

## Práctica de caja blanca y JUNIT

-Programa que vamos a analizar:

```
package ejercicio;
```

```
// Clase que implementa la gestión del fondo de una cuenta bancaria de una persona.
```

```
public class Cuenta {  
    // Atributos.  
    private String nombre;  
    private int saldo;  
    private int saldoCuenta;  
  
    // Métodos constructores.  
    public Cuenta(String nombre, int saldo) {  
        this.nombre=nombre;  
        this.saldo=saldo;  
        this.saldoCuenta=0;  
    }  
  
    // Ingresa dinero indicado en la cuenta, indicando la cantidad a ingresar.  
    public int ingresaFondo(float importe) {  
        if(this.saldo < importe)  
            this.saldoCuenta= saldoCuenta;  
        else {  
            this.saldo-=importe;  
            saldoCuenta+=importe;  
        }  
  
        return saldoCuenta;  
    }  
  
    // Muestra el estado del cliente.  
    public void muestraEstado() {  
        System.out.println("*****");  
        System.out.println("Cliente: "+this.nombre);  
        System.out.printf("Saldo: %d € \n",this.saldo);  
        System.out.println("*****");  
    }  
  
    // Extrae de la cuenta el importe indicado por parámetros.  
    public int sacaFondo(float importe) {
```

```

        if(saldoCuenta < importe)
            saldoCuenta=saldoCuenta;
        else {
            saldoCuenta-=importe;
            saldo+=importe;
        }

        return saldoCuenta;
    }

    //Devuelve el saldo del cliente
    public int getSaldo() {
        return saldo;
    }

    // Muestra el estado de la cuenta
    public void muestraFondo() {
        System.out.println("=====");
        System.out.printf("En la cuenta hay: %d € \n",saldoCuenta);
        System.out.println("=====");
    }
}

```

-Prueba de caja blanca a los métodos necesarios:

Prueba de caja blanca al método ingresaFondo:

```
// Ingresa dinero indicado en la cuenta, indicando la cantidad a ingresar.  
public int ingresaFondo(float importe) {  
    1 if(this.saldo < importe)  
        this.saldoCuenta= saldoCuenta;    2  
    3 else {  
        this.saldo-=importe; } 4  
        saldoCuenta+=importe;  
    }  
    return saldoCuenta; 5  
}
```

Esquema de caja blanca:



Complejidad ciclomática:

$$V(G)=5-5+2=2$$

Caminos:

1, 2, 5

1, 2, 3, 4, 5

Prueba:

Saldo	Cuenta	Importe	Resultado
200	0	150	Saldo=50 Fondo=150
100	0	200	Saldo=100 Fondo=0

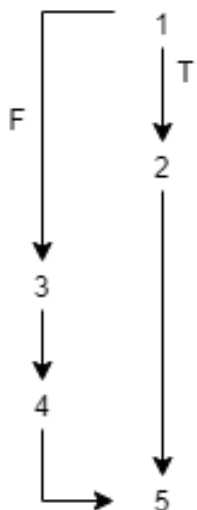
En el primer caso de prueba se realiza el ingreso correctamente.

En el segundo, no se realiza ingreso, ya que, queremos ingresar más dinero del que tenemos.

Prueba de caja blanca al método de sacaFondo:

```
// Extrae de la cuenta el importe indicado por parámetros.
public int sacaFondo(float importe) {
    1 if(saldoCuenta < importe)
        saldoCuenta=saldoCuenta; 2
    3 else {
        saldoCuenta-=importe;
        saldo+=importe; } 4
    }
    return saldoCuenta; 5
}
```

Esquema de caja blanca:



Complejidad ciclomática:

$$V(G)=5-5+2=2$$

Caminos:

1,2,5

1,2,3,4,5

Prueba:

Saldo	Cuenta	Importe	Resultado
10	100	70	Saldo=80 Fondo=30
20	150	200	Saldo=20 Fondo=150

En el primer caso de prueba se realiza la retirada del dinero de la cuenta correctamente.

En el segundo, no se realiza la retirada, ya que, queremos sacar más dinero del que tenemos en la cuenta.

