MODUL III

SORTING AND SEARCHING

1.1 PERMASALAHAN

1.1.1 Easy Mode

Teman Rijal bernama Eysar sedang merasa sedih atau galau. Ia mengunjungi rental PS milik Rijal, Eysar meminta bantuan kepada rizal. Eysar ingin membuat *playlist* di *Spotify* untuk menemani galau-nya. Rijal, berusaha menghibur temannya dengan bercanda dan memberikan rekomendasi lagu. Akhirnya, mereka sepakat untuk menamai *pla*

ylist tersebut "LOL *Playlist*".

```
================================================================================
||                                LOL Playlist                                ||
================================================================================
                    There is no song in the playlist yet.
================================================================================
```

Gambar 1.1 Playlist Kosong

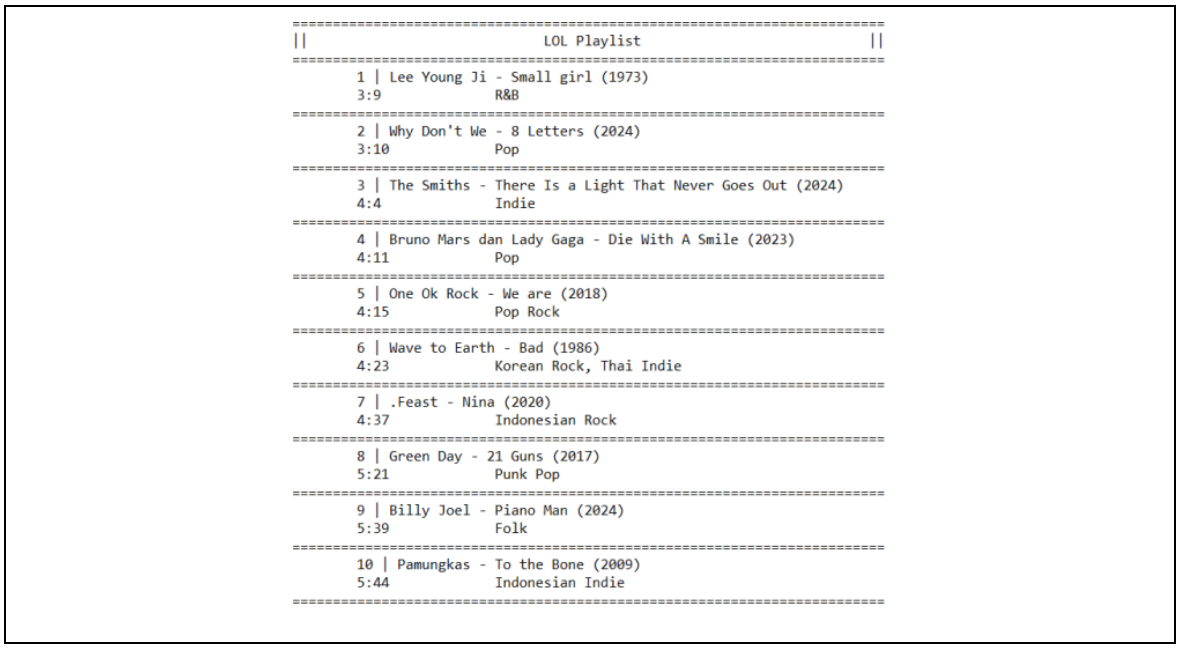
Eysar menambahkan lagu – lagu pilihan-nya kedalam *playlist*, eysar menambah 10 lagu, lagu ditambahkan menggunakan *double linkedlist*.

```
================================================================================
||                                LOL Playlist                                ||
================================================================================
        1 | Billy Joel - Piano Man (1973)
        5:39              Folk
================================================================================
        2 | Bruno Mars dan Lady Gaga - Die With A Smile (2024)
        4:11              Pop
================================================================================
        3 | .Feast - Nina (2024)
        4:37              Indonesian Rock
================================================================================
        4 | Wave to Earth - Bad (2023)
        4:23              Korean Rock, Thai Indie
================================================================================
        5 | Why Don't We - 8 Letters (2018)
        3:10              Pop
================================================================================
        6 | The Smiths - There Is a Light That Never Goes Out (1986)
        4:4               Indie
================================================================================
        7 | Pamungkas - To the Bone (2020)
        5:44              Indonesian Indie
================================================================================
        8 | One Ok Rock - We are (2017)
        4:15              Pop Rock
================================================================================
        9 | Lee Young Ji - Small girl (2024)
        3:9               R&B
================================================================================
        10 | Green Day - 21 Guns (2009)
        5:21              Punk Pop
================================================================================
```

