TAREA UNIDAD 3
Diseño y realización de pruebas

CFGS DAW. Módulo: Entorno de desarrollo (2023/24)

Dayro Morales Cruz

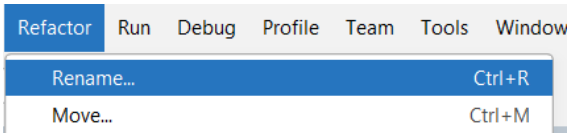
Indice

Ejercicio 1

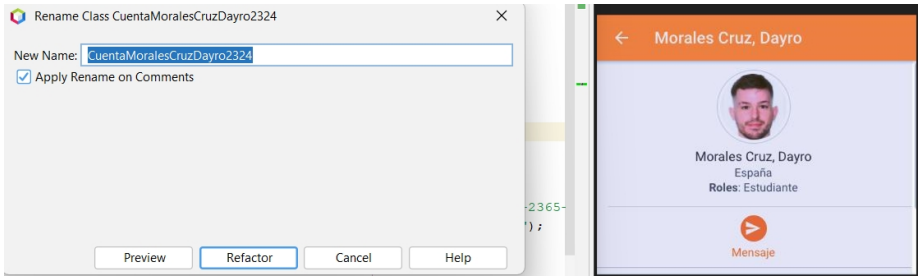
Renombra el **proyecto, la clase** y la **variable** con tu nombre cambiando XXX por **Apellido1Apellido2Nombre2324** en el caso de la clase. Deberá quedar algo así:
CuentaBancaria**Apellido1Apellido2Nombre2324**
Cuenta**Apellido1Apellido2Nombre2324**
miCuenta**Apellido1Apellido2Nombre2324**

Para cambiar el nombre del proyecto **CuentaBancariaXXX** basta con hacer click sobre él con el botón derecho y pulsando en Re*name* y cambiamos por **CuentaBancariaMoralesCruzDayro2324.**

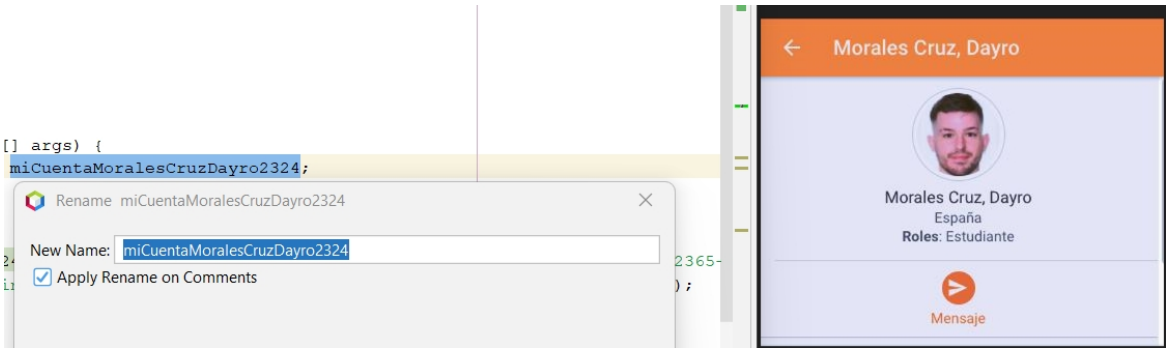
Para cambiar la clase **Cuenta** y la variable **miCuenta** tendremos que refactorizar seleccionando nuestra variable o clase y en la pestafa *Refactor/Rename* para cambiarlo en todo los códigos del proyecto.:



Abrimos nuestro archivo main, seleccionamos nuestra clase **CuentaXXX** y cambiamos el nombre por **CuentaMoralesCruzDayro2324:**



Ahora seleccionamos nuestra variable **miCuentaXXX** por **miCuentaMoralesCruzDayro2324** del mismo modo que cambiamos la clase:



Ejercicio 2

Realiza una ejecución paso a paso, que verifique el correcto funcionamiento de la aplicación.
Para ello establece dos puntos de ruptura, uno en cada método a ejecutar (retirar e ingresar).

