UD3.EXAME Práctico

DAM1-Contornos de Desenvolvemento 2024-25
26/03/2025

- Puedes utilizar apuntes y materiales que consideres pero deberás realizar los programas individualmente. En caso contrario se retirará el examen.
- Indica la autoría del código incluyendo un comentario con tu nombre y apellidos.

1. **Descarga,** se é o caso, o código fonte do exercicio do **repositorio** (ou onde che indique o profesor). Recoméndase configuralo nun novo proxecto Java. Precisarás a librería de probas de Junit5.
2. Resposta no propio documento editable, amplía as táboas se é necesario.
3. **Entrega:** O documento coas respostas en PDF e os ficheiros de código fonte que xeraras (Clases de probas solicitadas)
4. **Tiempo máximo:** 1:30 horas

1. Sistema de Bonificacións en Compras Online (6)

Dado o seguinte **enunciado** e a **implementación** levada a cabo pol@ programador@, aplica a técnica de Proba do Camiño Básico e realiza as seguintes tarefas:

1. Representa o **grafo de fluxo**
2. Calcula a **complexidade ciclomática** (de McCabe)
3. Detalla os **camiños independentes** e elabora os **casos de proba**
4. Implementa unha clase de proba en Junit5.
    a. Engade un comentario de Autoría na clase de probas.
5. Executa as probas, analiza os resultados, identifica posibles erros no código e como correxilos.
    a. Engade captura do resultado da execución das probas amosando cobertura.

> **Nota:** Para probar valores decimais utiliza o seguinte método:
> `assertEquals(valor esperado, valor real, tolerancia)`
> onde `tolerancia` é a marxe de erro tolerada na proba por cuestións de precisión. Podes usar o valor **`0.01`**

Enunciado

Unha empresa aplica bonificacións aos clientes segundo o total da súa compra e se pertencen ao programa de fidelización. As regras son:

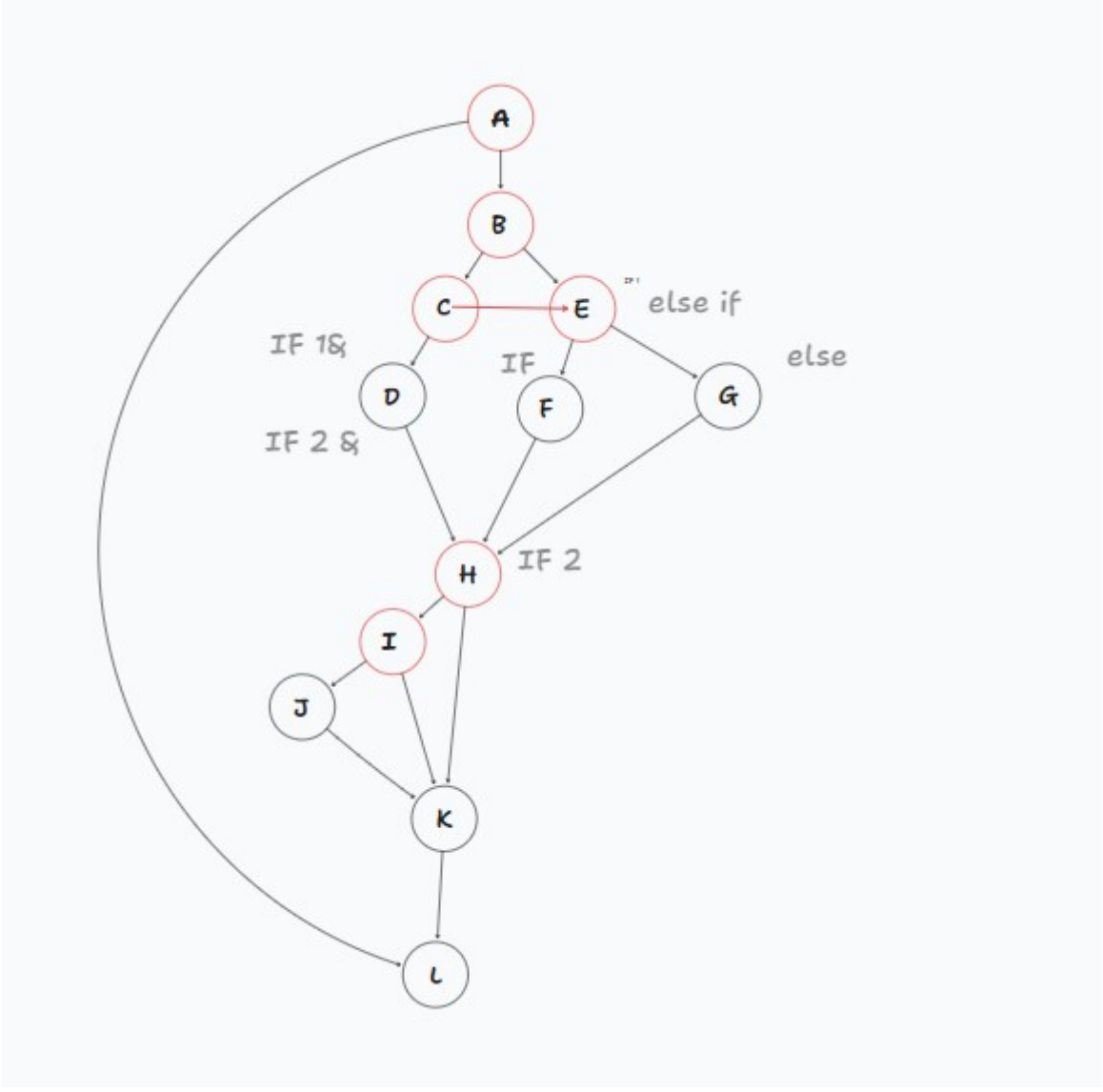
1. O importe da compra non pode ser inferior a 0€ (se ocorre, debe lanzar unha excepción).
2. Se o importe da compra é menor de 50€, non hai desconto.
3. Se está entre 50€ e 100€, aplícase un 5% de desconto.
4. Se iguala ou supera os 100€, aplícase un 10% de desconto.
5. Se o cliente pertence ao programa de fidelización e a compra iguala ou supera os 200€, aplícase un desconto extra do 5%.

