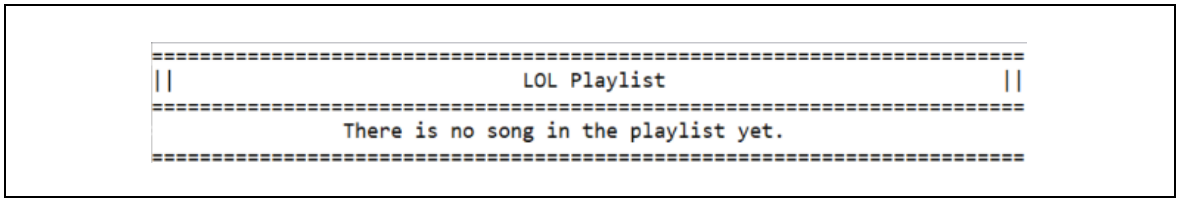
MODUL III

SORTING AND SEARCHING

1. PERMASALAHAN

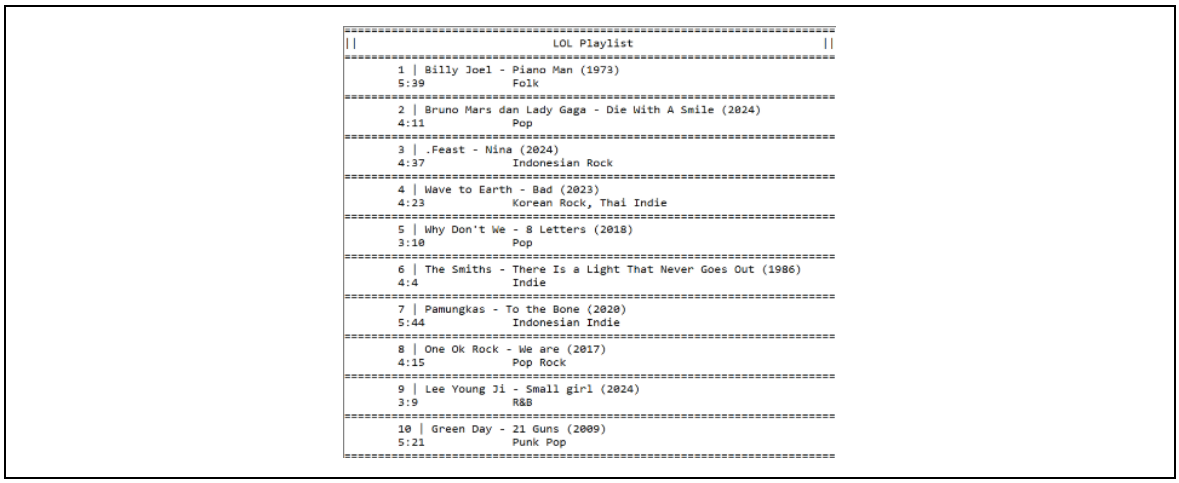
1.1 Program LOL *Playlist*

Kalian masih ingat dengan sosok bernama Rijal yang memiliki rental PS? Ia telah kembali pada cerita kali ini dengan banyak tokoh baru lain-nya. Di siang hari yang sangat terik ini. Rijal sedang sibuk di meja kasir rental PS miliknya ketika Eysar masuk. Eysar adalah teman dekat Rijal semasa perkuliahan dulu. Biasanya, Eysar datang ke sini dengan semangat untuk bermain PES, tapi kali ini wajahnya terlihat murung. “Kenapa Sar? Biasanya langsung nyari *stick*, kok sekarang malah bengong?” tanya Rijal, penasaran. Eysar duduk di sofa yang biasa digunakan pelanggan untuk menunggu giliran bermain. “Nggak tahu, Jal. Lagi galau Jal. Lagi nggak *mood* buat main” jawabnya sambil menghela napas. Rijal mengerutkan dahi. “LOL, galau kenapa tuh? Cerita dah. Kalau duduk doang di sini, nggak seru kan”. Eysar tersenyum tipis. “Kayaknya saya mau bikin *playlist* deh di Spotify. Buat nemenin kalau lagi galau begini”. Rijal tertawa kecil. “LOL, *playlist* galau nih. Biar *playlist*-nya penuh nuansa sendu kayak hatimu?” godanya. “LOL” kata Eysar sambil mengangguk, mengeluarkan ponsel dan membuka aplikasi Spotify yang masih kosong tanpa lagu di *playlist* barunya. Dengan sabar, ia mulai memilih lagu-lagu yang cocok. Rijal, yang duduk di sebelahnya sambil mengecek pemesanan pelanggan lain, sesekali memberikan rekomendasi lagu. “Apa nama *playlist* buat galaumu tu Sar?” kata Rijal yang tertarik dengan *playlist* Eysar. Eysar tertawa “Rijal Ganteng gimana?”. “LOL” kata Rijal sambil ikut tertawa. Eysar berkata sambil bersemangat “Nah itu aja gimana Jal? LOL *Playlist*?”. “LOL, iyadah” kata Rijal sambil tertawa.



