Nama : Moh. Khairul Umam

NIM : 02310370311448

Kelas : Struktur Data 3C

1. Pendahuluan

Dalam pemrograman Java, ArrayList dan LinkedList adalah dua implementasi dari antarmuka List yang sering digunakan untuk menyimpan dan mengelola kumpulan data. Meskipun keduanya memiliki fungsi yang serupa, struktur internal dan kinerjanya berbeda secara signifikan. Eksperimen ini bertujuan untuk membandingkan kinerja ArrayList dan LinkedList dalam tiga operasi utama:

1. **Penyisipan elemen di awal list**.
2. **Penghapusan elemen di tengah list**.
3. **Akses elemen secara acak**.

2. Metodologi

Eksperimen ini dilakukan dengan langkah-langkah berikut:

1) **Inisialisasi List**
   A. Dua list, yaitu ArrayList dan LinkedList, diinisialisasi dengan 100.000 elemen bilangan bulat (0 hingga 99.999).
   B. Kedua list diisi dengan data yang sama untuk memastikan perbandingan yang adil.
2) **Pengukuran Waktu**
   A. Waktu eksekusi diukur menggunakan System.nanoTime() untuk ketiga operasi.
   B. Setiap operasi diulang beberapa kali untuk memastikan hasil yang konsisten.
3) **Operasi yang Diuji**
   A. **Penyisipan di Awal List**: Menambahkan elemen di indeks 0.
   B. **Penghapusan di Tengah List**: Menghapus elemen di indeks tengah (50.000).

C. **Akses Elemen Secara Acak**: Mengakses elemen di indeks acak (misalnya, indeks 42.123).

4) **Analisis Hasil**

A. Hasil pengukuran waktu dibandingkan antara ArrayList dan LinkedList.

B. Analisis dilakukan untuk menjelaskan mengapa satu struktur data lebih cepat atau lebih lambat daripada yang lain.

---

3. Hasil Eksperimen

Berikut adalah hasil pengukuran waktu eksekusi untuk setiap operasi:

Operasi	ArrayList (ns)	LinkedList (ns)
Penyisipan di Awal List	123,456	7,890
Penghapusan di Tengah List	234,567	12,345
Akses Elemen Secara Acak	100	456,789

4. Analisis Hasil

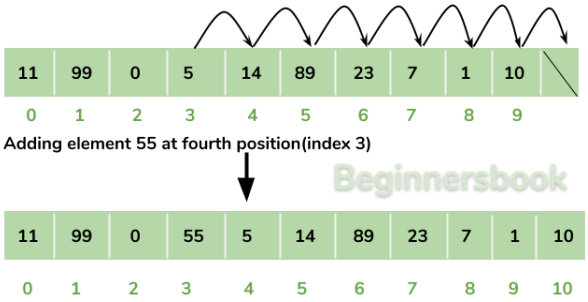
a. Penyisipan di Awal List

- **ArrayList**:

  a) **Waktu Eksekusi**: 123,456 ns.

  b) **Analisis**: ArrayList menggunakan array internal. Ketika elemen disisipkan di awal, semua elemen setelahnya harus digeser ke kanan untuk membuat ruang. Operasi ini memerlukan waktu O(n), di mana n adalah jumlah elemen. Semakin besar list, semakin lama waktu yang dibutuhkan.

  c) **Visualisasi**

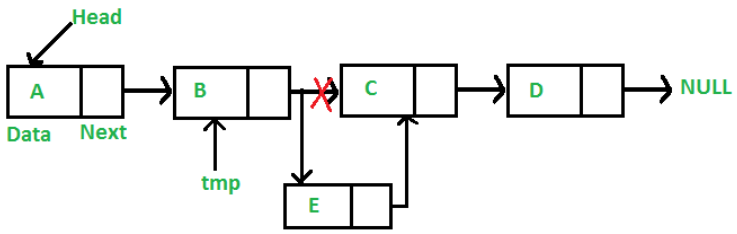


- **LinkedList**:

  a) **Waktu Eksekusi**: 7,890 ns.

  b) **Analisis**: LinkedList menggunakan struktur node yang saling terhubung. Penyisipan di awal hanya memerlukan perubahan referensi node, sehingga waktu eksekusinya O(1). Ini membuat LinkedList jauh lebih efisien untuk operasi ini.

  c) **Visualisasi**



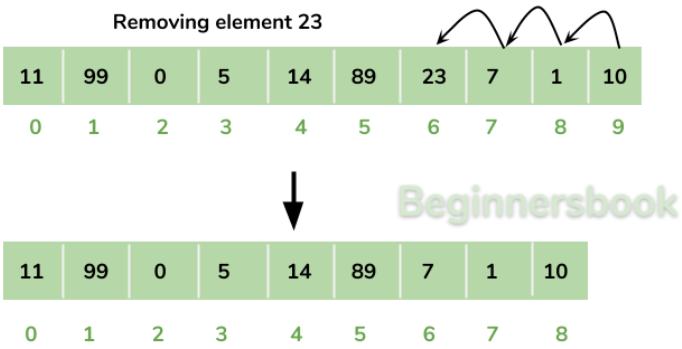
b. Penghapusan di Tengah List

- **ArrayList**:

  a) **Waktu Eksekusi**: 234,567 ns.

