Actividad 2: Javadoc y JUnit

Lee detenidamente el enunciado completo de esta actividad

Ejercicio 1: **Javadoc**

Requerimiento 1

Queremos documentar con **JavaDoc** una serie de clases que implementen una calculadora. Para ello hemos predefinido una serie de clases y métodos que el equipo tendrá que documentar. Esta actividad está pensada para que cada miembro del equipo documente una clase.

Se pide implementar el código mediante Java e IntelliJ. Además, hay que tener en cuenta que cada integrante del equipo tendrá que trabajar con **GIT** y todo el equipo tendrá un **repositorio GIT común (Github).**

Cada integrante del equipo tendrá la labor de implementar una clase completa haciendo los commits necesarios a su repositorio local y sincronizarse con el resto del equipo a través del repositorio remoto.

Cada integrante trabaje con una rama propia, pero cada vez que haga una subida al repositorio compartido fusione su rama con la rama máster para poder ir viendo los cambios del resto del equipo.

Se pide una clase main que permita probar las clases de la calculadora. Para ello cada integrante tendrá que modificar el código que llama a los métodos de su clase en la clase main, es decir, todos los integrantes del equipo tendrán que trabajar sobre la misma clase main. C integrante probará los métodos que ha realizado.

Nótese que en esta parte es probable que dé problemas de sincronización, por lo que se espera que los integrantes del equipo hablen y decidan como afrontar el problema.

Para toda la actividad se valorará la claridad de código y lo más importante respetar las decisiones documentadas en el Javadoc. En caso de que se decida modificar esa documentación (Ej. se han detectado carencias o errores) se podrá hacer siempre y cuando se justifique.

Cada clase implementada, deberá disponer de un comentario en el que figurará el nombre y el perfil de GitHub del autor de la misma.

Clase 1: Suma.

Esta clase implementará todos los métodos que están relacionados con la suma de la calculadora. Esta clase tendrá los siguientes métodos:

1. Suma de dos números reales, tendrá 2 parámetros de entrada y uno de salida que será la solución.
2. Suma de dos números enteros, tendrá 2 parámetros de entrada y uno de salida que será la solución.
3. Suma de tres números reales, tendrá 3 parámetros de entrada y uno de salida que será la solución.

4. Suma con valor acumulado, tendrá un parámetro de entrada y la clase deberá de guardar el valor acumulado.

Clase 2: Resta.

Esta clase implementará todos los métodos que están relacionados con la resta de la calculadora. Esta clase tendrá los siguientes métodos:

1. Resta de dos números reales, tendrá 2 parámetros de entrada y uno de salida que será la solución.
2. Resta de dos números enteros, tendrá 2 parámetros de entrada y uno de salida que será la solución.
3. Resta de tres números reales, tendrá 3 parámetros de entrada y uno de salida que será la solución.
4. Resta con valor acumulado, tendrá un parámetro de entrada y la clase deberá de guardar el valor acumulado.

Clase 3: Producto.

Esta clase implementará todos los métodos que están relacionados con la multiplicación de la calculadora. Esta clase tendrá los siguientes métodos:

1. Producto de dos números reales, tendrá 2 parámetros de entrada y uno de salida que será la solución.
2. Producto de dos números enteros, tendrá 2 parámetros de entrada y uno de salida que será la solución.
3. Producto de tres números reales, tendrá 3 parámetros de entrada y uno de salida que será la solución.
4. Potencia, tendrá dos parámetros de entrada (base y exponente) y uno de salida que será la solución

Clase 4: Cociente.

Esta clase implementará todos los métodos que están relacionados con la división de la calculadora. Esta clase tendrá los siguientes métodos:

1. División de dos números reales, tendrá 2 parámetros de entrada y uno de salida que será la solución.
2. División de dos números enteros, tendrá 2 parámetros de entrada y uno de salida que será la solución.
3. Inverso de un número real, tendrá un parámetro de entrada y uno de salida que será la solución.
4. Raíz de un número, tendrá un parámetro de entra y uno de salida que será la solución.

Valoración: 6 puntos sobre 10

Requerimiento 2

Generar la documentación de **JavaDoc** mediante IntelliJ o IDE equivalente.

Valoración: 2 puntos sobre 10

Requerimiento 3

Documentación de casos especiales, unos ejemplos podrían ser:

1. ¿Qué ocurre cuando alguno de los parámetros de entrada es cero?
2. Supongamos que la calculadora no puede utilizar números negativos ¿Cómo lo documentaríamos?
3. ¿Qué ocurriría si la división no da un número exacto?
4. ¿Qué ocurriría con la potencia si los números pasados son muy grandes?

Valoración: 2 puntos sobre 10

Ejercicio 2: **Pruebas unitarias. JUnit**

Requerimiento 1

Realizar las pruebas unitarias del proyecto de la actividad 2 (la calculadora).