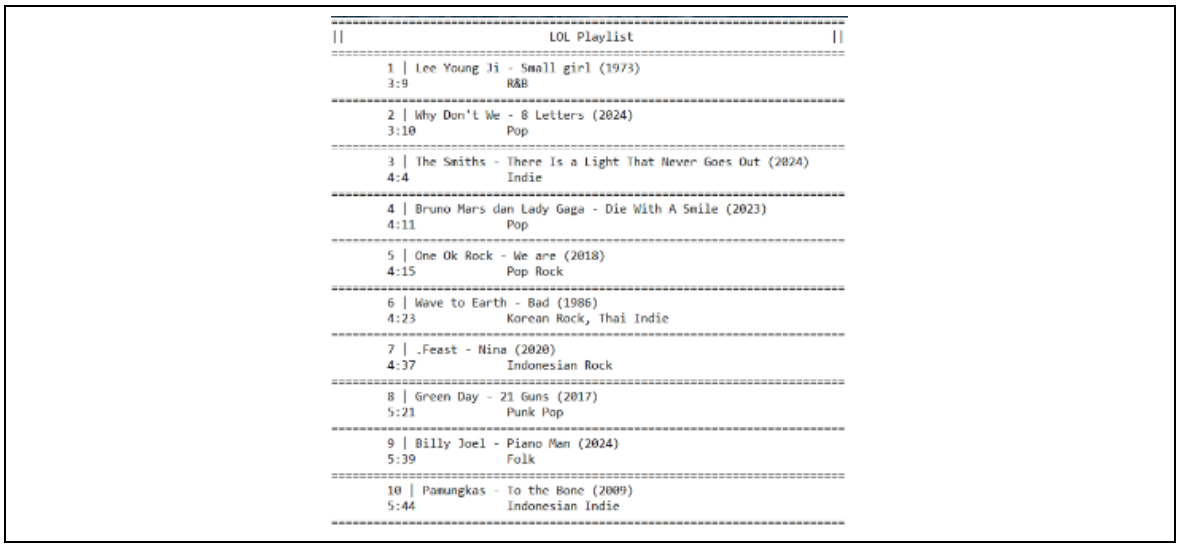
Gambar 3.1 Tampilan *Playlist*

*Tampilan di saat belum ada lagu yang ditambahkan. “Saya masukin sepuluh lagu dulu ga sih, ” kata Eysar sambil menambahkan lagu-lagu pilihan-nya ke dalam *playlist*. *Note : gunakan *double linked list*.



