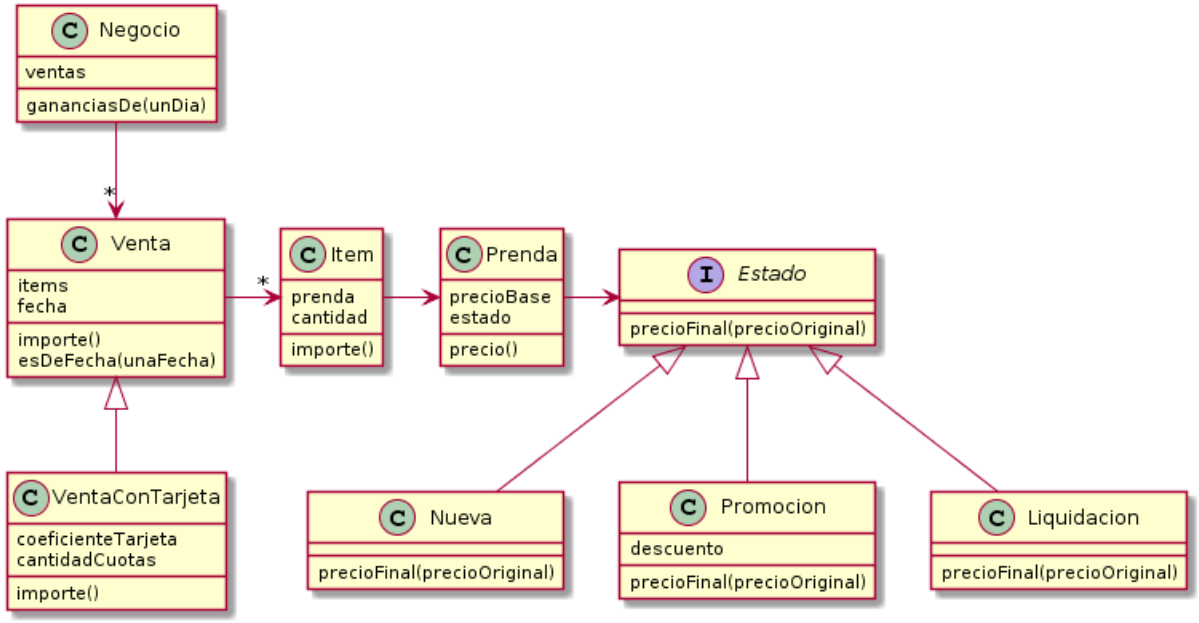
Solución 1

- Se descarta utilizar herencia para los distintos tipos de estado de la Prenda, ya que estos pueden variar con el tiempo.
- Se podría haber utilizado un dictionary/map/hash en la venta para los ítems, pero se decide crear la abstracción Item.




```
clase Prenda
  metodo precio
    retornar estado.precioFinal(precioBase)

clase Nueva
  metodo precioFinal(precioOriginal)
    retornar precioOriginal

clase Promocion
  metodo precioFinal(precioOriginal)
    retornar precioOriginal - descuento

clase Liquidacion
  metodo precioFinal(precioOriginal)
    retornar precioOriginal * 0.5

clase Item
  metodo importe()
```

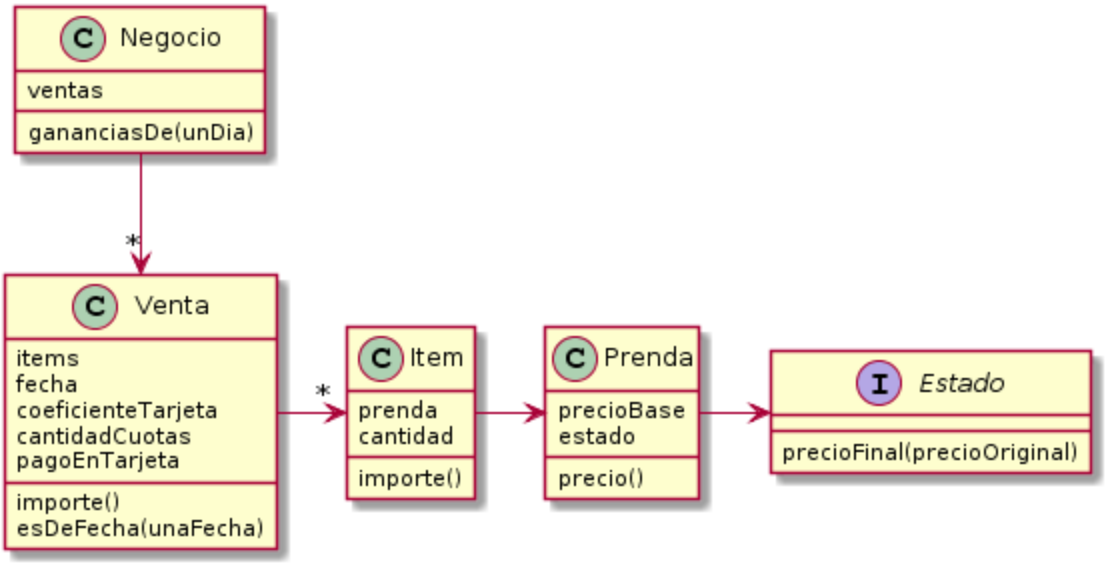
```
    retornar prenda.precio() * cantidad

clase Venta
  metodo importe()
    retornar items.sum(item -> item.importe())

clase  VentaConTarjeta
  metodo importe()
    retornar coeficienteTarjeta * cantidadCuotas + 0.01 *
super() + super()

clase Negocio
  metodo gananciasDe(fecha)
    retornar ventas
     .filter(venta -> venta.esDeFecha(fecha))
     .sum(venta -> venta.importe())
```

Solución 2



Se utilizará de la solución 1, lo siguiente:

- Importe de un ítem
- Importe de una prenda
- Obtener las ganancias de una fecha

En esta opción en vez de utilizar herencia para los dos tipos de ventas que tenemos, utilizaremos un booleano.

```
clase Venta
  metodo importe()
    importe = costoItems()
    if (pagoEnTarjeta)
      importe += importe * 0.01 + cantidadCuotas *
coeficienteTarjeta
    retornar importe
```