Gambar 1.2 Setelah Membah Lagu

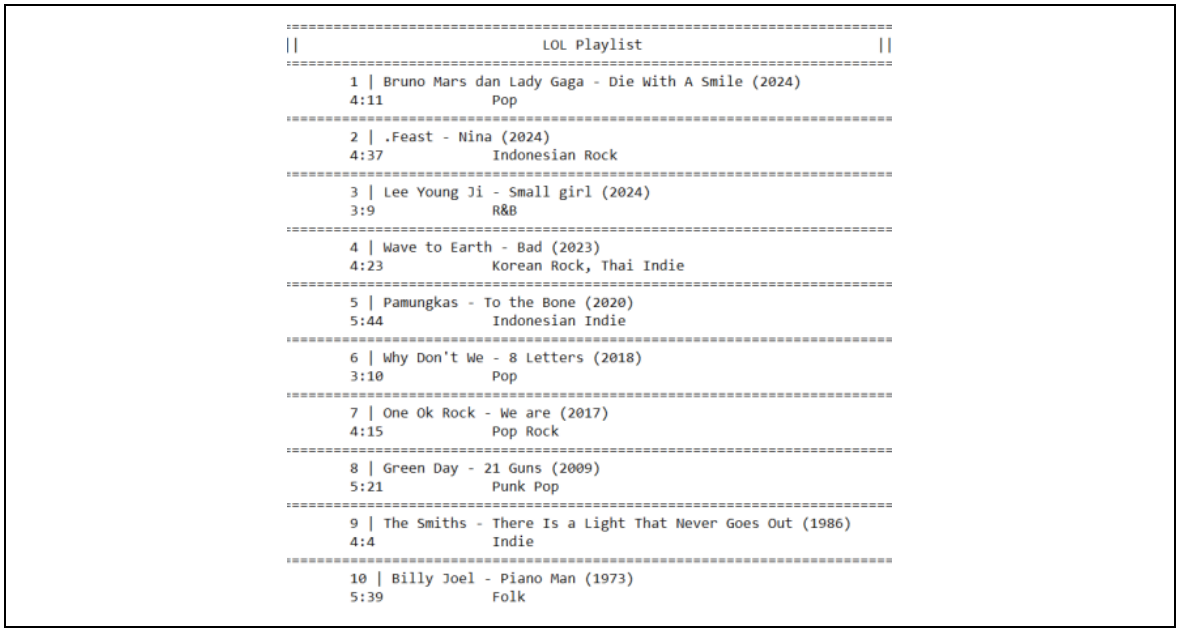
Setelah playlistnya terisi dengan sepuluh lagu, Eysar mulai berpikir untuk mengurutkan lagu-lagu ini dengan lebih spesifik lagi. Kali ini, ia ingin mengurutkannya berdasarkan

durasi lagu agar alurnya terasa lebih pas. Lagu di urutkan menggunakan algotitma bubble sort (Asc) berdasarkan durasi menit dan detik.



Gambar 1.2 Mengurutkan Berdasarkan Durasi

Namun, Eysar tiba-tiba punya ide lain, ia ingin mengurutkan lagu berdasarkan yang terbaruu di rilis, Ia pun mengaktifkan opsi pengurutan otomatis berdasarkan tahun rilis, sehingga lagu-lagu terbaru akan berada di uturan teratas *playlist*. Mengurutkan menggunakan algoritma *selection sort* (Desc) berdasarkan tahun rilis.



Gambar 1.3 Mengurutkan Berdasarkan Tahun Rilis

Akhirnya setelah cukup lama menyusun dan menata, Eysar menatap *playlist*-nya dengan senyum lega. *Playlist* itu kini rapi, terurut dari durasi, dan lagu-lagu terbaru. Namun, suatu hari saat kembali ke rental PS Rijal, ia ingin mencari tahu letak salah satu lagu favoritnya

yang ada di playlist. Eysar melakukan searching untuk mencari lagu berdasarkan jdul lagu menggunakan algoritma *linier search*.

Gambar 1.4 Mencari Lagu

Sejak saat itu, Eysar semakin senang menyusun dan memperbaiki playlist-nya, memastikan lagu-lagu favoritnya mudah ditemukan kapan saja. Rijal, yang melihat semangat temannya, hanya bisa tertawa kecil sambil kembali melayani pelanggan di rental PS

1.2 HASIL PERCOBAAN

1.2.1

1.2.2 Easy Mode

1. Algoritma
    a. Langkah pertama membuat kelas lagu
        i. Deklarasi parameter seperti judul lagu, artis, tahun rilis, durasi menit, durasi detik, genre, prev dan next.
        ii. Buat konstruktor untuk mengalokasikan data yang akan mengatur nilai tersebut saat lagu baru ditambahkan.
    b. Langkah kedua membuat kelas *playlist*
        i. Deklarasi *head* untuk awal *playlist* atau menunjuk lagu pertama yang di tambahkan dalam *playlist*.
        ii. Buat konstruktor untuk menginisialisasi playlist dengan *head* yang di atur ke *null* untuk menandakan *playlist* kosong saat awal dibuat.
        iii. Buat fungsi *insert*-Lagu untuk menambahkan lagu ke *playlist*. Buat instance lagu baru dengan parameter (judul, artis, tahun rilis, durasi, menit, detik, genre).
        iv. Jika *head* == *null* maka lagu baru akan menjadi *head*. Jika tidak maka iterasi dilakukan sampai akhir dan menambah lagu baru sebagai *next* dari lagu terakhir.
        v. Membuat fungsi urut-durasi, fungsi ini menggunakan algoritma *bubble sort* dimana pada perulangan pertama, iterasi berlanjut selama *swap* dilakukan. Lagu akan ditukar dengan lagu selanjutnya jika durasi lagu saat ini lebih besar dari legu berikutnya, jika menit nya sama, maka dibandingkan detik nya.
        vi. Membuat fungsi urut-tahun-rilis mengunakan algoritma *selection sort* untuk lagu i di *playlist*, dicari lagu *max* yang tahun rilis nya terbesar dari i hingga akhir *playlist*. Jika ditemukan lagu dengan tahun rilis lebih besar dari lagu pada posisi i, kedua lagu akan ditukar menggunakan *swap*.
        vii. Buat fungsi *swap* untuk menukar semua atribut antara dua objek lagu. Dengan menukar atribut judul, artis, tahun rilis, durasi (menit dan detik), serta genre, kedua objek lagu secara efektif saling bertukar posisi dalam *playlist*.
        viii. Membuat fungsi *Search*-lagu untuk mencari ;agu berdasarkan judul lagu, setiap lagu diperiksa apakah lagu sama dengan judul yang dicari, gunakan equalsIgnoreCase untuk mengabaikan perbedaan huruf.

