EJEMPLO PRUEBAS UNITARIAS USANDO JUNIT 5 Y MOCKITO 2

Considere una empresa de eventos que desea contar con un sistema de apoyo a la atención de clientes, en particular administrando los presupuestos de sus clientes.

La Figura 1 muestra parte del diagrama de clases asociado del sistema de interés.

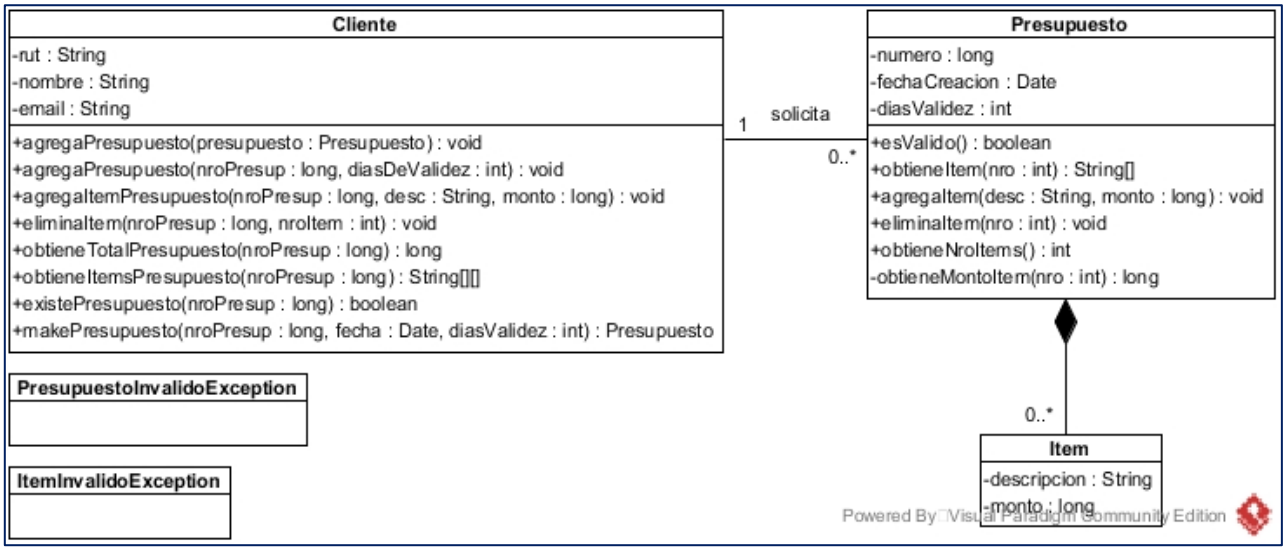

Figura 1. Diagrama de clases Presupuestos con Ítems de un Cliente.

La Figura 2 muestra la estructura del proyecto creado en Eclipse del sistema.