Implementación

```
public class DiscountSystem {
    public static double applyDiscount(double amount, boolean isLoyalCustomer) {
        if (amount < 0) {
            throw new IllegalArgumentException("O importe non pode ser negativo");
        }

        double discount = 0.0;

        if (amount >= 50 && amount <= 100) {
            discount = 0.05;
        } else if (amount > 100) {
            discount = 0.10;
        }

        if (isLoyalCustomer && amount >= 200) {
            discount -= 0.05;
        }

        double finalAmount = amount - (amount * discount);

        return finalAmount;
    }
}
```

Solución

Grafo de fluxo



Complexidade ciclomática

V(G) = 7

Camiños independentes

1. **A,B,C,D,H,I,J,K,L**
2. **A,L**
3. **A,B,C,D,H,K,L**
4. **A,B,E,F,H,I,J,K,L**
5. **A,B,E,G,H,K,L**
6. **A,B,E,F,H,K,L**

7. **A,B,E,G,H,I,J,K,L**

Casos de proba

Entrada	Valor Esperado
-1 , FALSE	ERROR
30, FALSE	30
50, FALSE	47,5
110, FALSE	99
200, TRUE	170
200 , FALSE	180

Clase de Probas en Junit5

```
@Test
void testApplyDiscountMenorQue0() {
    try {
        double precio = DiscountSystem.applyDiscount(-1, isLoyalCustomer:false);    The value of
        fail("O importe non pode ser negativo");
    } catch (Exception e) {
        // TODO: handle exception    TODO: handle exception
    }
}

@Test
void testSinDescuento() {
    assertEquals(30, DiscountSystem.applyDiscount(amount:30, isLoyalCustomer:false), 0.01);
}

@Test
void testConDescuentodel005() {
    assertEquals(47.5, DiscountSystem.applyDiscount(amount:50, isLoyalCustomer:false), 0.01);

}

@Test
void testConDescuentodel010() {
    assertEquals(99, DiscountSystem.applyDiscount(amount:110, isLoyalCustomer:false), 0.01);

}

@Test
void testConDescuentodel015() {
    assertEquals(170, DiscountSystem.applyDiscount(amount:200, isLoyalCustomer:true), 0.01);

}

@Test
void testConDescuentodel10False() {
    assertEquals(180, DiscountSystem.applyDiscount(amount:200, isLoyalCustomer:false), 0.01);

}
```