Para ello cada miembro del equipo deberá de probar una clase creada por otro miembro del equipo. Se valorará que no se hagan parejas de prueba, esto es, si el alumno A prueba el código del alumno B, el alumno B no debería de probar el código del alumno A.

Ojo, se deben de probar todos los métodos y todas las posibles casuísticas de ellos (prestando especial atención a los casos especiales).

Valoración: 10 puntos sobre 10

Consideraciones

Para toda la actividad se valorará:

- El trabajo en equipo, evaluando la colaboración, comunicación y gestión de tareas.
- El orden y la claridad en la documentación generada (JavaDoc y casos especiales).
- La correcta implementación y funcionalidad de los métodos, así como la cobertura de las pruebas unitarias con JUnit.
- El uso adecuado de GIT y GitHub, incluyendo mensajes descriptivos en los commits, organización en ramas, y fusiones bien gestionadas.

Entrega

La entrega debe incluir:

1. **Código fuente comprimido (ZIP o equivalente):**
    - Archivos fuente en Java.
    - Ficheros generados de JavaDoc.
    - Pruebas unitarias realizadas con JUnit.
2. **Documento formal (3 a 4 páginas):**
   Este documento deberá incluir:
    - **Portada:**
        - Título de la actividad.
        - Nombres de los integrantes del equipo.
        - URL del repositorio de GitHub compartido.
    - **Introducción:** Breve descripción de la actividad y su propósito.
    - **Metodología:** Pasos realizados, uso de GIT y herramientas empleadas.
    - **Problemas y soluciones:** Principales retos encontrados y cómo se abordaron.
    - **Conclusiones:** Reflexión sobre el trabajo en equipo y el aprendizaje.
    - **Desglose de tareas:** Descripción de las labores realizadas por cada integrante.
3. **Uso del repositorio de GitHub compartido:**
    - Cada integrante debe utilizar un comentario en su código que incluya la URL del repositorio GitHub compartido y su perfil.
    - El profesor (***luisgs-unir***) debe figurar como colaborador del repositorio para poder evaluar directamente el trabajo y el historial de cambios.
4. **Comentarios individuales:**
    - Cada entrega individual (código de las clases o pruebas unitarias) debe incluir en los comentarios iniciales:
        - Nombre del integrante que desarrolló la parte del proyecto.
        - URL del repositorio GitHub compartido.
        - Breve descripción de los cambios realizados.
5. **Gestión en GitHub:**
    - La rama principal debe contener una versión funcional del proyecto.
    - Las ramas individuales deben estar bien gestionadas, con fusiones claras y mensajes descriptivos en los commits.

Notas adicionales

- El profesor revisará el repositorio para verificar el uso adecuado de GIT, la estructura del proyecto y la colaboración entre los integrantes.
- La inclusión del profesor como colaborador del repositorio es obligatoria para que pueda realizar un seguimiento completo del trabajo en equipo.

Rúbrica

Criterio	Descripción	Nivel Excelente (4)	Nivel Bueno (3)	Nivel Suficiente (2)	Nivel Insuficiente (1)	Peso
Documentación con JavaDoc	Calidad, claridad y completitud de la documentación generada para las clases y métodos.	Documentación completa, precisa y profesional. Incluye detalles técnicos y casos especiales relevantes.	Documentación completa, pero con pequeños errores o falta de algún detalle menor.	Documentación parcial, con errores frecuentes o ausencia de aspectos clave en métodos y clases.	Documentación incompleta, confusa o inexistente.	18%
Casos especiales	Inclusión y tratamiento adecuado de los casos especiales en la documentación.	Todos los casos especiales identificados y documentados claramente con explicaciones adecuadas.	La mayoría de los casos especiales identificados, con explicaciones claras.	Algunos casos especiales identificados, pero sin suficiente claridad o detalle.	No se identificaron ni documentaron casos especiales.	18%
Implementación del código	Correcta implementación y funcionalidad de los métodos, siguiendo las especificaciones indicadas.	Implementación correcta y funcional de todos los métodos, sin errores de compilación ni ejecución.	Implementación funcional, pero con errores menores o falta de algún detalle en las especificaciones.	Implementación parcial, con errores significativos o métodos sin implementar.	Implementación incompleta, con múltiples errores de ejecución o incumplimiento de las especificaciones.	10%
Pruebas unitarias (JUnit)	Calidad, cobertura y claridad de las pruebas realizadas sobre el código de otros integrantes.	Pruebas completas y claras que cubren todas las casuísticas relevantes, incluidas las excepciones.	Pruebas completas pero con casos especiales poco desarrollados o algunas casuísticas faltantes.	Pruebas incompletas o con errores significativos en su implementación.	No se realizaron pruebas unitarias o son inadecuadas para validar los métodos.	24%
Trabajo en equipo	Colaboración, comunicación, gestión de tareas y uso adecuado del repositorio GIT compartido.	Excelente colaboración y comunicación. Uso impecable de GIT, con ramas bien gestionadas y commits descriptivos.	Buena colaboración y comunicación, con pequeñas áreas de mejora en el uso de GIT.	Colaboración básica, con algunos problemas en la gestión de tareas o uso de GIT.	Falta de colaboración evidente, sin evidencia de comunicación ni gestión adecuada en GIT.	15%
Informe y entrega final	Calidad y organización del documento formal, cumplimiento de los requisitos de entrega.	Documento bien estructurado, completo y claro, con todos los requisitos cumplidos.	Documento claro, pero con faltas menores en la estructura o requisitos.	Documento incompleto o poco claro, con varios requisitos faltantes.	Documento desorganizado o inexistente, con gran parte de los requisitos incumplidos.	15%