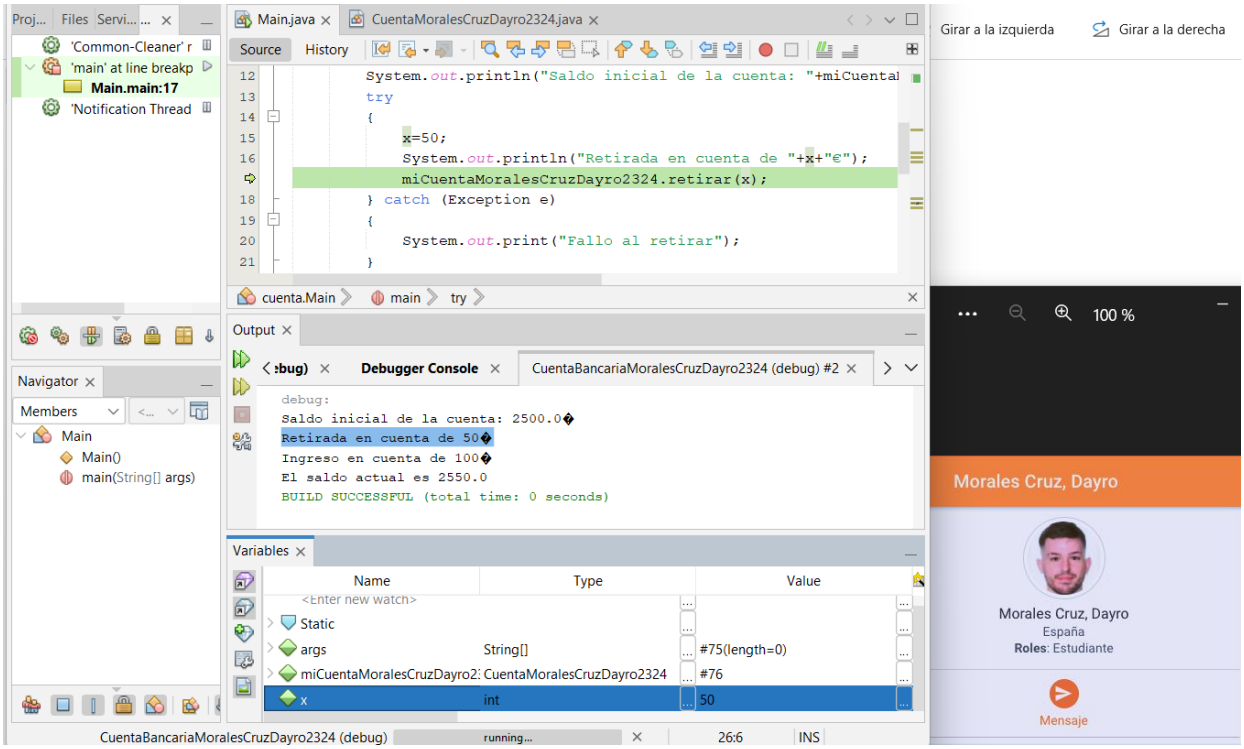
Indica los valores que marca la inspección de variables tras ejecutar las instrucciones en el método main:
miCuentaApellido1Apellido2Nombre2324.retirar(50);
miCuentaApellido1Apellido2Nombre2324.ingresar(25);

Copia en un documento de texto 2 capturas de pantalla en las que se visualicen el valor de las variables después de la llamada a cada método. Es decir, una en la que se visualice el valor de la variable saldo (en el inspector de variables) después de pasar por el método retirar y otra captura de igual modo tras pasar por el método ingresar.

