate.now();
    this. costoBasico = costoBasico;
    this. costoExportacion= costoExportacion;
    this. costoFinal= costoFinal;
    }

    public double get. CostoTotal() {
        return this. costoFinal;
    }

    public boolean estaComprendido ( LocalDate inicio, LocalTime fin) {
        return ( this.getFechaExportacion(). isAfter (inicio)) && ( this.getFechaExportacion (). isBefore ((fin));
    }

    public LocalDate getFechaExportacion () {
        return this. fechaExportacion;
    }

}
```

```java
public class PedidoDeExportacion {
    private String destino;
    private LocalDate fechaExportacion;
    private String nombreEmpresa;
    private List<Producto> productos;

    public PedidoDeExportacion (String destino, LocalDate fechaExportacion, String nombreEmpresa, List<producto> productos) {
        this.destino = destino;
        this.fechaExportacion = fechaExportacion;
        this.nombreEmpresa = nombreEmpresa;
        this.productos = productos;
    }

    public void agregarProducto (Producto nuevo) {
        this.productos.add(nuevo);
    }

    public Factura getFactura () {
        Factura nuevo = new Factura (getFechaExportacion(), costoBasico(), getCostoExportacion, getCostoFinal);
        return nuevo;
    }

    private double getCostoBasico () {
        double total = this.productos.stream().mapToDouble(Producto :: getCostoBasico).sum();
        return total;
    }

    private double getCostoExportacion () {
        return this.getCostoBasico * 0,05;
    }

    private double getCostoFinal () {
        return this.getCostoBasico + this.getCostoExportacion();
    }
}
```

```
public abstract class Producto {
    private String descripcion;

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

public abstract double getCostoBasico();

}
```

```java
public class Bien extends Producto {
    private int cantUnidades;
    private double pesoUnitario;
    private double valorUnitario;

    public Bien (string descripcion, double pesoUnitario, double valorUnitario, int cant) {
        super (descripcion);
        this.cantUnidades = cant;
        this.pesoUnitario = pesoUnitario;
        this.valorUnitario = valorUnitario;
    }

    public double getCostoBasico () {
        double peso = this.getPesoUnitario() * this.getCant();
        double valor = this.getValorUnitario() * this.getCant();

        if (peso >= 100) {
            return valor + valorFord;
        }
        return valor;
    }

    public double getPesoUnitario () {
        return this.pesoUnitario;
    }
    public int getCant() {
        return this.cantUnidades;
    }
    public double getValorUnitario () {
        return this.valorUnitario;
    }

}
```

```java
public class Servicio extends Producto {
    private double costoManoObra;
    private double costoMateriales;

    public Servicio (String descripcion, double manoObra, double materiales) {
        super (descripcion);
        this.costoManoObra= manoObra;
        this.costoMateriales= materiales;
    }

    public double getCostoBasico () {
        return this.getCostoManoObra() + this.getCostoMateriales();
    }

    public double getCostoManoObra () {
        return this.costoManoObra;
    }

    public double getCostoMateriales () {
        return this.costoMateriales;
    }

}
```

```java
public class TestFactura {
    private Factura factura1, factura2;
    private Bien bien1;
    private Servicio servicio1;
    private List <Producto> productos1, productos2;

    @BeforeEach
    public void setUp(){

        this.bien1= new Bien ("celulares", 100, 10000, 8);
        this.servicio1= new Servicio ("medicina", 50000, 1000);
        this.productos1= new ArrayList <Producto> ();
        this.productos1.add (servicio1);
        this.productos1.add( bien1);
        factura1= new Factura (LocalDate.now(), 51000, 2550, 53550, productos1);
        this.productos2= new ArrayList <Producto> ();
        factura2= new Factura (LocalDate.now(), 0, 0, 0, productos2);

    }
    @Test
    public void testListaVacia(){
                assertEquals (0, this.factura2.getCostoTotal());
    }
    @Test
    public void testListaConProductos () {
        assertEquals (53550, this.factura1.getCostoTotal());
    }

}
```