ix. Buat fungsi menampilkan *playlist*, jika *playlist* kosong maka akan menampilkan pesan bahwa *playlist* kosong. Jka tidak maka setiap lagu ditampilkan beserta atributnya

2. *Source Code*

Lagu.java

```
public class Lagu {
    String judul;
    String artis;
    int tahunRilis;
    int durasiMenit;
    int durasiDetik;
    String genre;
    Lagu prev;
    Lagu next;

    public Lagu(String judul, String artis, int tahunRilis, int
durasiMenit, int durasiDetik, String genre) {
        this.judul = judul;
        this.artis = artis;
        this.tahunRilis = tahunRilis;
        this.durasiMenit = durasiMenit;
        this.durasiDetik = durasiDetik;
        this.genre = genre;
        this.prev = null;
        this.next = null;
    }
}
```

Playlist.java

```
public class Playlist {
    private Lagu head;

    public Playlist() {
        this.head = null;
    }

     public void insertLagu(String judul, String artis, int
tahunRilis, int durasiMenit, int durasiDetik, String genre) {
          Lagu newLagu = new Lagu(judul, artis, tahunRilis,
durasiMenit, durasiDetik, genre);
        if (head == null) {
            head = newLagu;
        } else {
            Lagu temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newLagu;
            newLagu.prev = temp;
        }
    }

    public void urutDurasi() {
        if (head == null) return;

        boolean swapped;
        do {
            swapped = false;
            Lagu current = head;
```

```
                while (current.next != null) {
                                    if    (current.durasiMenit    >
        current.next.durasiMenit ||
                                         (current.durasiMenit   ==
        current.next.durasiMenit     &&     current.durasiDetik     >
        current.next.durasiDetik)) {
                        swap(current, current.next);
                        swapped = true;
                    }
                    current = current.next;
                }
            } while (swapped);
        }

        public void urutTahunRilis() {
            if (head == null) return;

            for (Lagu i = head; i != null; i = i.next) {
                Lagu max = i;
                for (Lagu j = i.next; j != null; j = j.next) {
                    if (j.tahunRilis > max.tahunRilis) {
                        max = j;
                    }
                }
                if (max != i) {
                    swap(i, max);
                }
            }
        }

        private void swap(Lagu a, Lagu b) {
            String tempJudul = a.judul;
            String tempArtis = a.artis;
            int tempTahunRilis = a.tahunRilis;
            int tempDurasiMenit = a.durasiMenit;
            int tempDurasiDetik = a.durasiDetik;
            String tempGenre = a.genre;

            a.judul = b.judul;
            a.artis = b.artis;
            a.tahunRilis = b.tahunRilis;
            a.durasiMenit = b.durasiMenit;
            a.durasiDetik = b.durasiDetik;
            a.genre = b.genre;

            b.judul = tempJudul;
            b.artis = tempArtis;
            b.tahunRilis = tempTahunRilis;
            b.durasiMenit = tempDurasiMenit;
            b.durasiDetik = tempDurasiDetik;
            b.genre = tempGenre;
        }

        public Lagu searchLagu(String judul) {
            Lagu current = head;
            while (current != null) {
                if (current.judul.equalsIgnoreCase(judul)) {
                    return current;
                }
                current = current.next;
            }
```

```
                return null;
            }

            public void tampilkanPlaylist() {
                if (head == null) {
                   System.out.println("There is no song in the playlist
      yet");
                    return;
                }
                Lagu current = head;
                int index = 1;
                while (current != null) {
                     System.out.println("==============================
      ==============================");
                     System.out.printf("%4d | %s - %s (%d)\n", index,
      current.artis, current.judul, current.tahunRilis);
                    System.out.printf("    %d:%02d              %s\n",
      current .durasiMenit, current.durasiDetik, current.genre);
                    current = current.next;
                    index++;
                }
                 System.out.println("==============================
      ==========================");
            }
        }
```

Main.java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Playlist playlist = new Playlist();

        // Menambahkan 50 lagu ke dalam playlist
        tambahLagu(playlist);

        // Menampilkan playlist awal
        System.out.println("==============================");
                        System.out.println("||                    LOL
      Playlist                ||");
                System.out.println("============================");
                playlist.tampilkanPlaylist();
                System.out.println("============================");

                // Mengurutkan berdasarkan durasi
                long startTime = System.nanoTime();
                playlist.urutDurasi();
                long endTime = System.nanoTime();
                   System.out.println("Bubble  sort:  "  +  (endTime  -
      startTime) + " nanodetik");

                  System.out.println("\n#  Tampilan  setelah  diurutkan
      berdasarkan durasi (Asc):");
                System.out.println("============================");
                playlist.tampilkanPlaylist();
                System.out.println("============================");

                // Mengurutkan berdasarkan tahun rilis
                startTime = System.nanoTime();
                playlist.urutTahunRilis();
                endTime = System.nanoTime();
```

```
            System.out.println("Algoritma Selection sort: " +
(endTime - startTime) + " nanodetik");

            System.out.println("\n# Tampilan setelah diurutkan
berdasarkan tahun rilis (Desc):");
        System.out.println("==============================");
        playlist.tampilkanPlaylist();
        System.out.println("==============================");