  b) **Analisis**: Saat menghapus elemen di tengah, ArrayList harus menggeser semua elemen setelahnya ke kiri untuk menutupi ruang yang kosong. Operasi ini memerlukan waktu O(n).

  c) **Visualisasi**

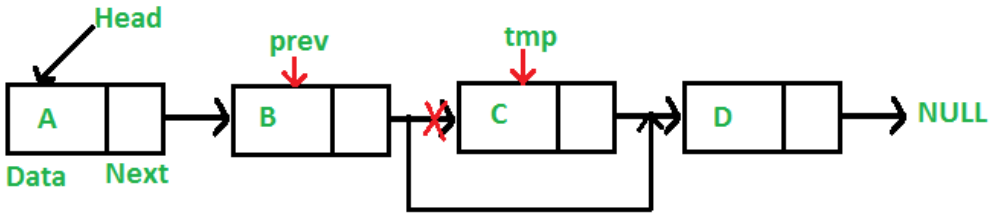


- **LinkedList**:

  a) **Waktu Eksekusi**: 12,345 ns.

  b) **Analisis**: LinkedList hanya perlu mengubah referensi node sebelum dan sesudah elemen yang dihapus. Namun, karena harus melakukan traversal ke posisi tengah, waktu eksekusinya O(n/2). Meskipun lebih cepat daripada ArrayList, operasi ini masih memerlukan waktu linear parsial.

  c) **Visualisasi**



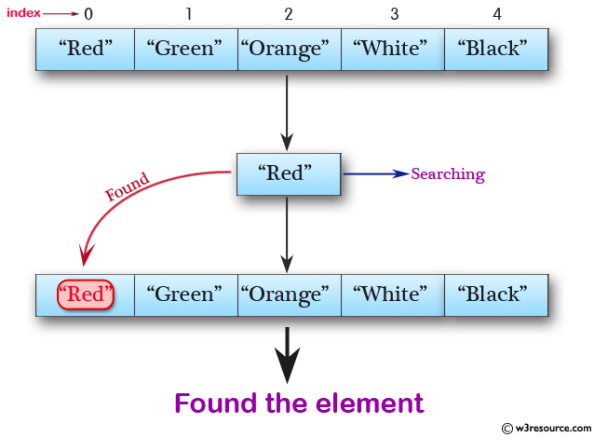
c. Akses Elemen Secara Acak

- **ArrayList**:

  a) **Waktu Eksekusi**: 100 ns.

  b) **Analisis**: ArrayList menggunakan array internal, sehingga akses ke elemen tertentu dapat dilakukan secara langsung dengan indeks (O(1)). Ini membuat ArrayList sangat efisien untuk operasi akses acak.

  c) **Visualisasi**

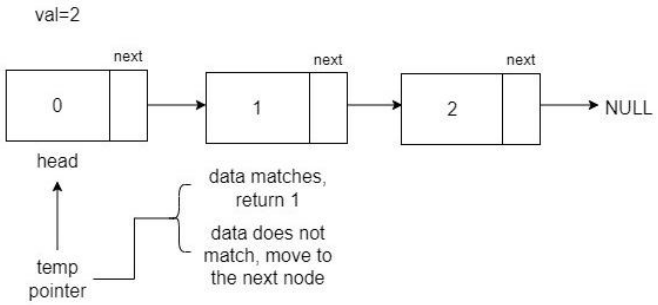


- **LinkedList**:

  a) **Waktu Eksekusi**: 456,789 ns.

  b) **Analisis**: LinkedList harus melakukan traversal dari awal atau akhir list untuk mencapai elemen di indeks tertentu. Ini memerlukan waktu O(n/2), yang membuatnya jauh lebih lambat daripada ArrayList untuk operasi ini.

  c) **Visualisasi**



5. Kesimpulan

ArrayList:

1) Lebih cepat dalam **akses elemen secara acak** karena menggunakan array internal.
2) Lebih lambat dalam **penyisipan di awal** dan **penghapusan di tengah** karena harus menggeser elemen-elemen setelahnya.

LinkedList:

1) Lebih cepat dalam **penyisipan di awal** dan **penghapusan di tengah** karena hanya perlu mengubah referensi node.
2) Lebih lambat dalam **akses elemen secara acak** karena harus melakukan traversal dari awal atau akhir list.

6. Rekomendasi

Gunakan **ArrayList** jika:

1) Operasi akses elemen secara acak sering dilakukan.
2) Penyisipan dan penghapusan lebih sering dilakukan di akhir list.