Captura/s do resultado de executar as probas

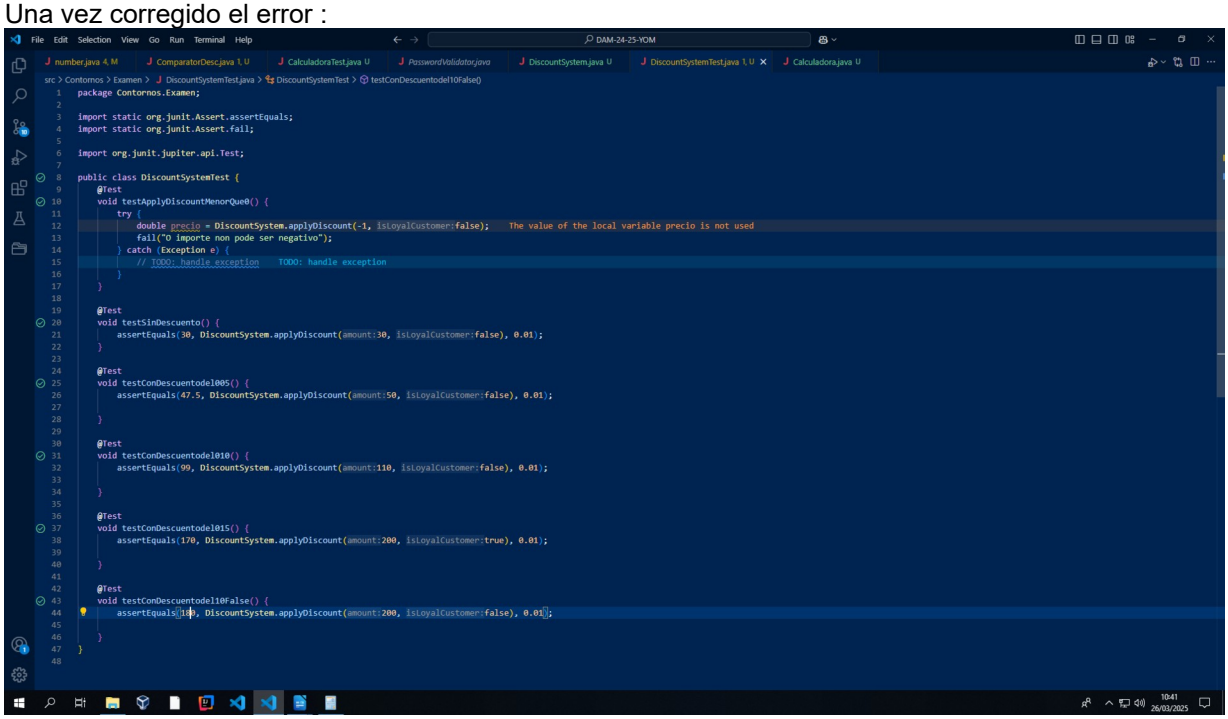
```
35
36      @Test
37      void testConDescuentodel015(){
38          assertEquals(170, DiscountSystem.applyDiscount(amount:200, isLoyalCustomer:true), 0.01);
```

