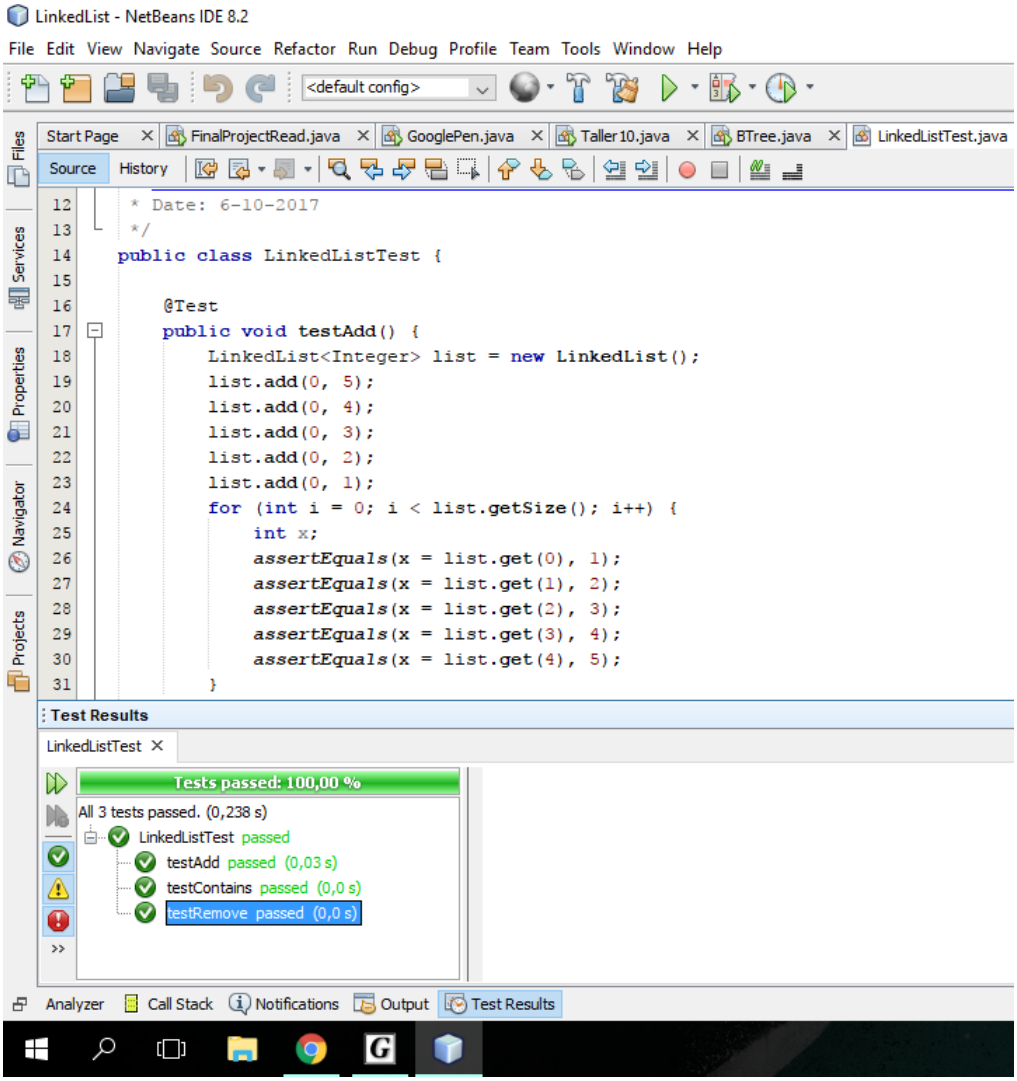
Laboratorio Nro. 04: Implementación de listas enlazadas

Agustín Nieto García
Universidad Eafit
Medellín, Colombia
anietog1@eafit.edu.co

David Immanuel Trefftz Restrepo
Universidad Eafit
Medellín, Colombia
ditrefftzr@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. Resultados en JUnit.



DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

2. Explicación del numeral 2.1 (Uva – The Blocks Problem).

Para la solución del problema, consistente en la manipulación de bloques que se podían apilar uno sobre otro, se creó una estructura de datos que permitiera realizar estas manipulaciones de manera eficiente y sencilla, mucho más sencilla que usando listas, sin embargo, con la idea de una lista en la implementación.

Las acciones realizables eran las siguientes: apilar y remover. Así, lo único que había que realizar para alguna de las operaciones era remover algunas relaciones o agregarlas y ya todo quedaba solucionado. Si se pedía apilar a sobre b, lo único que había que hacer es que el previo de a fuera b y el posterior de b fuera a, y eso es todo. Para remover, pues simplemente se hacía que el posterior de b fuera nadie y el anterior de a tampoco existiera. Y así para todos los casos, que había que realizar iterativamente.

3. Complejidad del numeral 2.1.

La complejidad del ejercicio se podría considerar O(1) debido a que el tamaño máximo definido para todos los problemas, es decir, la cantidad de bloques es 25, por lo que en cuando a la definición de las operaciones, consistentes en verificar/encontrar, limpiar y mover serían O(25), O(25) y O(1) en el peor de los casos (que no pueden ocurrir simultáneamente) y esto, trivialmente, es O(1) para todo. Sin embargo, si no se limitase la cantidad posible de bloques, la complejidad de cada operación sería:

Búsqueda/Verificación O(n)
Limpieza O(n)
Movimiento O(1)

4. Explicación de 'm' y 'n' en los cálculos de complejidad del numeral 2.1.

Búsqueda/Verificación:

O(n) → Siempre se busca por la coincidencia entre dos bloques a y b en un mismo stack. Pueden haber dos situaciones: a delante de b o b delante de a. Y así se procede, buscando alguna de las dos situaciones. En el peor de los casos ninguno ocurre y se recorrerían todos los elementos.

Limpieza:

O(n) → A la hora de realizar los movimientos hay que limpiar (remover) los bloques posteriores, bien al que se desea mover, bien al que se le desea

agregar. En el peor de los casos, hay que recorrer n – 1 bloques removiendo sus enlaces con el anterior y el posterior.

Mover:

O(1) → Consiste solo en cambiar uno que otro enlace, sin embargo, no se puede hacer sin la búsqueda y la limpieza, ya que sería un movimiento ilegal.

4) Simulacro de Parcial

1. **a.** lista.size()
   b. lista.add(auxiliar.pop());

2. **a**. !auxiliar1.isEmpty()
   b. !auxiliar2.isEmpty()
   c. personas.offer(edad);

3. O(n^2)

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co