        // Mencari lagu
        Scanner scanner = new Scanner(System.in);
        System.out.print("Judul lagu yang ingin dicari: ");
        String judul = scanner.nextLine();
        Lagu foundLagu = playlist.searchLagu(judul);
        if (foundLagu != null) {
            System.out.println("Lagu yang sesuai dengan judul
'" + judul + "' ditemukan:");
                System.out.printf("      %s  -  %s  (%d)\n",
foundLagu.artis, foundLagu.judul, foundLagu.tahunRilis);
            System.out.printf("    %d:%02d              %s\n",
foundLagu.durasiMenit,                 foundLagu.durasiDetik,
foundLagu.genre);
        } else {
            System.out.println("Lagu dengan judul '" + judul +
"' tidak ditemukan.");
        }

        scanner.close();
    }

    private static void tambahLagu(Playlist playlist) {
        // Lagu Awal
         playlist.insertLagu("Piano Man", "Billy Joel", 1979,
5, 39, "Folk");
       playlist.insertLagu("Die With A Smile", "Bruno Mars dan
Lady Gaga", 2024, 4, 11, "Pop");
          playlist.insertLagu("Nina", ".Feast", 2024, 4, 37,
"Indonesian Rock");
         playlist.insertLagu("Bad", "Wave to Earth", 2023, 4,
23, "Korean Rock, Thai Indie");
        playlist.insertLagu("8 Letters", "Why Don't We", 2018,
3, 10, "Pop");
         playlist.insertLagu("There Is a Light That Never Goes
Out", "The Smiths", 1986, 4, 4, "Indie");
         playlist.insertLagu("To the Bone", "Pamungkas", 2020,
5, 44, "Indonesian Indie");
         playlist.insertLagu("We are", "One Ok Rock", 2017, 4,
15, "Pop Rock");
       playlist.insertLagu("Small girl", "Lee Young Ji", 2024,
3, 9, "R&B");
         playlist.insertLagu("21 Guns", "Green Day", 2009, 5,
21, "Punk Pop");

        // Lana Del Rey
          playlist.insertLagu("Born to Die", "Lana Del Rey",
2012, 4, 42, "Pop");
          playlist.insertLagu("Summertime Sadness", "Lana Del
Rey", 2012, 4, 25, "Pop");
         playlist.insertLagu("Young and Beautiful", "Lana Del
Rey", 2013, 4, 28, "Pop");
```

```
        playlist.insertLagu("Love", "Lana Del Rey", 2017, 4,
22, "Pop");
       playlist.insertLagu("Doin' Time", "Lana Del Rey", 2019,
3, 36, "Reggae");
       playlist.insertLagu("Chemtrails Over the Country Club",
"Lana Del Rey", 2021, 3, 30, "Pop");
        playlist.insertLagu("Blue Banisters", "Lana Del Rey",
2021, 3, 34, "Pop");
        playlist.insertLagu("Hope is a Dangerous Thing for a
Woman Like Me to Have", "Lana Del Rey", 2019, 3, 12, "Pop");
       playlist.insertLagu("West Coast", "Lana Del Rey", 2014,
4, 16, "Pop");
        playlist.insertLagu("Ride", "Lana Del Rey", 2012, 4,
50, "Pop");

        //  Ziva Magnolya
        playlist.insertLagu("Satu", "Ziva Magnolya", 2020, 4,
10, "Pop");
            playlist.insertLagu("Cinta Luar Biasa", "Ziva
Magnolya", 2020, 4, 20, "Pop");
        playlist.insertLagu("Kisahku", "Ziva Magnolya", 2021,
3, 45, "Pop");
            playlist.insertLagu("Bukan Cinta Biasa", "Ziva
Magnolya", 2021, 4, 15, "Pop");
       playlist.insertLagu("Pupus", "Ziva Magnolya", 2021, 4,
30, "Pop");
        playlist.insertLagu("Sampai Jumpa", "Ziva Magnolya",
2022, 3, 50, "Pop");
         playlist.insertLagu("Kisah Kita", "Ziva Magnolya",
2022, 4, 5, "Pop");
        playlist.insertLagu("Bisa Aja", "Ziva Magnolya", 2022,
3, 40, "Pop");
        playlist.insertLagu("Takkan Ada Cinta yang Sia-Sia",
"Ziva Magnolya", 2022, 4, 20, "Pop");
       playlist.insertLagu("Bisa Aja", "Ziva Magnolya", 2022,
3, 40, "Pop");

        // Henry Moodie
         playlist.insertLagu("Lose Control", "Henry Moodie",
2021, 3, 30, "Pop");
       playlist.insertLagu("Love Again", "Henry Moodie", 2022,
3, 50, "Pop");
        playlist.insertLagu("I Wish", "Henry Moodie", 2022, 3,
20, "Pop");
       playlist.insertLagu("All I Want", "Henry Moodie", 2022,
3, 15, "Pop");
        playlist.insertLagu("Goodbye", "Henry Moodie", 2022,
3, 40, "Pop");
        playlist.insertLagu("Home", "Henry Moodie", 2022, 3,
30, "Pop");
       playlist.insertLagu("Never Let You Go", "Henry Moodie",
2022, 3, 35, "Pop");
         playlist.insertLagu("Better Days", "Henry Moodie",
2022, 3, 50, "Pop");
        playlist.insertLagu("Runaway", "Aurora", 2015, 4, 10,
"Pop");
        playlist.insertLagu(" The River", "Aurora", 2016, 4,
20, "Pop");

        // Raye
```

```
                playlist.insertLagu("Escapism", "Raye", 2022, 3, 40,
        "Pop");
                playlist.insertLagu("Natalie Don't", "Raye", 2021, 3,
        30, "Pop");
               playlist.insertLagu("Call on Me", "Raye", 2021, 3, 20,
        "Pop");
                playlist.insertLagu("Regardless", "Raye", 2021, 3, 50,
        "Pop");
                playlist.insertLagu("Love Me Again", "Raye", 2022, 3,
        45, "Pop");
                playlist.insertLagu("I Don't Want You", "Raye", 2022,
        3, 35, "Pop");
                playlist.insertLagu("Body Talk", "Raye", 2022, 3, 25,
        "Pop");
               playlist.insertLagu("Shhh", "Raye", 2022, 3, 15, "Pop");
                playlist.insertLagu("The Thrill", "Raye", 2022, 3, 30,
        "Pop");
                playlist.insertLagu("Ice Cream Man", "Raye", 2022, 3,
        20, "Pop");
            }
        }
```

1.3 ANALISIS DATA

1.3.1 *Sourch Code* Sesuai Soal dan Sebelum Analisis