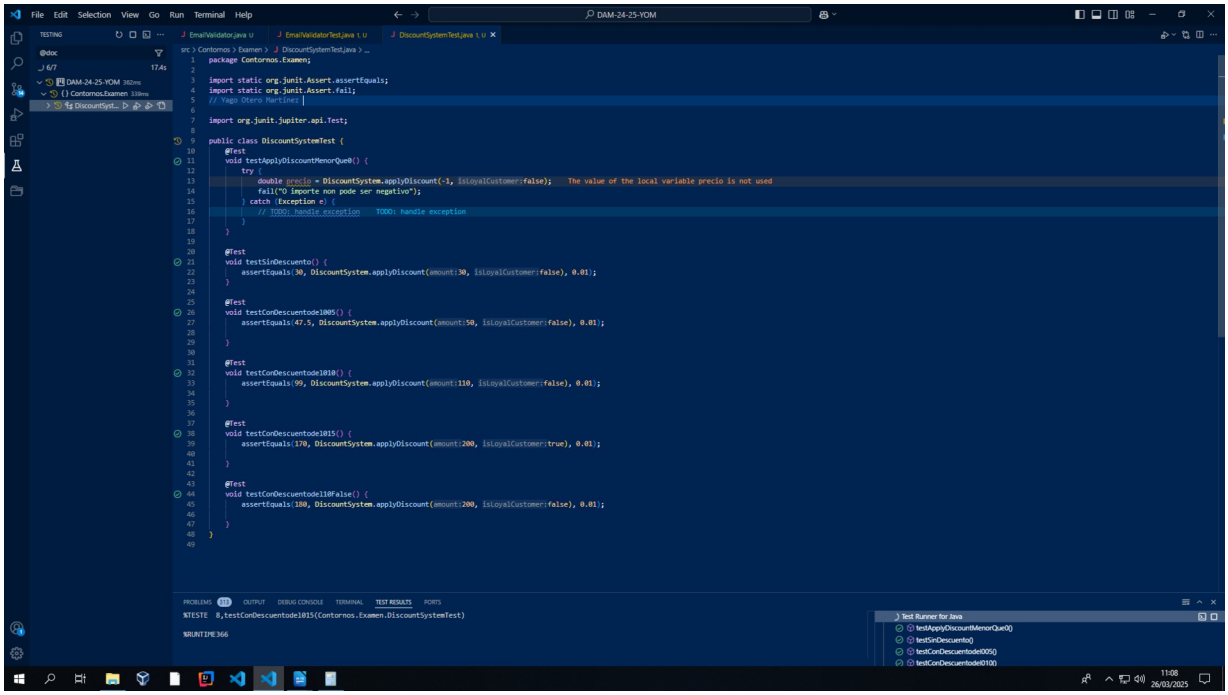
Expected [170.0] but was [190.0] testConDescuentodel015()

Expected	Actual
-170.0	+190.0

```
39
```

Una vez corregido el error :





Erro/s atopados

El descuento del Loyar esta restando en vez de sumando

2. Validación de correos electrónicos (4)

Dado o seguinte **enunciado** e a **implementación** levada a cabo pol@ programador@:

1. Crea unha táboa de clases de equivalencia
2. Xera casos de proba correspondentes indicando as clases de equivalencia cubiertas en cada caso.
3. Implementa unha clase de proba en Junit5.
    a. Engade un comentario de Autoría na clase de probas.
4. Executa as probas e amosa o resultado.

Enunciado:

Crea unha clase EmailValidator con un método isValid(String email). O correo é válido se:

- É distinto de null
- Contén exactamente un símbolo @.
- O dominio (parte despois do @) ten polo menos un punto (.).
- O nome de usuario (parte antes do @) non está baleiro.

Implementación:

```
public class EmailValidator {
    public static boolean isValid(String email) {
        if (email == null || !email.contains("@")) {
            return false;
        }
        String[] parts = email.split("@");
        if (parts.length != 2 || parts[0].isEmpty()) {
            return false;
        }
        return parts[1].contains(".");
    }
}
```

Solución

- É distinto de null
- Contén exactamente un símbolo @.
- O dominio (parte despois do @) ten polo menos un punto (.).
- O nome de usuario (parte antes do @) non está baleiro.

Táboa de Clases de Equivalencia

Condición de Entrada	Clases Válidas	Clases Non Válidas
• É distinto de null	!= null (1)	Null (2)
• Contén exactamente un símbolo @.	Que tenga un @ (3)	Que no lo tenga (4)
• O dominio (parte despois do @) ten polo menos un punto (.).	Después del arroba que haya un . (5)	Que no haya ninguno en el dominio (6)
• O nome de usuario (parte antes do @) non está baleiro.	Que haya una letra como mínimo antes del @ (7)	Que vaya el @ directo (8)