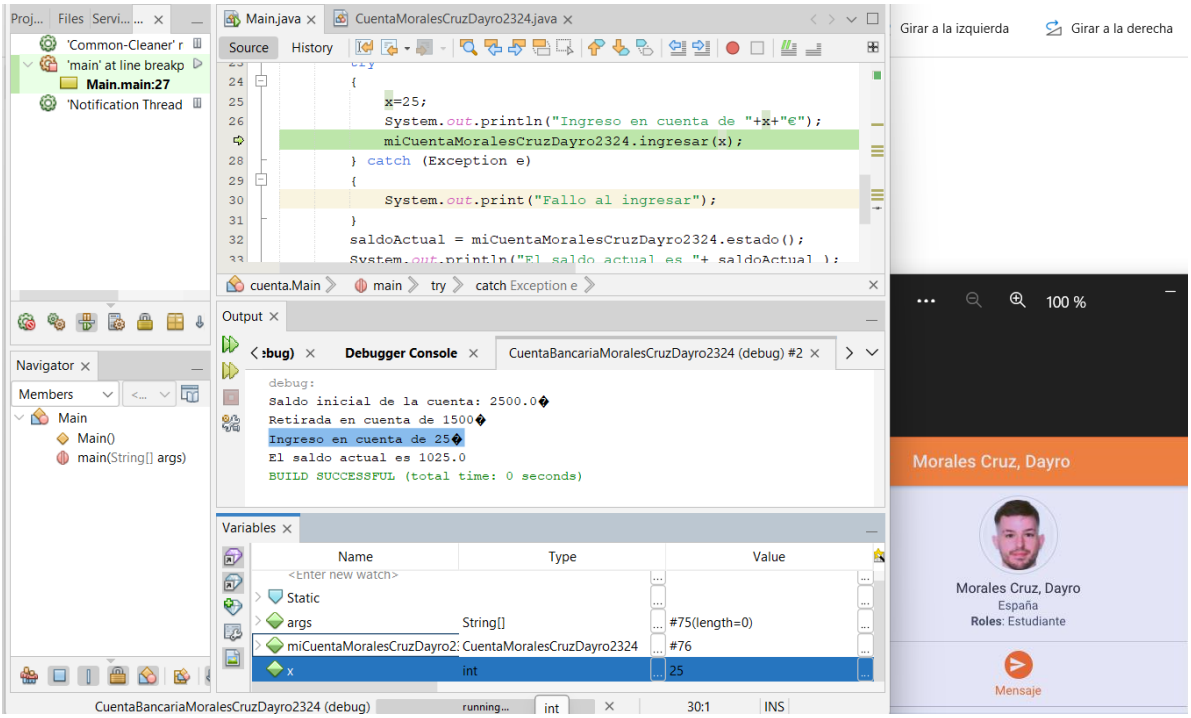
PLAN DE PRUEBAS

Los siguientes ejercicios deben hacerse para una instancia con los siguientes valores:
CuentaXXX("XXX","ES55-3058-2365-8522-1234-5678",2500,0);

Aqui mostramos modificando los datos por un retiro de saldo de 50 quedando el resultado final de 2550. Mostramos la variable modificada. Creamos el punto de ruptura pulsando sobre la linea 15 en su margen izquierdo.



Aqui mostramos modificando los datos por un ingreso de saldo de 25 quedando el resultado final de 1025. Mostramos la variable modificada



Ejercicio 3

Diseña los casos de prueba que permitan verificar el método retirar con un valor límite superior válido.

Método retirar					
Parámetros de entrada	Reglas	Clases válidas	Valores limites válidos	Clases inválidas	Valores límite inválidos
double cantidad	**· La cantidad a retirar no debe exceder al saldo**	**· cantidad <= saldo inicial**	**· cantidad = saldo=2500**	**· saldo inicial < cantidad**	**· saldo inicial +1 = saldo +1**

Casos de prueba para datos inválidos	
testRetirar_ValidoLimiteSuperior()	**Saldo inicial**

Caso de prueba para limite superior válido

```
@Test
    public void testRetirar_ValidoLimiteSuperior() throws Exception {
        System.out.println("Test de valor VÁLIDO LÍMITE SUPERIOR: 
retirar(2500).");
        double cantidad = 2500;
        CuentaMoralesCruzDayro2324 miCuenta = new 
CuentaMoralesCruzDayro2324("Antonio Sánchez", "ES55-3058-2365-8522-1234-5678", 
2500, 0);
        try {
            miCuenta.retirar(cantidad);
            double saldoActual = miCuenta.estado();
            assertTrue(saldoActual == 0);
        } catch (Exception e) {
            System.out.println(e);
            fail("Excepción no esperada: " + e);
        }
    }
```

Ejercicio 4

Asimismo debes diseñar los casos de prueba necesarios para comprobar si se obtiene la salida deseada cuando se introducen valores no válidos en ese mismo método (se debe diseñar un caso de prueba para cada tipo de valor no válido posible).

Copia todos los casos de prueba en el documento de texto.

