Trabajo practico 2:

Este informe detalla los errores encontrados durante la ejecución de las pruebas unitarias en el proyecto, especificando para cada clase de test los fallos detectados, sus causas técnicas y las soluciones aplicadas. Se incluye también una comparación de desempeño en pruebas concurrentes.

# Clase de Test: LogServiceTest

## Test: addLogs

* • Error detectado: Se agregaba el doble de elementos a los logs.
* • Causa técnica: Llamada duplicada a list.add(log) dentro del método addLog.
* • Solución aplicada: Eliminar la línea repetida que agregaba el log dos veces.

## Test: getLogsByUser

* • Error detectado: Usuarios con el mismo nombre eran tratados como distintos.
* • Causa técnica: La clase User no implementaba equals y hashCode.
* • Solución aplicada: Implementar equals y hashCode basados únicamente en el campo name.

## Test: hasAllLogsConcurrent (1)

* • Error detectado: NullPointerException al ejecutar el test concurrente.
* • Causa técnica: El constructor vacío de User no inicializaba el campo name.
* • Solución aplicada: Inicializar el campo name con un string vacío para evitar null.

## Test: hasAllLogsConcurrent (2)

* • Error detectado: expected:<100000> but was:<99963>. No se agregaban todas las acciones.
* • Causa técnica: Acceso concurrente no seguro a logs y a las listas internas.
* • Solución aplicada: Usar ConcurrentHashMap y Collections.synchronizedList.

Comparación de desempeño:  
- Solución con método synchronized y HashMap volatile: promedio 260 ms  
- Solución con ConcurrentHashMap y listas sincronizadas: promedio 185 ms

# Clase de Test: UserServiceTest

## Test: testLoginUser

* • Error detectado: No se mantenía el incremento correcto del ID de usuario.
* • Causa técnica: Uso incorrecto del operador lastId++.
* • Solución aplicada: Reemplazar lastId = lastId++ por lastId + 1 al crear el nuevo usuario.

## Test: testAddNullUser

* • Error detectado: Se lanzaba excepción al intentar guardar un usuario null.
* • Causa técnica: No se verificaba si el usuario era null antes de guardarlo.
* • Solución aplicada: Agregar verificación if (user != null) antes de guardar.

## Test: testSortedByName

* • Error detectado: Orden de usuarios incorrecto.
* • Causa técnica: El método getUsers() ordenaba por id en lugar de name.
* • Solución aplicada: Modificar getUsers() para ordenar por name alfabéticamente.

# Clase de Test: CurrentUserTest

## Test: testGetsDefaultUser

* • Error detectado: El usuario por defecto era null.
* • Causa técnica: Comparación lógica incorrecta en getCurrent.
* • Solución aplicada: Corregir condición: if (current == null).

## Test: testCurrentConcurrent

* • Error detectado: Acceso concurrente generaba múltiples instancias de usuario actual.
* • Causa técnica: getCurrent() no era thread-safe.
* • Solución aplicada: Declarar el método como synchronized.

# Clase de Test: TodoServiceTest

## Test: testCreateTodo

* • Error detectado: El ToDo creado no tenía contenido asignado.
* • Causa técnica: No se asignaba el valor de content al crear el ToDo.
* • Solución aplicada: Asegurar la asignación del contenido en el constructor o método correspondiente.

## Test: testTimerDelete

* • Error detectado: Se borraban ToDo con tiempo menor al especificado.
* • Causa técnica: Comparación lógica invertida.
* • Solución aplicada: Corregir la condición para eliminar correctamente los elementos.