Casos de proba con clases de equivalencia válidas

Entrada1	Entrada2	Entrada3	Clases incluidas
yago@yago.com	a@a.com	A@b.com	1,3,5,7

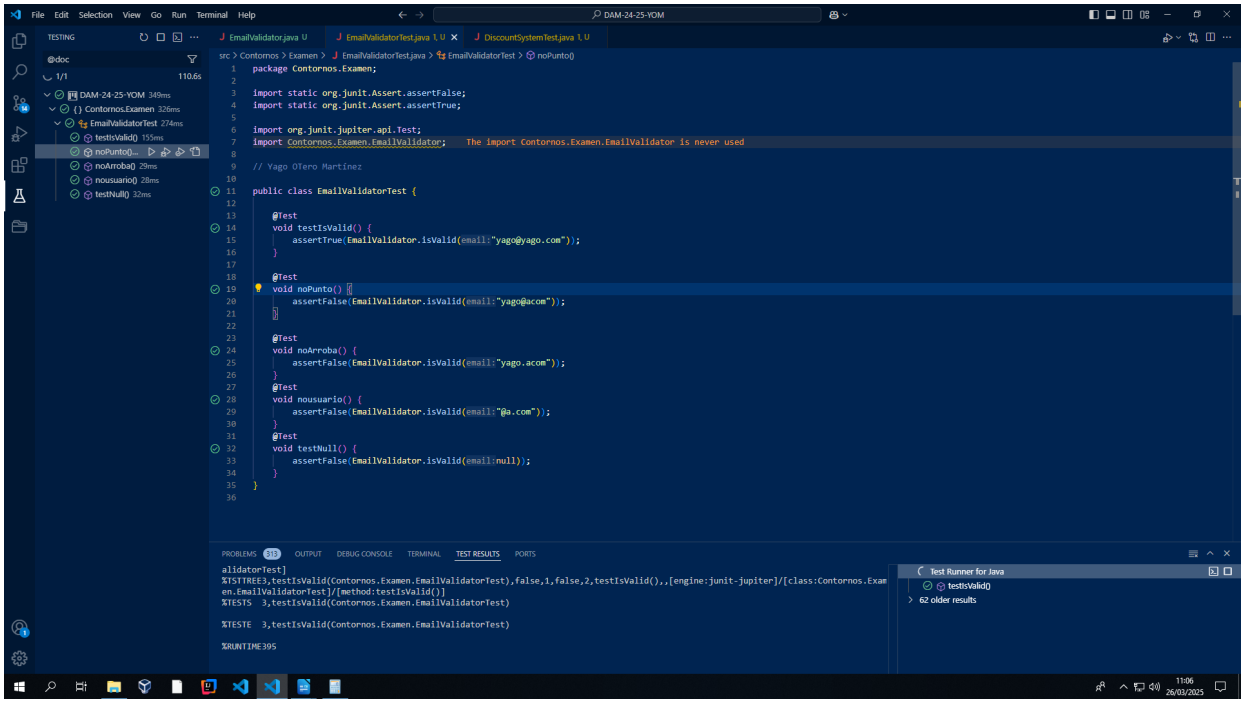
Casos de proba con clases de equivalencia non válidas

Entrada1	Entrada2	Entrada3	Clases incluidas
@yago.com			1,3,5,8
null			2,4,6,8
yago@yago			1,3,6,7
@yago			1,3,6,8
@			1,,3,6,8
@.			1,3,5,8

Clase de Probas en Junit5

```java
public class EmailValidatorTest {

    @Test
    void testIsValid() {
        assertTrue(EmailValidator.isValid(email:"yago@yago.com"));
    }

    @Test
    void noPunto() {
        assertFalse(EmailValidator.isValid(email:"yago@acom"));
    }

    @Test
    void noArroba() {
        assertFalse(EmailValidator.isValid(email:"yago.acom"));
    }
    @Test
    void nousuario() {
        assertFalse(EmailValidator.isValid(email:"@a.com"));
    }
    @Test
    void testNull() {
        assertFalse(EmailValidator.isValid(email:null));
    }
}
```

Captura do resultado de executar as probas



No te puedo añadir capturas de pantalla del coverage debido a que una vez que se terminan los test se queda en bucle cargando