```
public class Lagu {
    String judul;
    String artis;
    int tahunRilis;
    int durasiMenit;
    int durasiDetik;
    String genre;
    Lagu prev;
    Lagu next;

      public Lagu(String judul, String artis, int tahunRilis, int
durasiMenit, int durasiDetik, String genre) {
        this.judul = judul;
        this.artis = artis;
        this.tahunRilis = tahunRilis;
        this.durasiMenit = durasiMenit;
        this.durasiDetik = durasiDetik;
        this.genre = genre;
        this.prev = null;
        this.next = null;
    }
}
```

Script di atas adalah mendeklarasikan atribut dari lagu seperti "`judul`", "`artis`", "`tahunRilis`", "`durasiMenit`", "`durasiDetik`" dan "`gendre`" serta "`prev`" dan "`next`" untuk *double linkedlist*. Konstruktor lagu untuk membuat objek baru dari kelas lagu untuk menerima parameter atribut dan mengalokasikan nya.

```
public class Playlist {
    private Lagu head;

    public Playlist() {
        this.head = null;
    }

    public void insertLagu(String judul, String artis, int tahunRilis,
int durasiMenit, int durasiDetik, String genre) {
         Lagu newLagu = new Lagu(judul, artis, tahunRilis, durasiMenit,
durasiDetik, genre);
        if (head == null) {
            head = newLagu;
        } else {
            Lagu temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newLagu;
            newLagu.prev = temp;
        }
```

Script di atas menggunakan struktur data *double linked list* untuk menyimpan objek lagu seperti judul, artis, tahun rilis, durasi dan genre. Konstruktor menginisialisasi "`head == null`" menandakan *palylist* awal kosong. Program memeriksa apakah "`head == null`" jika

iya, maka lagu menjadi *head*. Jika tidak, maka program menelusuri dari *head* hinggga akhir *list*, dan menyimpan referensi *next* dari node terakhir menunjuk ke lagu baru yang di tambahkan.

```
public void urutDurasi() {
        if (head == null) return;

        boolean swapped;
        do {
            swapped = false;
            Lagu current = head;
            while (current.next != null) {
                if (current.durasiMenit > current.next.durasiMenit ||
                     (current.durasiMenit == current.next.durasiMenit &&
current.durasiDetik > current.next.durasiDetik)) {
                    swap(current, current.next);
                    swapped = true;
                }
                current = current.next;
            }
        } while (swapped);
    }
```

Script di atas adalah untuk mengurutkan lagu berdasarkan durasi, program menggunakan algoritma *bubble sort* secara *ascending. Loop do-while* terus berjalan selama masih ada pertukaran "`swapped = true`", menandakan bahwa *list* belum sepenuhnya terurut. *Loop while* menelusuri setiap node dalam *list*, membandingkan durasi lagu pada node saat ini dengan node berikutnya. Jika durasi lagu pada node *current* lebih besar dari node berikutnya "`current.next`", maka lagu tersebut perlu ditukar agar lagu dengan durasi lebih pendek berada di depan. Jika menit nya sama, detik nya akan di bandingkan, jika kondisi pertukaran terpenuhi maka akan di panggil fungsi *swap*.

```
    public void urutTahunRilis() {
        if (head == null) return;

        for (Lagu i = head; i != null; i = i.next) {
            Lagu max = i;
            for (Lagu j = i.next; j != null; j = j.next) {
                if (j.tahunRilis > max.tahunRilis) {
                    max = j;
                }
            }
            if (max != i) {
                swap(i, max);
            }
        }
    }
```

Script di atas adalah algoritma adalah *selection sort* untuk mengurutkan lagu berdasarkan tahun rilis. *Loop for* pertama "`Lagu i = head; i != null; i = i.next`" untuk menandai node saat ini (i) yang akan diisi dengan nilai *max* dan *max* diinisialisasi sebagai i. *Loop for* kedua "`Lagu j = i.next; j != null; j = j.next`" untuk mencari

lagu dengan tahun rilis terbesar dalam sisa list mulai dari node setelah i. Jika ditemukan "`j.tahunRilis > max.tahunRilis`", maka *max* diubah ke node j, menunjukkan bahwa lagu pada node j memiliki tahun rilis yang lebih baru. Jika *max* berbeda dari i maka panggil "`swap(i, max)`".

```
    private void swap(Lagu a, Lagu b) {
        String tempJudul = a.judul;
        String tempArtis = a.artis;
        int tempTahunRilis = a.tahunRilis;
        int tempDurasiMenit = a.durasiMenit;
        int tempDurasiDetik = a.durasiDetik;
        String tempGenre = a.genre;

        a.judul = b.judul;
        a.artis = b.artis;
        a.tahunRilis = b.tahunRilis;
        a.durasiMenit = b.durasiMenit;
        a.durasiDetik = b.durasiDetik;
        a.genre = b.genre;

        b.judul = tempJudul;
        b.artis = tempArtis;
        b.tahunRilis = tempTahunRilis;
        b.durasiMenit = tempDurasiMenit;
        b.durasiDetik = tempDurasiDetik;
        b.genre = tempGenre;
    }
```

Script di atas berfungsi untuk menukar data antara dua node Lagu, yaitu a dan b dalam linked *list* dengan cara menukar nilai atributnya, tanpa mengubah posisi node di dalam daftar. Pertama, nilai atribut node a disimpan sementara dalam variabel *temp*, sehingga node a dapat mengambil nilai dari b, dan b kemudian mengambil nilai dari *temp*.