-Realización de prueba con JUNIT:

```
package ejercicio;

import static org.junit.Assert.*;

import org.junit.Test;

public class CuentaTest {

    /*Probamos al cliente Antonio, el cual tiene en su cartera 100 y quiere
    ingresar 120, como no puede ingresar más de lo que tiene pues devuelve, el
    dinero que ya tenga en la cuenta, en este caso 0, ya que la cuenta no tiene
    dinero.*/

    @Test
    public void IngresoIncorrectoTest() {
        Cuenta cli = new Cuenta ("Antonio", 100);
        int resultado = cli.ingresaFondo(120);
        assertEquals("Saldo de la cuenta incorrecta",0, resultado);
    }

    /*Probamos al cliente Manuel, el cual tiene en su cartera 500 y quiere
    ingresar 450, por lo que a el saldo suyo le restamos lo que vamos a
    ingresar y se lo sumamos al saldo de la cuenta, en este caso en la cuenta
    tendría 450.*/

    @Test
    public void IngresoCorrectoTest() {
        Cuenta cli2 = new Cuenta ("Manuel", 500);
        int resultado = cli2.ingresaFondo(450);
        assertEquals("Saldo de la cuenta incorrecta", 450, resultado);
    }

    /*Probamos a la cliente María, el cual tiene en su cartera 500 y quiere
    ingresar primero 100 y luego 300, por lo que a el saldo suyo le restamos lo
    que vamos a ingresar y se lo sumamos al saldo de la cuenta, en este caso en
    la cuenta tendria 400. 100 del primer ingreso y 300 del segundo.*/

    @Test
    public void DosIngresosTest() {
        Cuenta cli3 = new Cuenta ("María", 500);
        cli3.ingresaFondo(100);
        int resultado = cli3.ingresaFondo(300);
        assertEquals("Saldo del cliente incorrecto",400, resultado);
    }
}
```

```
/*Probamos al cliente Rafael, el cual tiene en su cartera 120 y quiere
ingresar primero 100. Como queremos saber cuánto dinero se le quedaría, nos
debería de devolver 20.*/
```

```
@Test
public void saldoClienteTest() {
    Cuenta cli4 = new Cuenta ("Rafael", 120);
    cli4.ingresaFondo(100);
    int resultado = cli4.getSaldo();
    assertEquals("Saldo del cliente incorrecto",20, resultado);
}
```

```
/*Probamos a la cliente Rosalía, el cual tiene en su cartera 200 y quiere
ingresar primero 100 y luego 50. Como queremos saber cuanto dinero se le
quedaría, nos debería de devolver 50.*/
```

```
@Test
public void saldoCliente2Test() {
    Cuenta cli5 = new Cuenta ("Rosalía", 200);
    cli5.ingresaFondo(100);
    cli5.ingresaFondo(50);
    int resultado = cli5.getSaldo();
    assertEquals("Saldo del cliente incorrecto",50, resultado);
}
```

```
/*Probamos al cliente Marisa, el cual tiene en su cartera 200 y quiere
ingresar primero 100 y luego 50. Luego vamos a sacar 20. Cómo queremos
saber cuánto dinero se le quedaría en la cuenta, nos debería de devolver
130.*/
```

```
@Test
public void sacaCuentaTest() {
    Cuenta cli6 = new Cuenta ("Marisa", 200);
    cli6.ingresaFondo(100);
    cli6.ingresaFondo(50);
    int resultado = cli6.sacaFondo(20);
    assertEquals("Saldo de la cuenta incorrecto",130, resultado);
}
```

```
/*Probamos al cliente Dolores, el cual tiene en su cartera 200 y quiere
ingresar primero 100 y luego 50. Luego vamos a sacar 20. Como queremos
saber cuánto dinero se le quedaría a la persona, nos debería de devolver
70.*/
```

```

@Test
public void sacaCuenta2Test() {
    Cuenta cli7 = new Cuenta ("Dolores", 200);
    cli7.ingresaFondo(100);
    cli7.ingresaFondo(50);
    cli7.sacaFondo(20);
    int resultado = cli7.getSaldo();
    assertEquals("Saldo del cliente incorrecto",70, resultado);
}

```

/\*Probamos al cliente Luis, el cual tiene en su cartera 200 y quiere ingresar primero 100 y luego 50. Luego vamos a sacar 300, que es más de lo que tiene en el banco, por lo que no se va a producir la extracción y va a devolver lo que hay en el banco.\*/

```

@Test
public void sacaCuenta3Test() {
    Cuenta cli8 = new Cuenta ("Luis", 200);
    cli8.ingresaFondo(100);
    cli8.ingresaFondo(50);
    int resultado = cli8.sacaFondo(300);
    assertEquals("Saldo del cliente incorrecto",150, resultado);
}

}

```


-Resultados de las pruebas:









Finished after 0,039 seconds

---

Runs: 8/8    ❌ Errors: 0    ❌ Failures: 0

---

▼  ejercicio.CuentaTest [Runner: JUnit 4] (0,004 s)

-  saldoClienteTest (0,001 s)
-  DosIngresosTest (0,000 s)
-  IngresoCorrectoTest (0,001 s)
-  IngresoIncorrectoTest (0,000 s)
-  saldoCliente2Test (0,000 s)
-  sacaCuentaTest (0,000 s)
-  sacaCuenta2Test (0,000 s)
-  sacaCuenta3Test (0,002 s)