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. Exercicio Caixa Branca (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

Implementa unha clase MathUtils con un método factorial(int n) que devolva o factorial dun número enteiro positivo. O método debe lanzar unha excepción se n é negativo.

Implementación

```
public class MathUtils {
    public static int factorial(int n) {
        if (n < 0) {
            throw new IllegalArgumentException("O número debe ser positivo");
        }
        int resultado = 1;
        for (int i = 1; i <= n; i--) {
            resultado *= i;
        }
        return resultado;
    }
}
```

Solución

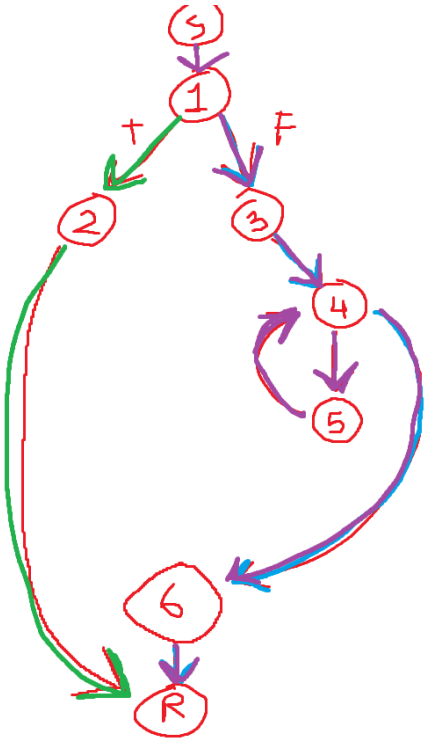
Grafo de fluxo

```
public class MathUtils {
    public static int factorial(int n) {
        if (n < 0) {
            throw new IllegalArgumentException("O número debe ser positivo");
        }
        int resultado = 1;
        for (int i = 1; i <= n; i--) {
            resultado *= i;
        }
        return resultado;
    }
}
```

S → 1 → 2 → R

S → 1 → 3 → 4 → 6 → R

S → 1 → 3 → 4 → 5 → 4 → 6 → R



Complexidade ciclomática

V(G) = 3

Camiños independentes

1. **S → 1 → 2 → R**
2. **S → 1 → 3 → 4 → 6 → R**
3. **S → 1 → 3 → 4 → 5 → 4 → 6 → R**

Casos de proba

Entrada	Valor Esperado
1	1
2	2
3	6

Clase de Probas en Junit5

```java
package ud5.pruebaExamen;

import static org.junit.Assert.assertEquals;

import org.junit.jupiter.api.Test;

public class MathUtilsTest {
    @Test
    void testFactorial1() {
        assertEquals(1, MathUtils.factorial(n:1));
    }

    @Test
    void testFactorial2() {
        assertEquals(2, MathUtils.factorial(n:2));
    }

    @Test
    void testFactorial3() {
        assertEquals(6, MathUtils.factorial(n:3));
    }
}
```

Captura/s do resultado de executar as probas

// Amosa captura da primeira execución das probas e da cobertura.

Erro/s atopados

```
// Error: Todos los resultados dan 0. El bucle va para atrás de forma infinita eventualmente multiplicando por 0
```

2. Exercicio Caixa Negra (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 PasswordValidator con un método isValid(String password). A clave é válida se:

- Ten polo menos 8 caracteres.
- Contén polo menos unha letra maiúscula.
- Contén polo menos un número.

Implementación:

```
public class PasswordValidator {
    public static boolean isValid(String password) {
        if (password.length() <= 8) {
            return false;
        }
        boolean hasUpperCase = false;
        boolean hasDigit = false;
        for (char c : password.toCharArray()) {
            if (Character.isLowerCase(c)) {
                hasUpperCase = true;
            }
            if (Character.isDigit(c)) {
                hasDigit = false;
            }
        }
        return hasUpperCase || hasDigit;
    }
}
```

Solución

Táboa de Clases de Equivalencia

Condición de Entrada	Clases Válidas	Clases Non Válidas
PASSWORD	Cadena de como mínimo 8 caracteres (1) Mínimo debe haber 1 dígito (2) Mínimo debe haber una mayúscula (3)	Longitud de menos de 8 (4) No contiene un dígito (5) No contiene una mayúscula (6)

Casos de proba con clases de equivalencia válidas

Entrada1	Entrada2	Entrada3	Clases incluidas
`contraseñaSegura123`	`abcdeABCDE123`	`B2c3d4e5f6`	`(1)(2)(3)`

Casos de proba con clases de equivalencia non válidas

Entrada1	Entrada2	Entrada3	Clases incluidas
…			

Clase de Probas en Junit5

// Pega o teu código aquí. Engade o ficheiro á entrega

Captura do resultado de executar as probas

// Amosa captura da primeira execución das probas