```
public Lagu searchLagu(String judul) {
        Lagu current = head;
        while (current != null) {
            if (current.judul.equalsIgnoreCase(judul)) {
                return current;
            }
            current = current.next;
        }
        return null;
    }
```

Script "`searchLagu`" melakukan pencarian sekuensial pada linked list untuk menemukan lagu berdasarkan judul yang diberikan. Dimulai dari node head, mengecek setiap node, membandingkan atribut judul dari node tersebut dengan judul yang dicari menggunakan "`equalsIgnoreCase`" untuk mengabaikan huruf besar-kecil. Jika ditemukan kecocokan, metode ini mengembalikan node lagu yang sesuai, jika tidak ada yang cocok sampai node terakhir, metode ini mengembalikan null, menandakan bahwa lagu dengan judul tersebut tidak ditemukan dalam *playlist*.

```
public void tampilkanPlaylist() {
        if (head == null) {
            System.out.println("There is no song in the playlist yet");
            return;
        }
        Lagu current = head;
        int index = 1;
        while (current != null) {
           System.out.println("=======================================");
                 System.out.printf("%4d  |  %s  -  %s  (%d)\n",  index,
current.artis, current.judul, current.tahunRilis);
             System.out.printf("    %d:%02d              %s\n", current
.durasiMenit, current.durasiDetik, current.genre);
            current = current.next;
            index++;
        }
       System.out.println("=========================================");
    }
}
```

Script di atas menampilkan semua lagu yang ada dalam *playlist*. Jika *playlist* kosong tampilkan pesan "`There is no song in the playlist yet`" dan berhenti. Jika ada lagu di dalam *playlist*, mulai dari node *head*, tampilkan informasi setiap lagu, termasuk nomor indeks, nama artis, judul lagu, tahun rilis, durasi dalam format menit dan genre lagu. Setelah mencetak data satu lagu, iterasi melanjutkan ke node berikutnya sampai semua lagu dalam *playlist* telah ditampilkan.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Playlist playlist = new Playlist();

        // Menambahkan 50 lagu ke dalam playlist
        tambahLagu(playlist);

        // Menampilkan playlist awal
        System.out.println("=====================================");
              System.out.println("||                                  LOL
Playlist                ||");
        System.out.println("=====================================")
        playlist.tampilkanPlaylist();
        System.out.println("=====================================");

        // Mengurutkan berdasarkan durasi
        long startTime = System.nanoTime();
        playlist.urutDurasi();
        long endTime = System.nanoTime();
         System.out.println("Bubble sort: " + (endTime - startTime) + "
nanodetik");

         System.out.println("\n# Tampilan setelah diurutkan berdasarkan
durasi (Asc):");
        System.out.println("=====================================");
        playlist.tampilkanPlaylist();
        System.out.println("=====================================");

        // Mengurutkan berdasarkan tahun rilis
```

```
        startTime = System.nanoTime();
        playlist.urutTahunRilis();
        endTime = System.nanoTime();
         System.out.println("Algoritma Selection sort: " + (endTime -
startTime) + " nanodetik");

         System.out.println("\n# Tampilan setelah diurutkan berdasarkan
tahun rilis (Desc):");
        System.out.println("====================================");
        playlist.tampilkanPlaylist();
        System.out.println("====================================");

        // Mencari lagu
        Scanner scanner = new Scanner(System.in);
        System.out.print("Judul lagu yang ingin dicari: ");
        String judul = scanner.nextLine();
        Lagu foundLagu = playlist.searchLagu(judul);
        if (foundLagu != null) {
            System.out.println("Lagu yang sesuai dengan judul '" + judul
+ "' ditemukan:");
             System.out.printf("    %s - %s (%d)\n", foundLagu.artis,
foundLagu.judul, foundLagu.tahunRilis);
              System.out.printf("     %d:%02d                    %s\n",
foundLagu.durasiMenit, foundLagu.durasiDetik, foundLagu.genre);
        } else {
            System.out.println("Lagu dengan judul '" + judul + "' tidak
ditemukan.");
        }

        scanner.close();
    }

    private static void tambahLagu(Playlist playlist) {
        // Lagu Awal
         playlist.insertLagu("Piano Man", "Billy Joel", 1979, 5, 39,
"Folk");
         playlist.insertLagu("Die With A Smile", "Bruno Mars dan Lady
Gaga", 2024, 4, 11, "Pop");
         playlist.insertLagu("Nina", ".Feast", 2024, 4, 37, "Indonesian
Rock");
        playlist.insertLagu("Bad", "Wave to Earth", 2023, 4, 23, "Korean
Rock, Thai Indie");
         playlist.insertLagu("8 Letters", "Why Don't We", 2018, 3, 10,
"Pop");
        playlist.insertLagu("There Is a Light That Never Goes Out", "The
Smiths", 1986, 4, 4, "Indie");
         playlist.insertLagu("To the Bone", "Pamungkas", 2020, 5, 44,
"Indonesian Indie");
         playlist.insertLagu("We are", "One Ok Rock", 2017, 4, 15, "Pop
Rock");
         playlist.insertLagu("Small girl", "Lee Young Ji", 2024, 3, 9,
"R&B");
         playlist.insertLagu("21 Guns", "Green Day", 2009, 5, 21, "Punk
Pop");

        // Lana Del Rey
        playlist.insertLagu("Born to Die", "Lana Del Rey", 2012, 4, 42,
"Pop");
        playlist.insertLagu("Summertime Sadness", "Lana Del Rey", 2012,
4, 25, "Pop");
```

```
        playlist.insertLagu("Young and Beautiful", "Lana Del Rey", 2013,
4, 28, "Pop");
        playlist.insertLagu("Love", "Lana Del Rey", 2017, 4, 22, "Pop");
         playlist.insertLagu("Doin' Time", "Lana Del Rey", 2019, 3, 36,
"Reggae");
         playlist.insertLagu("Chemtrails Over the Country Club", "Lana
Del Rey", 2021, 3, 30, "Pop");
        playlist.insertLagu("Blue Banisters", "Lana Del Rey", 2021, 3,
34, "Pop");
        playlist.insertLagu("Hope is a Dangerous Thing for a Woman Like
Me to Have", "Lana Del Rey", 2019, 3, 12, "Pop");
         playlist.insertLagu("West Coast", "Lana Del Rey", 2014, 4, 16,
"Pop");
        playlist.insertLagu("Ride", "Lana Del Rey", 2012, 4, 50, "Pop");

        //  Ziva Magnolya
       playlist.insertLagu("Satu", "Ziva Magnolya", 2020, 4, 10, "Pop");
         playlist.insertLagu("Cinta Luar Biasa", "Ziva Magnolya", 2020,
4, 20, "Pop");
          playlist.insertLagu("Kisahku", "Ziva Magnolya", 2021, 3, 45,
"Pop");
        playlist.insertLagu("Bukan Cinta Biasa", "Ziva Magnolya", 2021,
4, 15, "Pop");
           playlist.insertLagu("Pupus", "Ziva Magnolya", 2021, 4, 30,
"Pop");
         playlist.insertLagu("Sampai Jumpa", "Ziva Magnolya", 2022, 3,
50, "Pop");
        playlist.insertLagu("Kisah Kita", "Ziva Magnolya", 2022, 4, 5,
"Pop");
         playlist.insertLagu("Bisa Aja", "Ziva Magnolya", 2022, 3, 40,
"Pop");
          playlist.insertLagu("Takkan Ada Cinta yang Sia-Sia", "Ziva
Magnolya", 2022, 4, 20, "Pop");
         playlist.insertLagu("Bisa Aja", "Ziva Magnolya", 2022, 3, 40,
"Pop");

        // Henry Moodie
       playlist.insertLagu("Lose Control", "Henry Moodie", 2021, 3, 30,
"Pop");
         playlist.insertLagu("Love Again", "Henry Moodie", 2022, 3, 50,
"Pop");
       playlist.insertLagu("I Wish", "Henry Moodie", 2022, 3, 20, "Pop");
         playlist.insertLagu("All I Want", "Henry Moodie", 2022, 3, 15,
"Pop");
          playlist.insertLagu("Goodbye", "Henry Moodie", 2022, 3, 40,
"Pop");
        playlist.insertLagu("Home", "Henry Moodie", 2022, 3, 30, "Pop");
        playlist.insertLagu("Never Let You Go", "Henry Moodie", 2022, 3,
35, "Pop");
         playlist.insertLagu("Better Days", "Henry Moodie", 2022, 3, 50,
"Pop");
         playlist.insertLagu("Runaway", "Aurora", 2015, 4, 10, "Pop");
        playlist.insertLagu(" The River", "Aurora", 2016, 4, 20, "Pop");

        // Raye
        playlist.insertLagu("Escapism", "Raye", 2022, 3, 40, "Pop");
       playlist.insertLagu("Natalie Don't", "Raye", 2021, 3, 30, "Pop");
        playlist.insertLagu("Call on Me", "Raye", 2021, 3, 20, "Pop");
        playlist.insertLagu("Regardless", "Raye", 2021, 3, 50, "Pop");
       playlist.insertLagu("Love Me Again", "Raye", 2022, 3, 45, "Pop");
```

```
          playlist.insertLagu("I Don't Want You", "Raye", 2022, 3, 35,
"Pop");
        playlist.insertLagu("Body Talk", "Raye", 2022, 3, 25, "Pop");
        playlist.insertLagu("Shhh", "Raye", 2022, 3, 15, "Pop");
        playlist.insertLagu("The Thrill", "Raye", 2022, 3, 30, "Pop");
        playlist.insertLagu("Ice Cream Man", "Raye", 2022, 3, 20, "Pop");
    }
}
```

