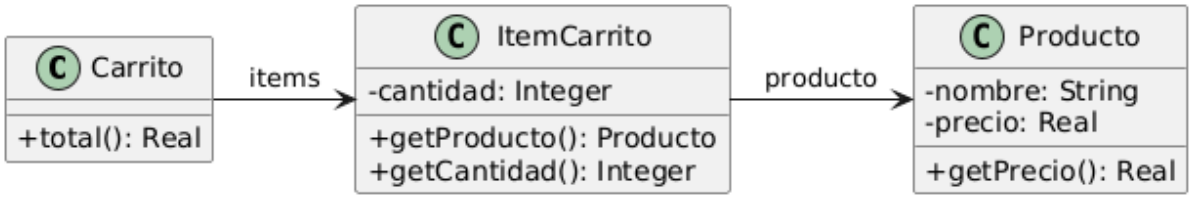
Ejercicio 2.4 Carrito de Compra




```
public class Producto {
    private String nombre;
    private double precio;

    public double getPrecio() {
        return this.precio;
    }
}

public class ItemCarrito {
    private Producto producto;
    private int cantidad;

    public Producto getProducto() {
        return this.producto;
    }

    public int getCantidad() {
        return this.cantidad;
    }
}

public class Carrito {
    private List<ItemCarrito> items;

    public double total() {
            return this.items.stream()
                .mapToDouble(item →
```

```
                    item.getProducto().getPrecio() * item.getCantidad())
                .sum();
    }
}
```

Paso uno: Análisis

- Las tareas están bien repartidas, cada clase se encarga de una tarea diferente y posee los datos y el comportamiento adecuado. El mal olor aparece cuando nos fijamos como está implementado el método **total()**
- El método **total()** de la clase **Carrito** le pide a cada item de su lista que calcule la multiplicación del precio de su producto por su cantidad comprada. El problema que podemos ver acá es que una clase está utilizando los métodos de otra para resolver algo dentro suyo

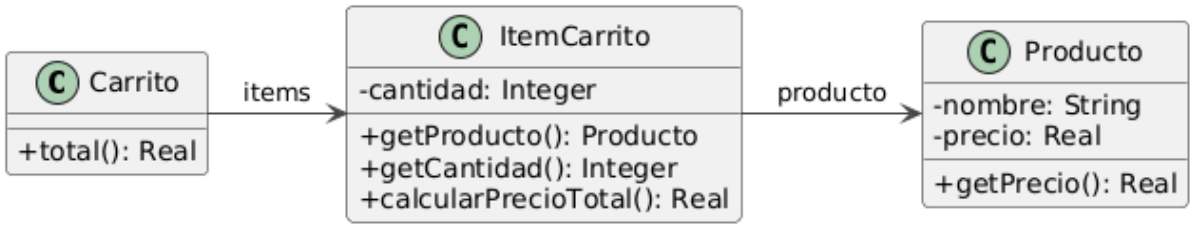
Paso dos: Refactoring

1. **Code Smell:** Envidia de Atributos → El método está invocando métodos de la clase **ItemCarrito** para calcular el precio total de un item
    a. **Refactoring a aplicar:** Extract Function → Sacamos afuera como método aparte la cadena de mensajes

    // Esto lo hago mas que nada para mecanizar el proceso de refactoring, ya sé que podría ahorrarme tener que crear el método en esta clase y después moverlo a la de ItemCarrito

    b. **Refactoring a aplicar:** Move Function → Movemos el método a la clase donde pertenece, **ItemCarrito**

Resultado final



```
public class Producto {
    private String nombre;
    private double precio;

    public double getPrecio() {
        return this.precio;
    }
}

public class ItemCarrito {
    private Producto producto;
    private int cantidad;

    public Producto getProducto() {
        return this.producto;
    }

    public int getCantidad() {
        return this.cantidad;
    }

    public double calcularPrecioTotal() {
        return this.producto.getPrecio() * getCantidad();
    }
}

public class Carrito {
    private List<ItemCarrito> items;

    public double total() {
        return this.items.stream()
                .mapToDouble(item -> item.calcularPrecioTotal()).sum();
    }
}
```