

PARTE 1: Implementación en JAVA

En este primer apartado he decidido coger como caso de uso para implementar es Añadir productos al Carrito por parte del CLIENTE, implementado sería una clase Carrito.java que contiene lo siguiente:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change
 this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.mavenproject1;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author usuario
 */
public class Carrito
{
    private String usuariold;
    private List<String> productos;
    private double total;

    public Carrito(String usuariold) {
        this.usuariold = usuariold;
        this.productos = new ArrayList<>();
        this.total = 0.0;
    }

    public boolean añadirProducto(String productold, double precio)
    {
        productos.add(productold);
        total += precio;
        return true;
    }

    public String getUsuariold() {
        return usuariold;
    }

    public void setUsuariold(String usuariold) {
        this.usuariold = usuariold;
    }

    public List<String> getProductos() {
```

```

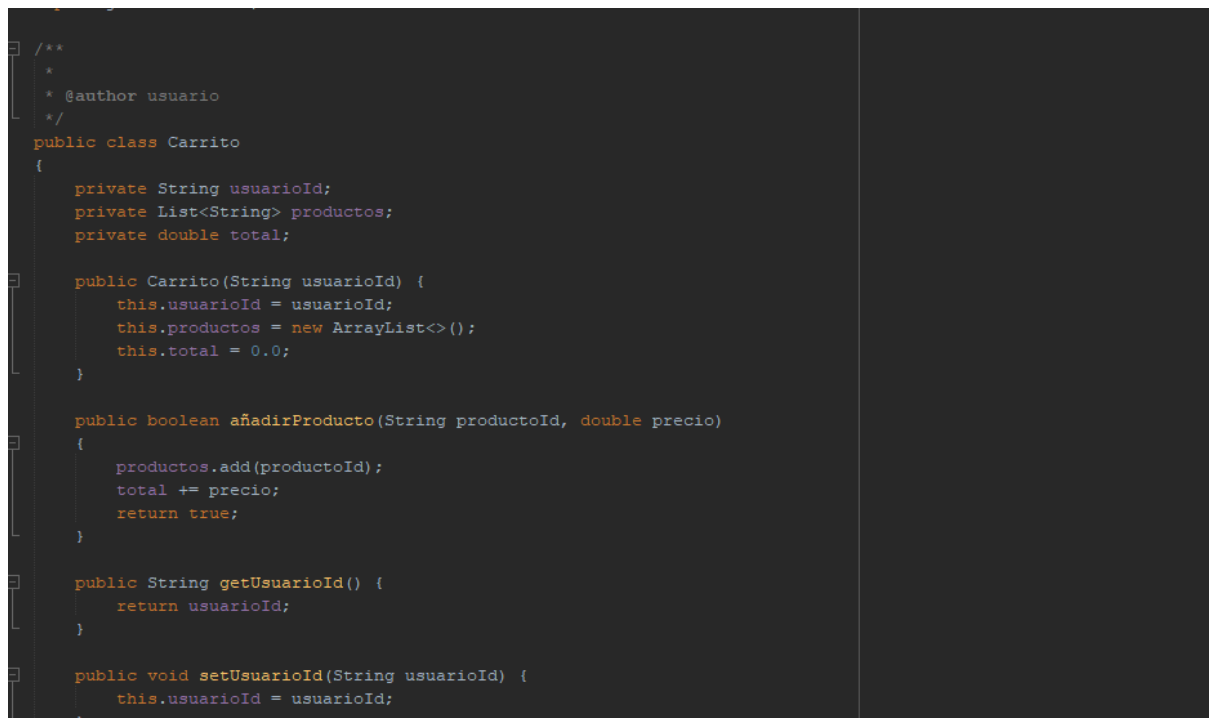
        return productos;
    }

    public void setProductos(List<String> productos) {
        this.productos = productos;
    }

    public double getTotal() {
        return total;
    }

    public void setTotal(double total) {
        this.total = total;
    }
}

```



```

/**
 *
 * @author usuario
 */
public class Carrito
{
    private String usuarioId;
    private List<String> productos;
    private double total;

    public Carrito(String usuarioId) {
        this.usuarioId = usuarioId;
        this.productos = new ArrayList<>();
        this.total = 0.0;
    }

    public boolean añadirProducto(String productoId, double precio)
    {
        productos.add(productoId);
        total += precio;
        return true;
    }

    public String getUsuarioId() {
        return usuarioId;
    }

    public void setUsuarioId(String usuarioId) {
        this.usuarioId = usuarioId;
    }
}

```

PARTE 2: Pruebas

Ahora con la clase de Carrito.java creamos un JUNIT Test que la he llamado CarritoTest, esta es la siguiente:

```

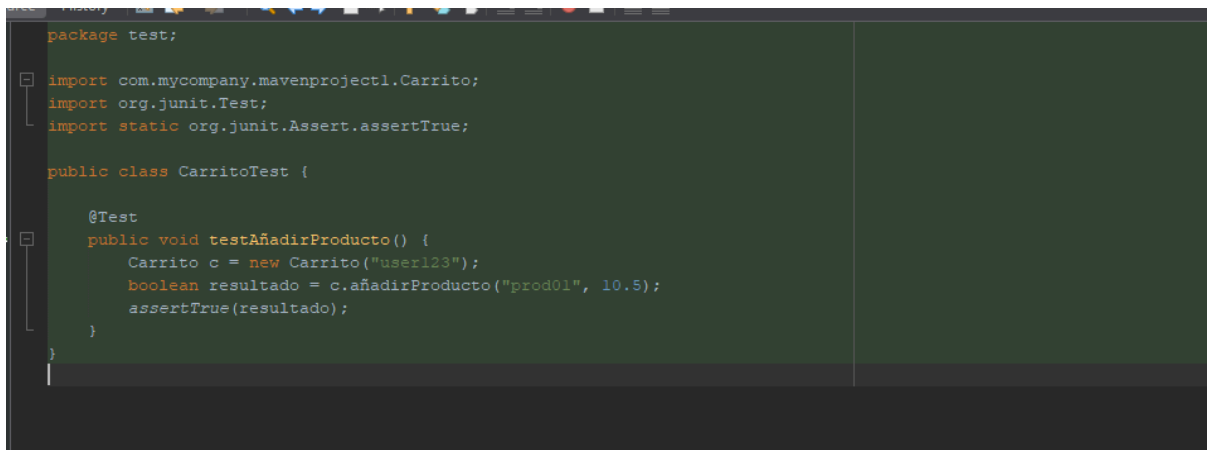
package test;

import com.mycompany.mavenproject1.Carrito;
import org.junit.Test;
import static org.junit.Assert.assertTrue;

public class CarritoTest {

    @Test
    public void testAñadirProducto() {
        Carrito c = new Carrito("user123");
        boolean resultado = c.añadirProducto("prod01", 10.5);
        assertTrue(resultado);
    }
}

```



```

package test;

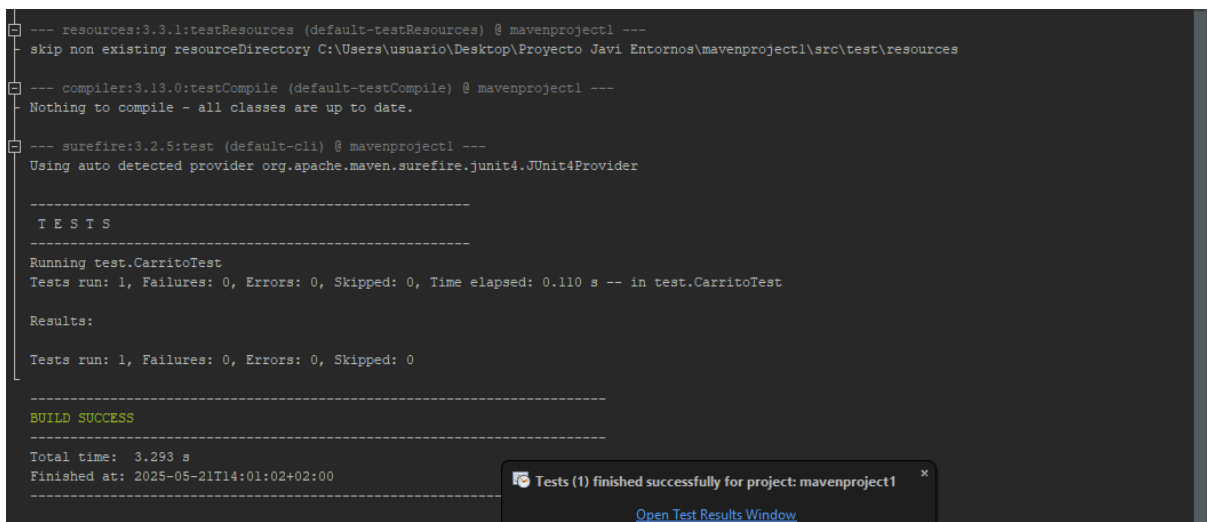
import com.mycompany.mavenproject1.Carrito;
import org.junit.Test;
import static org.junit.Assert.assertTrue;

public class CarritoTest {

    @Test
    public void testAñadirProducto() {
        Carrito c = new Carrito("user123");
        boolean resultado = c.añadirProducto("prod01", 10.5);
        assertTrue(resultado);
    }
}

```

A continuación, he hecho el test correspondiente para probarlo:



```

--- resources:3.3.1:testResources (default-testResources) @ mavenproject1 ---
- skip non existing resourceDirectory C:\Users\usuario\Desktop\Proyecto Javi Entornos\mavenproject1\src\test\resources

--- compiler:3.13.0:testCompile (default-testCompile) @ mavenproject1 ---
- Nothing to compile - all classes are up to date.

--- surefire:3.2.5:test (default-cli) @ mavenproject1 ---
Using auto detected provider org.apache.maven.surefire.junit4.JUnit4Provider

-----
T E S T S
-----
Running test.CarritoTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.110 s -- in test.CarritoTest

Results:

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 3.293 s
Finished at: 2025-05-21T14:01:02+02:00

```

Tests (1) finished successfully for project: mavenproject1

[Open Test Results Window](#)

Ahora para el grafo de flujo podemos descomponer la idea de la siguiente forma:

Nodo 1: Inicio del método.

Nodo 2: Se añade el producto a la lista.

Nodo 3: Se suma el precio al total.

Nodo 4: Se retorna `true`.

Nodo 5: Fin del método.

Su representación gráfica sería la siguiente:



Realmente solo existe un camino básico que es el mencionado anteriormente
Un caso de prueba que podríamos llevar a cabo es el siguiente:

Parámetro	Valor
<code>productoId</code>	"prod01"
<code>precio</code>	10.5
Resultado esperado	<code>true</code>

PARTE 3:SCV