Script "`java.util.Scanner`" adalah *library* yang digunakan untuk membaca *input* dari pengguna. Dalam fungsi main terdapat inisialisasi objek *playlist* untuk menyimpan dafttar lagu, menambah lagu dengan memanggil fungsi "`playlist.insertLagu`". menampilkan *playlist* awal, untuk menampilkan semua lagu yang di tambahkan, menampilkan pengurutan berdasarkan durasi dengan memanggil fungsi "`urutDurasi`" dan s*cript* "`endTime - startTime`" untuk menghitung selisih waktu eksekusi *bubble sort*. Kemudian menampilkan berdasarkan tahun rilis dengan memanggil fungsi "`urutTahunRilis`" secara descending menggunakan Selection Sort, mencatat waktu selesai, dan menampilkan durasi waktu yang dibutuhkan. Memanggil "`searchLagu`" untuk mencari lagu sesuai judul, dan menampilkan informasi lagu jika ditemukan atau pesan jika lagu tidak ditemukan.

1.3.2 Setelah Melakukan Analisis

Setelah melakukan analisis, pengurutan menggunakan *bubble sort* dan *marge sort* kurang efektif untuk data yang besar karena kompleksitas waktu $O(n^2)$ dan berdasarkan hasil eksekusi program *bubble sort* membutuhkaan waktu 384.400 nanodetik sedangkan *selection sort* 67400 nanodetik.

