1.- **¿Qué diferencia piensas que hay entre un Error y un Failure?**

La diferencia principal es que un Failure se da al realizar la prueba, es un indicativo de que tu código no pasa el test, por ejemplo assertEquals(1,2) produciría un failure.

Un error es un fallo inesperado que ocurre mientras se está intentando ejecutar el test, por ejemplo cuando salta una excepción.

2.- **Explica por qué este test no es unitario.**

No se trata de un test unitario ya que hace uso de otros módulos.

3.- ¿Qué utilidad crees que tiene realizar una clase de test parametrizada? Pon algún ejemplo.

Para que al realizar un test no sea tedioso y repetitivo probar diferentes casos lo

cual nos lleva a repetir código dentro de nuestros test.

El ejemplo es: ParamShoppingCarTest. Aquí usamos un array de precios para hacer diferentes pruebas.

4.- **Captura las pantallas y adjúntalas al fichero de respuestas. Explica qué son las líneas verdes y rojas, los números y porcentajes que se muestran.**

JaCoCoverage analysis of project "PR4" (powered by JaCoCo from EclEmma)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
pr4		89%		83%	3	20	5	40	1	14	0	3
Total	14 of 127	89%	2 of 12	83%	3	20	5	40	1	14	0	3

pr4

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
ShoppingCart		87%		90%	2	14	4	28	1	9	0	1
Product		93%		50%	1	5	1	10	0	4	0	1
ProductNotFoundException		100%		n/a	0	1	0	2	0	1	0	1
Total	14 of 127	89%	2 of 12	83%	3	20	5	40	1	14	0	3

ShoppingCart

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
imprimeFactura(Writer)		0%		n/a	1	1	4	4	1	1
isEmpty()		88%		50%	1	2	0	1	0	1
getBalance()		100%		100%	0	2	0	7	0	1
findProduct(String)		100%		100%	0	3	0	5	0	1
removeItem(Product)		100%		100%	0	2	0	3	0	1
ShoppingCart()		100%		n/a	0	1	0	3	0	1
addItem(Product)		100%		n/a	0	1	0	2	0	1
empty()		100%		n/a	0	1	0	2	0	1
getItemCount()		100%		n/a	0	1	0	1	0	1
Total	12 of 95	87%	1 of 10	90%	2	14	4	28	1	9

Product

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
equals(Object)		86%		50%	1	2	1	4	0	1
Product(String, double)		100%		n/a	0	1	0	4	0	1
getTitle()		100%		n/a	0	1	0	1	0	1
getPrice()		100%		n/a	0	1	0	1	0	1
Total	2 of 29	93%	1 of 2	50%	1	5	1	10	0	4

ProductNotFoundException

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
ProductNotFoundException()		100%		n/a	0	1	0	2	0	1
Total	0 of 3	100%	0 of 0	n/a	0	1	0	2	0	1

En estas graficas se puede observar el porcentaje cubierto con los test realizados, las líneas/instrucciones que no han sido cubiertas así como las ramas (en condicionales, por ejemplo) que no han sido probadas.

```
    public void empty() {
        _items = new ArrayList<Product>();
    }

    /**
     * Indicates whether the cart is empty.
     * @return true if the cart is empty;
     *     false otherwise.
     */
    public boolean isEmpty() {
        return (_items.size() == 0);
    }

    public void imprimeFactura(Writer writer) throws IOException
    {
        writer.append(_items.toString());
        writer.flush();
        writer.close();
    }

    public boolean findProduct(String s){
        for(Product p: _items){
            if(p.getTitle().equals(s))
                return true;
        }
        return false;
    }
}
```

Esta imagen representa las líneas en rojo que no han sido probadas, en verde las que han sido probadas y en amarillo las que han sido parcialmente probadas.