Gambar 3.2 Daftar *Playlist* Lol

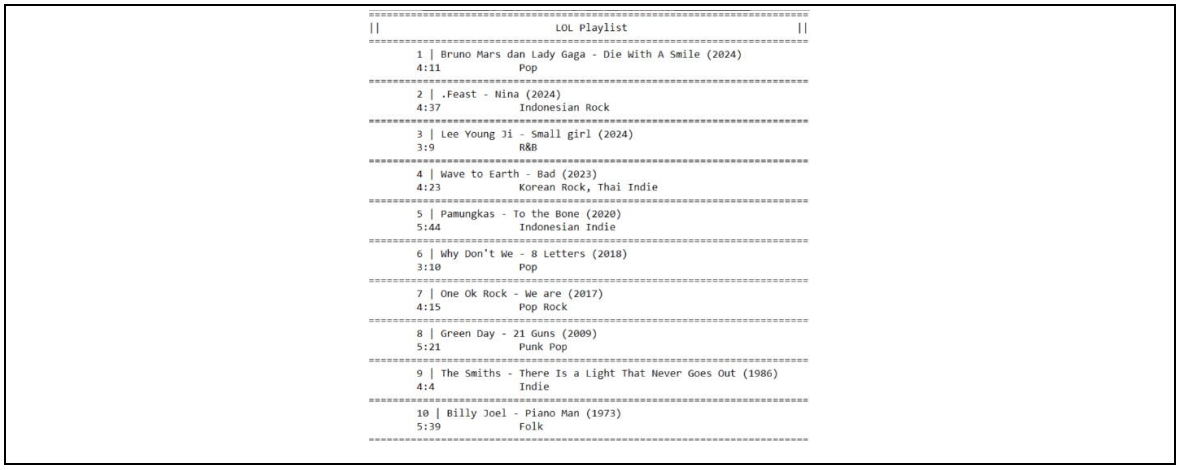
*Tampilan sesudah ada lagu yang ditambahkan *Note : atributnya ada judul lagu, nama artis, tahun rilis, durasi (dalam menit dan detik), dan genre. Setelah *playlist*-nya terisi dengan sepuluh lagu, Eysar mulai berpikir untuk mengurutkan lagu-lagu ini dengan lebih spesifik lagi. Kali ini, ia ingin mengurutkannya berdasarkan durasi lagu agar alurnya terasa lebih pas. Rijal, yang sedang membereskan stik PS di rak, mendekat lagi setelah mendengar keinginan Eysar. “Kalau urutin dari durasi, berarti lagu paling pendek dulu, baru yang lebih panjang kan?”. Eysar mengangguk. Dengan sedikit bantuan dari Rijal, ia memeriksa durasi setiap lagu di *playlist* dan mulai menyusun ulang. Satu per satu, lagu-lagu tersebut diurutkan dari yang terpendek hingga yang terpanjang. Setelah selesai, *playlist* itu terlihat lebih rapi dan teratur. *Note : Mengurutkan menggunakan algoritma *Bubble Sort* (Asc) berdasarkan durasi menit dan detik.



Gambar 3.3 Tampialn *Playlist* Berdasarkan Durasi

*Tampilan sesudah lagu di urutkan berdasarkan menit dan detik. Namun, Eysar tiba-tiba punya ide lain. “Jal, kalau saya urutinnya berdasarkan lagu terbaru yang rilis”. Rijal tersenyum mengerti. “Jadi urutannya berdasarkan tahun rilis terbaru kan Sar?”. Eysar

mengangguk lagi. Ia pun mengaktifkan opsi pengurutan otomatis berdasarkan tahun rilis, sehingga lagu-lagu terbaru akan berada di uturan teratas *playlist*. *Note : Mengurutkan menggunakan algoritma *Selection Sort* (Desc) berdasarkan tahun rilis.



Gambar 3.4 Tampialn *Playlist* Berdasarkan Tahun

*Tampilan sesudah lagu di urutkan berdasarkan tahun 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. Rijal menepuk pundaknya sambil tersenyum. "*Playlist* siap menemani galaumu tuh Sar" candanya. Eysar tertawa kecil. "LOL, makasi udah bantu ya Jal". Setelah *playlist*-nya tertata rapi, Eysar merasa *playlist* itu sudah cukup sempurna. Namun, suatu hari saat kembali ke rental PS Rijal, ia ingin mencari tahu letak salah satu lagu favoritnya yang ada di *playlist*. "Jal, saya lagi pengen dengerin satu lagu di *playlist* nih, tapi lupa letaknya di urutan ke berapa" kata Eysar sambil membuka Spotify. Rijal tertawa kecil. "LOL, lagu di *playlist*-mu kan udah ada sepuluh lebih sekarang. Kalau di-*scroll* satu-satu bisa lama, Sar". *Note: silahkan menambahkan lagu lagi secara bebas sampai 15 lagu.

Eysar mengangguk setuju, lalu mulai mengetik judul lagu yang dia ingat di kolom pencarian dalam *playlist*-nya. Setelah beberapa detik, lagu yang ia cari muncul, lengkap dengan nomor urutnya. "Yes, ketemu! Lagunya ada di urutan nomor lima!" seru Eysar dengan senyum puas. Rijal tersenyum, lalu menggodanya. "Nah, enak kan? Makanya *playlist*-mu diatur. Biar pas nyari gampang."