Anexo I: Ampliación de clases

Clase 5: Módulo

Esta clase proporciona operaciones relacionadas con el **módulo** (residuo de una división) y el **valor absoluto**.

Métodos:

1. **Cálculo del módulo (residuo de una división entera).**
    - Entrada: int a, int b
    - Salida: int (el residuo de la división de a entre b)
    - **Fórmula:**
        - Residuo: M = a % b
2. **Cálculo del valor absoluto de un número.**
    - Entrada: double a
    - Salida: double (el valor absoluto de a)
    - **Fórmula:**
        - Si a ≥ 0, entonces |a| = a
        - Si a < 0, entonces |a| = -a

Clase 6: Trigonometría

Esta clase implementa **funciones trigonométricas** básicas como seno, coseno y tangente.

Métodos:

1. **Cálculo del seno de un ángulo.**
    - Entrada: double angulo (en radianes)
    - Salida: double (el seno del ángulo)
    - **Fórmula:**
        - sin(angulo) = cateto opuesto / hipotenusa
2. **Cálculo del coseno de un ángulo.**
    - Entrada: double angulo (en radianes)
    - Salida: double (el coseno del ángulo)
    - **Fórmula:**
        - cos(angulo) = cateto adyacente / hipotenusa

3. **Cálculo de la tangente de un ángulo.**
    - Entrada: double angulo (en radianes)
    - Salida: double (la tangente del ángulo)
    - **Fórmula:**
        - tan(angulo) = sin(angulo) / cos(angulo)

Clase 7: Logaritmos y Exponenciales

Esta clase proporciona funciones para calcular **logaritmos y exponenciales**.

Métodos:

1. **Cálculo del logaritmo natural (base e).**
    - Entrada: double x (x > 0)
    - Salida: double (logaritmo natural de x)
    - **Fórmula:**
        - ln(x) = y, si e^y = x
2. **Cálculo del logaritmo en base 10.**
    - Entrada: double x (x > 0)
    - Salida: double (logaritmo base 10 de x)
    - **Fórmula:**
        - log10(x) = y, si 10^y = x
3. **Cálculo de la exponencial (e^x).**
    - Entrada: double x
    - Salida: double (valor de e elevado a x)
    - **Fórmula:**
        - e^x

Clase 8: Estadística

Esta clase proporciona métodos para calcular **medidas estadísticas**.

Métodos:

1. **Cálculo de la media aritmética.**
    - Entrada: double[] numeros (lista de valores)
    - Salida: double (media de los valores)

- **Fórmula:**
  - media = (x1 + x2 + ... + xn) / n

2. **Cálculo de la varianza.**
   - Entrada: double[] numeros (lista de valores)
   - Salida: double (varianza de los valores)
   - **Fórmula:**
     - varianza = [(x1 - media)^2 + (x2 - media)^2 + ... + (xn - media)^2] / n
3. **Cálculo de la desviación estándar.**
   - Entrada: double[] numeros (lista de valores)
   - Salida: double (desviación estándar de los valores)
   - **Fórmula:**
     - desviación estándar = raíz cuadrada(varianza)

Clase 9: Conversión de Unidades

Esta clase contiene métodos para realizar conversiones de **unidades de temperatura y ángulos**.

Métodos:

1. **Conversión de grados Celsius a Fahrenheit.**
   - Entrada: double celsius
   - Salida: double (temperatura en Fahrenheit)
   - **Fórmula:**
     - F = (C × 9/5) + 32
2. **Conversión de grados Fahrenheit a Celsius.**
   - Entrada: double fahrenheit
   - Salida: double (temperatura en Celsius)
   - **Fórmula:**
     - C = (F - 32) × 5/9
3. **Conversión de grados a radianes.**
   - Entrada: double grados
   - Salida: double (ángulo en radianes)
   - **Fórmula:**
     - radianes = grados × π / 180

4. **Conversión de radianes a grados.**
    - Entrada: double radianes
    - Salida: double (ángulo en grados)
    - **Fórmula:**
        - grados = radianes × 180 / π