Untuk mengurutkan *playlist* berdasarkan durasi. Alternatif nya dapat diganti dengan menggunakan algoritma *merge sort* dan quick sort jauh lebih efisien untuk pengurutan data besar karena kompleksitasnya yang lebih rendah dibandingkan dengan algoritma sederhana seperti *bubble sort* atau *selection sort* yang memerlukan waktu O(n2). Keduanya memanfaatkan pembagian data dan menghindari perbandingan berulang, yang membuatnya lebih hemat waktu dan lebih praktis dalam menangani data dalam jumlah besar.

Namun *marge sort* akan membutuhkan lebih banyak memori, maka lebih tepat nya untuk menggunakan *quick sort* dengan kompleksitas O(n log n) dan memiliki kelebihan lebih cepat dan efesien dalam penggunaan memori karena mengurutkan data *in-place* sehingga tidak membutuhkan ruang tambahan besar dan mempu menangani data acak dengan sangat baik.

```
public class Playlist {
    Lagu head;

    public void insertLagu(String judul, String artis, int tahunRilis,
int durasiMenit, int durasiDetik, String genre) {
        Lagu newLagu = new Lagu(judul, artis, tahunRilis, durasiMenit,
durasiDetik, genre);
        if (head == null) {
            head = newLagu;
        } else {
            Lagu temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newLagu;
            newLagu.prev = temp;
        }
    }

    public void tampilkanPlaylist() {
        Lagu temp = head;
        while (temp != null) {
            System.out.printf("%s - %s (%d)\n", temp.artis, temp.judul,
temp.tahunRilis);
            System.out.printf("%d:%02d              %s\n",
temp.durasiMenit, temp.durasiDetik, temp.genre);
            temp = temp.next;
        }
    }

    public void urutDurasi() {
        head = mergeSortByDurasi(head);
    }

    private Lagu mergeSortByDurasi(Lagu head) {
        if (head == null || head.next == null) return head;

        Lagu middle = getMiddle(head);
        Lagu nextOfMiddle = middle.next;
        middle.next = null;

        Lagu left = mergeSortByDurasi(head);
        Lagu right = mergeSortByDurasi(nextOfMiddle);

        return sortedMergeByDurasi(left, right);
    }

    private Lagu sortedMergeByDurasi(Lagu a, Lagu b) {
        if (a == null) return b;
        if (b == null) return a;

        Lagu result;
        if (a.durasiMenit < b.durasiMenit ||
           (a.durasiMenit == b.durasiMenit && a.durasiDetik <=
b.durasiDetik)) {
            result = a;
            result.next = sortedMergeByDurasi(a.next, b);
            if (result.next != null) result.next.prev = result;
            result.prev = null;
        } else {
            result = b;
```

```
            result.next = sortedMergeByDurasi(a, b.next);
            if (result.next != null) result.next.prev = result;
            result.prev = null;
        }
        return result;
    }

    private Lagu getMiddle(Lagu head) {
        if (head == null) return head;

        Lagu slow = head, fast = head;
        while (fast.next != null && fast.next.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        return slow;
    }

    public void urutTahunRilis() {
        head = quickSortByTahunRilis(head, getLastNode(head));
    }

    private Lagu getLastNode(Lagu node) {
        while (node != null && node.next != null) {
            node = node.next;
        }
        return node;
    }

    private Lagu quickSortByTahunRilis(Lagu low, Lagu high) {
        if (high != null && low != high && low != high.next) {
            Lagu pivot = partition(low, high);
            quickSortByTahunRilis(low, pivot.prev);
            quickSortByTahunRilis(pivot.next, high);
        }
        return low;
    }

    private Lagu partition(Lagu low, Lagu high) {
        int pivot = high.tahunRilis;
        Lagu i = low.prev;

        for (Lagu j = low; j != high; j = j.next) {
            if (j.tahunRilis > pivot) {  // Descending order
                i = (i == null) ? low : i.next;
                swap(i, j);
            }
        }

        i = (i == null) ? low : i.next;
        swap(i, high);
        return i;
    }

    private void swap(Lagu a, Lagu b) {
        String tempJudul = a.judul;
        String tempArtis = a.artis;
        int tempTahun = a.tahunRilis;
        int tempDurasiMenit = a.durasiMenit;
        int tempDurasiDetik = a.durasiDetik;
        String tempGenre = a.genre;
```

```
        a.judul = b.judul;
        a.artis = b.artis;
        a.tahunRilis = b.tahunRilis;
        a.durasiMenit = b.durasiMenit;
        a.durasiDetik = b.durasiDetik;
        a.genre = b.genre;

        b.judul = tempJudul;
        b.artis = tempArtis;
        b.tahunRilis = tempTahun;
        b.durasiMenit = tempDurasiMenit;
        b.durasiDetik = tempDurasiDetik;
        b.genre = tempGenre;
    }

    public Lagu searchLagu(String judul) {
        Lagu temp = head;
        while (temp != null) {
            if (temp.judul.equalsIgnoreCase(judul)) {
                return temp;
            }
            temp = temp.next;
        }
        return null;
    }
}
```

Setelah di eksekusi algoritma *marge sort* membutuhkan waktu 210.020 nanodetik u dan algoritma quick sort memburuhkan waktu 169.100 nanodetik untuk mengurutkan lagu berdasarkan tahun rilis. algoritma sorting diganti dengan *merge sort* untuk pengurutan berdasarkan durasi dan *quick sort* untuk pengurutan berdasarkan tahun rilis, yang masing-masing memiliki kompleksitas waktu O(n log n). Ini membuat proses pengurutan lebih cepat dan efisien.