Método retirar					
Parámetros de entrada	Reglas	Clases válidas	Valores limites válidos	Clases inválidas	Valores límite inválidos
double cantidad	**· No 0 · negativo · No mayor al saldo inicial**	**· 0< · cantidad <= saldo inicial**	**· 0 · 1 · saldo inicial -1 · saldo inicial**	**· cantidad <= 0 · saldo inicial < cantidad · no sea un número**	**· -1 · saldo inicial +1 · 'abc'**

Casos de prueba para datos inválidos	
testRetirar_InvalidoLimiteSuperior()	**Saldo inicial +1**
testRetirar_InvalidoLimiteInferior()	**-1**
testRetirar_NoNumerico()	**'abc'**

Caso de prueba para limite superior inválido

```
@Test
public void testRetirar_InvalidoLimiteSuperior() throws Exception {
    System.out.println("Test de valor INVÁLIDO LÍMITE SUPERIOR: retirar(saldo
inicial + 1).");
    double cantidad = 2501;
    CuentaMoralesCruzDayro2324 miCuenta = new
CuentaMoralesCruzDayro2324("Antonio Sánchez","ES55-3058-2365-8522-1234-
5678",2500,0);
    try {
        miCuenta.retirar(cantidad); // Esto debería lanzar una excepción
        fail("Se permitió una retirada superior al saldo inicial.");
    } catch (Exception e) {
        System.out.println(e);
        double saldoActual = miCuenta.estado();
        assertTrue(saldoActual == 2500);
    }
}
```

Caso de prueba para limite inferior inválido

```
@Test
public void testRetirar_InvalidoLimiteInferior() throws Exception {
    System.out.println("Test de valor INVÁLIDO LÍMITE INFERIOR: retirar(-1).");
    double cantidad = -1;
    CuentaMoralesCruzDayro2324 miCuenta = new
CuentaMoralesCruzDayro2324("Antonio Sánchez","ES55-3058-2365-8522-1234-
5678",2500,0);
    try {
        miCuenta.retirar(cantidad); // Esto debería lanzar una excepción
        fail("Se permitió una retirada negativa.");
    } catch (Exception e) {
        System.out.println(e);
        double saldoActual = miCuenta.estado();
        assertTrue(saldoActual == 2500);
    }
}
```

Caso de prueba para no númericos inválido

```
@Test
public void testRetirar_NoNumerico() throws Exception {
    System.out.println("Test de valor NO NUMÉRICO: retirar('abc').");
    String cantidad = "abc";
    CuentaMoralesCruzDayro2324 miCuenta = new
CuentaMoralesCruzDayro2324("Antonio Sánchez","ES55-3058-2365-8522-1234-
5678",2500,0);
    try {
        miCuenta.retirar(Double.parseDouble(cantidad)); // Esto debería lanzar
una excepción
        fail("Se permitió una retirada no numérica.");
    } catch (NumberFormatException e) {
        System.out.println(e);
        double saldoActual = miCuenta.estado();
        assertTrue(saldoActual == 2500);
    } catch (Exception e) {
        System.out.println(e);
        fail("Excepción no esperada: " + e);
    }
}
```

