Universidad Internacional Del Ecuador

Estructura de datos

Suite de tests (casos borde)

Estudiante:

Ricardo Alvarado

Docente:

Charlie Cárdenas

16/12/2025

PARTE 1 — IDENTIFICACIÓN DE CASOS BORDE

Caso Borde	Descripción	Por qué es crítico
1	Lista vacía	head == null y tail == null. Muchas operaciones pueden lanzar NullPointerException si no se valida este estado.
2	Lista con un solo elemento	head == tail. Los punteros prev y next deben ser null; errores aquí rompen la estructura de la lista.
3	Insertar en lista vacía	Primera inserción
4	Eliminar en lista vacía	deleteByValue() sin nodos
5	Eliminar head	Eliminar primer nodo
6	Eliminar tail	Eliminar último nodo
7	Eliminar único elemento	Lista queda vacía
8	Reverse lista vacía o de 1 elemento	reverse() sin cambios

PARTE 2 — DISEÑO DE CASOS DE PRUEBA

ID	Precondición	Acción	Resultado Esperado	Postcondición
TC-DLL-001	Lista vacía	deleteByValue(10)	Retorna false	head == null, tail == null
TC-DLL-002	Lista vacía	reverse()	Sin errores	Lista sigue vacía
TC-DLL-003	Lista vacía	insertAtEnd(5)	Inserta elemento	head == tail
TC-DLL-004	1 elemento	reverse()	Sin cambios	head == tail
TC-DLL-005	1 elemento	deleteByValue(valor)	Retorna true	Lista vacía (head == null, tail == null)
TC-DLL-006	3 elementos	Eliminar head	Retorna true	head.prev == null
TC-DLL-007	3 elementos	Eliminar tail	Retorna true	tail.next == null
TC-DLL-008	3 elementos	Eliminar intermedio	Retorna true	Enlaces prev/next consistentes
TC-DLL-009	3 elementos	Eliminar inexistente	Retorna false	Lista intacta
TC-DLL-010	4 elementos	reverse()	Orden invertido	head.prev == null, tail.next == null

Parte 3: Implementación de Pruebas

Clase de prueba: DoublyLinkedListTest.java

- Se implementaron 10 métodos de prueba usando **JUnit 5**.
- Se usaron assertions: assertEquals, assertNull, assertTrue, assertFalse.
- Se verifican punteros head.prev y tail.next cuando corresponde.
- Mensajes descriptivos incluidos en cada assertion.

Resumen de pruebas implementadas:

Caso	Descripción	Método JUnit	Resultado esperado
TC_DLL_001	Eliminar en lista vacía	deleteByValue	false
TC_DLL_002	Reversa lista vacía	reverse	Lista sigue vacía
TC_DLL_003	Insertar en lista vacía	insertAtEnd	Head = Tail = nodo insertado
TC_DLL_004	Reversa lista de un elemento	reverse	Lista sin cambios
TC_DLL_005	Eliminar único elemento	deleteByValue	Lista queda vacía
TC_DLL_006	Eliminar head	deleteByValue	Nuevo head.prev = null
TC_DLL_007	Eliminar tail	deleteByValue	Nuevo tail.next = null (falla intencional)
TC_DLL_008	Eliminar nodo intermedio	deleteByValue	Enlaces prev/next correctos
TC_DLL_009	Eliminar nodo inexistente	deleteByValue	false
TC_DLL_010	Reversa lista múltiple	reverse	head = último elemento, tail = primer elemento (falla intencional)

Resultados de ejecución (Maven):

Tests run: 10, Failures: 2, Errors: 0, Skipped: 0

- Tests fallidos:
    - TC_DLL_007_deleteTail: tail.next no se actualiza a null.
    - TC_DLL_010_reverseMultipleElements: head y tail no se intercambian al invertir.

Parte 4: Reporte de Cobertura

4.1 Tabla resumen de resultados

Total pruebas	Pasadas	Fallidas	Errores	Skipped
10	8	2	0	0

4.2 Captura de pantalla de ejecución de pruebas

4.3 Análisis de cobertura

- **Operaciones cubiertas:**
  - insertAtEnd → todos los casos que agregan elementos (TC_DLL_003, TC_DLL_004, TC_DLL_005, TC_DLL_006, TC_DLL_007, TC_DLL_008, TC_DLL_009, TC_DLL_010)
  - deleteByValue → TC_DLL_001, TC_DLL_005, TC_DLL_006, TC_DLL_007, TC_DLL_008, TC_DLL_009
  - reverse → TC_DLL_002, TC_DLL_004, TC_DLL_010
  - isEmpty → implícito en TC_DLL_001, TC_DLL_002, TC_DLL_003, TC_DLL_005
  - Verificación de punteros → TC_DLL_006, TC_DLL_007, TC_DLL_010
- **Gaps (casos faltantes):**
  - Insertar elementos en medio de la lista (no solo al final).
  - Eliminar múltiples nodos consecutivos.
  - Probar reversa después de eliminaciones mixtas.
  - Comportamiento con nodos duplicados.

4.4 Propuestas de mejora para cobertura

1. Agregar métodos de inserción en posición específica para probar casos intermedios.
2. Validar eliminación de varios nodos consecutivos y punteros prev/next.
3. Probar reversa en listas modificadas dinámicamente (insert/elim combinados).