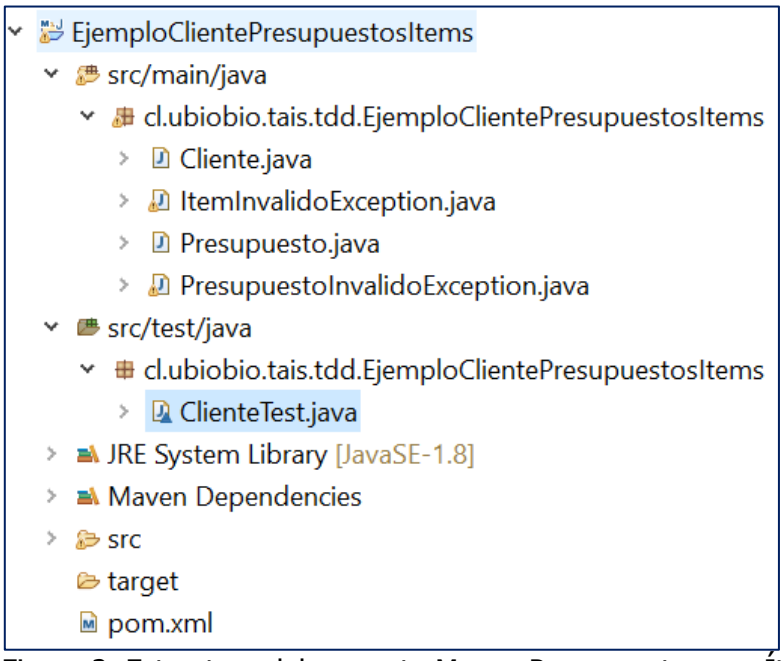

Figura 2. Estructura del proyecto Maven Presupuestos con Ítems de un Cliente.

```
package cl.ubiobio.tais.tdd.EjemploClientePresupuestosItems;

import static org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.mockito.Mockito.when;
import static org.mockito.Mockito.doThrow;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.anyString;
import static org.mockito.Mockito.anyLong;
import static org.mockito.Mockito.anyInt;
import static org.mockito.Mockito.times;
import static org.mockito.Mockito.never;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;



@ExtendWith(MockitoExtension.class)
class ClienteTest {
	private final static String PRESUPUESTO_NO_VIGENTE_EXCEPTION_MSG = "Presupuesto no vigente";
	private final static String PRESUPUESTO_INEXISTENTE_EXCEPTION_MSG = "Presupuesto inexistente";

	@Mock
	Presupuesto presupuesto;

	@InjectMocks
	Cliente cliente;
```

```
@Test    // Ej.1
void SiSeInvocaObtieneTotalPresupuestoYElClientePoseePresupuestoVigenteSeDebeObtenerSuMontoTotal()
                                                                                    throws Exception {
    // Arrange
    long montoObtenido;
    when(presupuesto.estaVigente()).thenReturn(true);
    when(presupuesto.obtieneMontoTotal()).thenReturn((long)250500);

    // Act
    montoObtenido = cliente.obtieneTotalPresupuesto();

    // Assert
    assertEquals(250500,montoObtenido);
}



@Test    // Ej.2
void siSeInvocaObtieneTotalPresupuestoYElClientePoseePresupuestoNoVigenteSeDebeRecibirUnaExcepcion() {
    // Arrange
    Exception exception;
    when(presupuesto.estaVigente()).thenReturn(false);

    // Act + Assert
    exception = assertThrows(PresupuestoInvalidoException.class,
                                                    ()->cliente.obtieneTotalPresupuesto());

    assertEquals(PRESUPUESTO_NO_VIGENTE_EXCEPTION_MSG, exception.getMessage());
}


@Test    // Ej.4
void siSeInvocaAgregaItemPresupuestoYElClientePoseePresupuestoVigenteSeDebePoderAgregarItem()
                                                                                    throws Exception {
    // Arrange
    String descripcion = "Arreglos florales";
    long monto = 95000;
    when(presupuesto.estaVigente()).thenReturn(true);

    // Act
    cliente.agregaItemPresupuesto(descripcion,monto);

    // Assert
    verify(presupuesto).agregaItem(descripcion,monto);
}
```

```
    @Test     // Ej.12
    void siSeInvocaObtieneItemsPresupuestoYElClientePoseePresupuestoVigenteSeDebePoderObtenerSusItems()
                                                                    throws PresupuestoInvalidoException {
        // Arrange
        String[] item1 = {"Arreglos florales", "95000"};
        String[] item2 = {"Arriendo local recepcion", "129000"};
        String[][] items;
        when(presupuesto.estaVigente()).thenReturn(true);
        when(presupuesto.obtieneNroItems()).thenReturn(2);
        when(presupuesto.obtieneItem(anyInt())).thenReturn(item1, item2);

        // Act
        items = cliente.obtieneItemsPresupuesto();

        // Assert
        assertNotNull(items);
        assertAll("items",
                () -> assertEquals("Arreglos florales",items[0][0]),
                () -> assertEquals("95000",items[0][1]),
                () -> assertEquals("Arriendo local recepcion",items[1][0]),
                () -> assertEquals("129000",items[1][1])
                );
        verify(presupuesto).estaVigente();                      // Demasiado?
        verify(presupuesto).obtieneNroItems();                  // Demasiado?
        verify(presupuesto, times(2)).obtieneItem(anyInt());    // Demasiado?
    }
```