Programación

Enumeraciones e Interfaces **Práctica 4.a**

Una empresa de distribución textil necesita implementar una funcionalidad en su sistema informático a través de la que poder filtrar todas las prendas de vestir con las que trabaja en función de que éstas cumplan o no una determinada condición.

Los datos que tiene registrado por cada prenda de vestir son:

- marca: cadena de caracteres que representa el nombre del fabricante de la prenda
- precio: double
- tipo: representa cuál es el tipo de la prenda de vestir. Los posibles valores son: camisa, pantalón, falda o chaqueta.
- color: representa el color de la prenda de vestir. Sus valores sólo pueden ser: rojo, verde, azul y amarillo.
- talla: las posibles tallas son XS, S, M, L, XL o XXL.

A esta funcionalidad de filtrado se le pasará como parámetro un ArrayList conteniendo todos las prendas disponibles así como otro objeto que represente la Condicion que se utilizará para seleccionar las prendas. Esta funcionalidad devolverá otro ArrayList conteniendo únicamente aquellas prendas para las que se la condición sea cierta.

```
ArrayList<PrendaVestir> filtrar(ArrayList<PredaVestir> prendasVestir, Condicion condicion)
```

Las posibles condiciones serán:

- **`PrecioMayor`**: comprueba si el precio de una prenda es mayor que un valor fijado en el constructor de esta condición.
- **`PrecioIgual`**: comprueba si el precio de una prenda es igual que un valor fijado en el constructor de esta condición.
- **`MarcaIgual`**: comprueba si la marca de la prenda es igual a otra que se ha fijado en el constructor de esta condición.
- **`TipoIgual`**: comprueba si el tipo de la prenda es igual al que se ha fijado en el constructor de esta condición.
- **`ColorIgual`**: comprueba si el color de la prenda es igual al que se ha establecido en el constructor de esta condición
- **`TallaIgual`**: comprueba si la prenda lo es de una talla igual a la fijada en el constructor de esta condición.

Todas estas condiciones se crearán a partir de métodos estáticos definidos en Condicion. Estos métodos tendrán el mismo nombre que el tipo de la condición que vayan a crear y recibirán como parámetro los datos que se exige en el constructor de la misma.

Además de las anteriores, también habrá condiciones Relacionales que permiten establecer relaciones and (condición And), or (condición Or) entre de dos condiciones cualquiera. Por último, también existirá la condición Not cuyo propósito será invertir el resultado de cualquier condición. La creación de las condiciones And, Or y Not se realizará a través de métodos default de Condicion:

- **`Condicion and(Condicion condicion)`**
  Crea y devuelve una condición And que combina la acción de la condición this con la del parámetro
- **`Condicion or(Condicion condicion)`**
  Crea y devuelve una condición Or que combina la acción de la condición this con la del parámetro
- **`Condicion not()`**
  Crea y devuelve una condición Not que invertirá el resultado de la condición this

Ejemplos:

Este ejemplo localiza todas las prendas de vestir que sean faldas, de la marca "Patrizia Pepe", de color verde y cuyo precio sea menor o igual a 150€.

```
ArrayList<PrendaVestir> prendas = new ArrayList<PrendaVestir>();
prendas.add( … );
: : :
Condicion condicion = Condicion.tipoIgual(Tipo.FALDA)
                              .and( Condicion.marcaIgual("Patrizia Pepe") )
                              .and( Condicion.colorIgual( Color.VERDE ) )
                              .and( Condicion.precioMayor(150).not() );

ArrayList<PrendaVestir> prendasFiltradas = Filtrador.filtrarPrendas(prendas, condicion);
```