INFORME. TESTING PROBLEMA 1.

TRABAJO TEÓRICO ISO II

PAREJA 3. DAVID DIONISIO LAUREANO Y SEBASTIÁN MEDINA SÁNCHEZ

- Proyecto: implementación de Pruebas para la clase Año.
- Rama de trabajo: hotfix.

1. Estrategia y Selección de Casos de Prueba

Para la definición de los casos de prueba se han seguido las estrategias de Caja Negra vistas en teoría, asegurando una cobertura funcional completa del dominio de entrada.

- Técnicas empleadas:
  - Clases de Equivalencia: para dividir el dominio de entrada en rangos válidos e inválidos.
  - Valores Límite: selección de valores en las fronteras de las clases (ej. -45, 1582).
  - Conjetura de Errores: selección de valores propensos a fallos (ej. 0, caracteres no numéricos).
- Criterio de Cobertura: se ha aplicado el criterio "Each Use", asegurando que cada valor interesante identificado sea utilizado al menos en un caso de prueba.

2. Definición de Valores de Prueba

Se han identificado y probado un total de 17 valores clave para la variable año, cubriendo todos los comportamientos esperados del sistema:

- Valores Inválidos:
  - -80: valor muy por debajo del límite permitido.
  - 0: caso especial no existente en el calendario histórico.
- Valores Límite:
  - -45: límite inferior válido.
- Lógica de Años Bisiestos:
  - 2000 (Múltiplo de 400): Bisiesto.
  - 2004 (Múltiplo de 4, no de 100): Bisiesto.
  - 1900 (Múltiplo de 100, no de 400): No Bisiesto.
  - 2001 (No múltiplo de 4): No Bisiesto.

3. Resolución de Tests Fallidos y Correcciones de Código

Durante la fase de ejecución de pruebas, se detectaron fallos iniciales que obligaron a refactorizar el código de la clase Año y la configuración del proyecto:

A. Validación del Constructor

- Fallo: el test testConstructor_Invalido_MenorQueMenos45 fallaba inicialmente porque el constructor aceptaba cualquier entero.
- Solución: ee implementó una cláusula de guarda en el constructor para lanzar una IllegalArgumentException si el año es 0 o menor que -45.

B. Cálculo de Bisiestos (Algoritmo Gregoriano)

- Fallo: el valor 1900 se detectaba erróneamente como bisiesto al ser divisible entre 4.
- Solución: se anidaron las condiciones lógicas para cumplir la regla: (año % 4 == 0) && ((año % 100 != 0) || (año % 400 == 0)).

C. Cobertura de Código (JaCoCo)

- Fallo: el reporte inicial de JaCoCo mostraba una cobertura del 31%, incumpliendo el requisito mínimo del 80%. Esto se debía a que las clases App (Main) e InterfazUsuario se contaban en la media, pero no son lógica de negocio.
- Solución: se modificó el pom.xml para añadir exclusiones (<excludes>) en el plugin de JaCoCo.
- Resultado Final: se alcanzó un 100% de cobertura en la clase crítica Año y un 100% en la gestión de excepciones, superando el objetivo de la práctica.