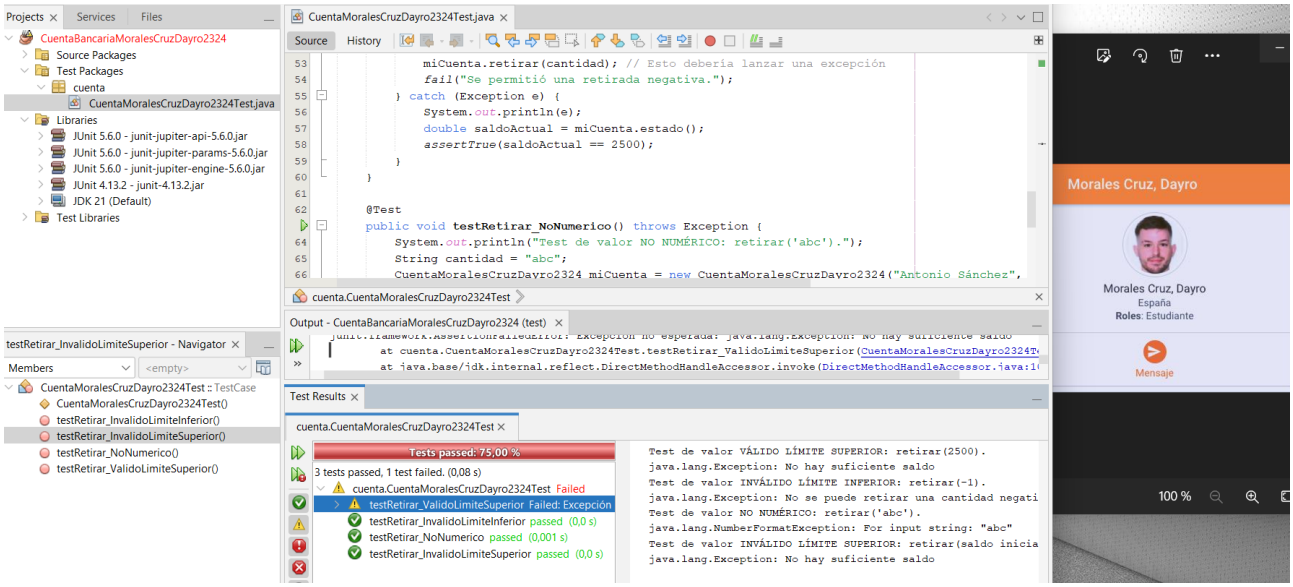
Ejercicio 5

Ejecuta las pruebas y comenta el resultado de cada una.

- **Copia en el documento de texto una captura con el resultado obtenido tras la ejecución de todas las pruebas.**
- **Argumenta en dicho documento el significado de la salida obtenida tras la ejecución de las pruebas y explica a qué se debe el resultado obtenido.**

En el <u>proyecto de Netbeans</u> solo deben aparecer los casos de prueba solicitados en la tarea:

En la siguiente captura podemos ver el proceso de los test. En Nuestro caso falló **1** de **4**



Pruebas de valor LÍMITE SUPERIOR VÁLIDO:
La prueba se supera cuando es posible efectuar una retirada con un valor que esté dentro del rango entre 0 y 2500 coinciendo la retirada con el saldo actual dando el saldo restante correctamente.

- **No se ha superado**

- El caso de prueba para el valor límite superior (2500).:

```
Testcase: testRetirar_ValidoLimiteSuperior(cuenta.CuentaMoralesCruzDayro2324Test):    FAILED
Excepción no esperada: java.lang.Exception: No hay suficiente saldo
junit.framework.AssertionFailedError: Excepción no esperada: java.lang.Exception: No hay suficiente saldo
        at cuenta.CuentaMoralesCruzDayro2324Test.testRetirar_ValidoLimiteSuperior(CuentaMoralesCruzDayro2324Test.java:27)
        at java.base/jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccessor.java:103)
```

“No hay saldo suficiente”
Observamos que se trata de un mensaje incoherente, pues el valor introducido es igual al saldo actual.
Esto demuestra que el método presenta algún error, pues ha devuelto una salida diferente a la esperada.

Si miramos dentro del código, identificamos el siguiente error:

```
if (estado()<= cantidad)
 throw new Exception ("No hay suficiente saldo");
```

Se podría arreglar, al igual que el fallo anterior eliminando el =. (estado() < cantidad):

```
if (estado()< cantidad)
 throw new Exception ("No hay suficiente saldo");
```

Pruebas de valor INVÁLIDO:
Las prueba se superan cuando se lanza el mensaje de excepción correcto del método o el propio programa impide introducir el valor dicho.

- **Se han superado**

- El caso de prueba para los valores límite superior inválido. La salida del test permite comprobar, en cada caso, que el saldo final coincide con el esperado. Lanzando el mensaje de “No hay suficiente saldo”

- El caso de prueba para los valores límite inferior inválido. La salida del test permite comprobar, en cada caso, que el saldo final coincide con el esperado. Lanzando el mensaje de excepción de “No se puede retirar una cantidad negativa.

- El caso de prueba para los valores límite no numérico inválido. La salida del test permite comprobar, en cada caso, que el saldo final coincide con el esperado. El propio programa evita que se pueda ingresar un dato no númerico, lanzando un error.