PARTE I: 6 puntos

Se plantea desarrollar un programa Java que permita la **gestión de una empresa agroalimentaria** que trabaja con tres tipos de productos:

- Productos frescos (ProductoFresco).
- Productos refrigerados (ProductoRefrigerado).
- Productos congelados (ProductoCongelado).

Todos los productos llevan esta información común:

- **Fecha de caducidad:** tipo LocalDate (fechaCaducidad)
- **Número de lote:** tipo entero (numLote)

A su vez, **cada tipo de producto lleva alguna información específica.**

- Los productos frescos deben llevar el paisOrigen (cadena de texto).
- Los productos refrigerados deben llevar el codigoSupervision (cadena de texto).
- Los productos congelados deben llevar la temperaturaCongelacion (valor de tipo entero).

Implementa el modelo con la jerarquía de clases que consideres adecuado.

Se proporciona una clase **ProductoDAO** para obtener una colección dinámica de Productos. Crea los constructores necesarios para que desaparezcan los errores de la clase DaoProducto.

En la clase principal obtén el listado de productos y muestra su información por consola de la siguiente manera:

```
Producto [fechaCaducidad=2025-02-14, numeroLote=3]-> ProductoCongelado [temperaturaCongelacion=-20]
Producto [fechaCaducidad=2024-02-16, numeroLote=6]-> ProductoFresco [paisOrigen=España]
Producto [fechaCaducidad=2025-02-13, numeroLote=5]-> ProductoCongelado [temperaturaCongelacion=5]
Producto [fechaCaducidad=2024-03-13, numeroLote=1]-> ProductoRefrigerado [codigoSupervision=A123]
Producto [fechaCaducidad=2024-02-16, numeroLote=2]-> ProductoFresco [paisOrigen=Marruecos]
Producto [fechaCaducidad=2024-03-13, numeroLote=4]-> ProductoRefrigerado [codigoSupervision=1234]
```

[1 punto] ⇒ Clase Producto según especificaciones (atributo/s, constructor/es).
[1 punto] ⇒ Clase ProductoFresco según especificaciones (atributo/s, constructor/es).
[1 punto] ⇒ Clase ProductoRefrigerado según especificaciones (atributo/s, constructor/es).
[1 punto] ⇒ Clase ProductoCongelado según especificaciones (atributo/s, constructor/es).
[1 punto] ⇒ Jerarquía correcta. Métodos necesarios.
[1 punto] ⇒ Se muestra el listado de todos los productos acorde la captura.

PARTE II: amplía el modelo con métodos abstractos. 4 puntos

Ten en cuenta que todos los productos, una vez creados se deben validar.

Usa, donde consideres, el siguiente método abstracto para completar esta parte del ejercicio:

```
public abstract boolean validar();
```

Ten en cuenta que, la forma de validarse cada tipo de producto es diferente. Se consideran productos no válidos si:

- **ProductoCongelado:** si la temperatura de congelación es superior o igual a 0º centígrados.
- **ProductoFresco:** si el país no se encuentra en la lista de países permitidos (España, Portugal, Francia e Italia).
- **ProductoRefrigerado**: si el códigoSupervision no cumple con el patrón de empezar con una letra mayúscula seguido de tres dígitos, por ejemplo B345;

Al ejecutar el programa debes obtener el listado de productos no válidos:

```
Productos no válidos:
Producto [fechaCaducidad=2025-02-13, numeroLote=5]-> ProductoCongelado [temperaturaCongelacion=5]
Producto [fechaCaducidad=2024-02-16, numeroLote=2]-> ProductoFresco [paisOrigen=Marruecos]
Producto [fechaCaducidad=2024-03-13, numeroLote=4]-> ProductoRefrigerado [codigoSupervision=1234]
```

[0,5 puntos] ⇒ Implementada la abstracción según especificaciones
[0,5 puntos] ⇒ Validación correcta del productoCongelado.
[1,5 puntos] ⇒ Validación correcta del productoFresco.
[1 punto] ⇒ Validación correcta del productoRefrigerado.
[0,5 puntos] ⇒ Se muestran los no válidos acorde a la captura.