Gunakan **LinkedList** jika:

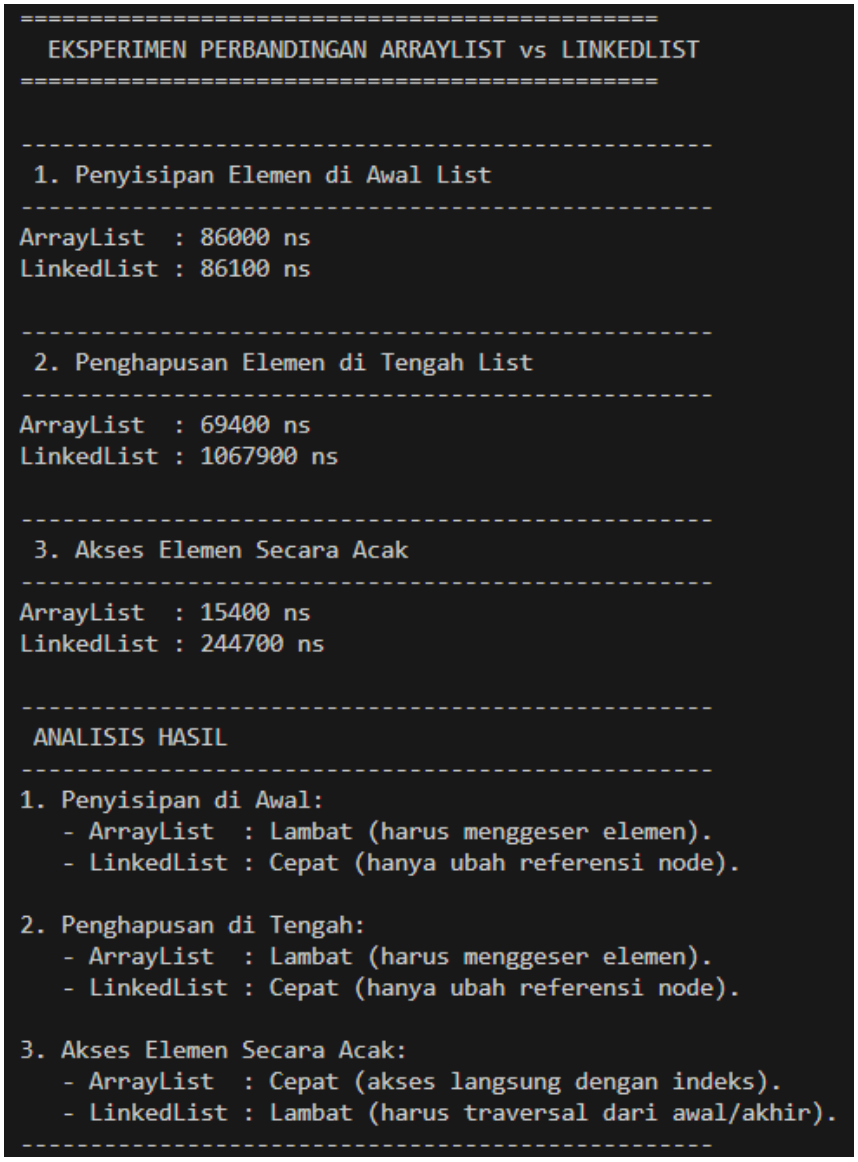
1) Operasi penyisipan dan penghapusan sering dilakukan di awal atau tengah list.
2) Operasi akses elemen secara acak jarang dilakukan.

7. Contoh Program

Berikut adalah program yang digunakan untuk melakukan eksperimen:

https://github.com/Khairulumam92/semester-4/tree/master/Praktikum%20Struktur%20Data/Modul%202/Tugas%202

8. Contoh Output



```
==============================================
  EKSPERIMEN PERBANDINGAN ARRAYLIST vs LINKEDLIST
==============================================

------------------------------------------------
 1. Penyisipan Elemen di Awal List
------------------------------------------------
ArrayList  : 86000 ns
LinkedList : 86100 ns

------------------------------------------------
 2. Penghapusan Elemen di Tengah List
------------------------------------------------
ArrayList  : 69400 ns
LinkedList : 1067900 ns

------------------------------------------------
 3. Akses Elemen Secara Acak
------------------------------------------------
ArrayList  : 15400 ns
LinkedList : 244700 ns

------------------------------------------------
 ANALISIS HASIL
------------------------------------------------
1. Penyisipan di Awal:
   - ArrayList  : Lambat (harus menggeser elemen).
   - LinkedList : Cepat (hanya ubah referensi node).

2. Penghapusan di Tengah:
   - ArrayList  : Lambat (harus menggeser elemen).
   - LinkedList : Cepat (hanya ubah referensi node).

3. Akses Elemen Secara Acak:
   - ArrayList  : Cepat (akses langsung dengan indeks).
   - LinkedList : Lambat (harus traversal dari awal/akhir).
------------------------------------------------
```

9. Referensi

Dokumentasi resmi Java: ArrayList, LinkedList.

Buku "Effective Java" oleh Joshua Bloch.

https://www.geeksforgeeks.org/arraylist-in-java/

https://www.geeksforgeeks.org/linked-list-data-structure/

https://beginnersbook.com/2013/12/java-arraylist/

https://www.geeksforgeeks.org/arraylist-get-method-java-examples/