Gambar 3.5 Pencarian Lagu dengan Algoritma Linear *Search*

*Tampilan hasil pencarian berdasarkan judul lagu menggunakan algoritma linear *search*. Eysar tertawa. "Benar juga, Jal. Sekarang nggak ribet lagi kalau mau cari-cari lagu

favorit!" 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.

2.1 Program *Readme*

1. Lakukan penambahan data manual terserah kalian dengan ketentuan : - *Easy* : Minimal jadi 50 lagu. - *Medium* : Minimal jadi 70 entitas. - *Hard* : Minimal jadi 8 *club* dan Minimal setiap *club* terdapat 18 pemain.
2. Lakukan pengurutan kembali setelah data di tambahkan sesuai ketentuan pada no 1 menggunakan algoritma yang sudah kalian buat sesuai level, lakukan penambahan secara asc, desc ,dan *random* lalu lakukan *sorting* ke data tersebut sesuai perintah level yang kalian ambil dan bandingan efektifitas waktu setiap *sorting*. Hitung waktu yang dibutuhkan setiap jenis algoritma *sorting* dalam melakukan pengurutan data bisa menggunakan "`System.nanoTime();`" atau algoritma lain yang sesuai.
3. Lakukan analisa dari hasil pengurutan yang telah dilakukan, bandingan hasil setiap algoritma *sorting* masing-masing level manakah algoritma yang paling efektif dan efisien sesuai dengan data yang ada, jika hasil analisa dari setiap algoritma yang sudah ditentukan menunjukkan hasil yang tidak efektif/tidak efisien satu sama lain, lakukan proses pengurutan dengan algoritma berbeda dan buktikan kenapa algoritma tersebut lebih baik dari sebelumnya

2. HASIL PERCOBAAN

2.1 Program LOL *Playlist*

1. Algoritma
   a. *Start*.
   b. Membuat kelas "`Node`" untuk menyimpan data lagu.
   c. Membuat kelas "`PlaylistMusik`" yang berfungsi sebagai *double linked list* , memiliki metode "`addLagu`" untuk menambahkan lagu di akhir *playlist*.
   d. Membuat metode "`swapData`" dalam "`PlaylistMusik`" untuk menukar data antara dua node berdasarkan kriteria tertentu tanpa memindahkan posisi *node*.
   e. Membuat metode "`bubbleByDurasi`" dalam "`PlaylistMusik`" untuk mengurutkan *playlist* berdasarkan durasi lagu menggunakan algoritma *Bubble Sort*.
   f. Membuat metode "`selectionByTahunRilis`" dalam "`PlaylistMusik`" untuk mengurutkan *playlist* berdasarkan tahun rilis.
   g. Membuat metode "`searchJudul`" untuk mencari lagu dalam *playlist* berdasarkan judul yang diberikan dan menampilkan data lagu tersebut jika ditemukan.
   h. Membuat metode "`display`" untuk menampilkan semua lagu dalam *playlist*, atau pesan bahwa *playlist* kosong jika belum ada lagu yang ditambahkan.
   i. Menginisialisasi program di kelas "`Main`" dengan membuat objek "`PlaylistMusik`", menambahkan beberapa lagu, dan menampilkan *playlist* sebelum serta setelah pengurutan.
   j. Program melakukan pengurutan *playlist* berdasarkan durasi dan tahun rilis, serta mencari lagu tertentu dalam *playlist*.
   k. Finish.
2. *Source Code*

```
public class Node {
    String judul, namaArtis, genre;
    int tahunRilis, durasi;
    Node next, prev;

    // Konstruktor
    public Node(String judul, String namaArtis, String genre, int
tahunRilis, int durasi) {
        this.judul = judul;
        this.namaArtis = namaArtis;
        this.genre = genre;
        this.tahunRilis = tahunRilis;
        this.durasi = durasi;
        this.next = null;
        this.prev = null;
    }
```

```
    // Mengonversi durasi ke menit
    public int convertKeMenit() {
        return durasi / 60;
    }

     // Mendapatkan sisa detik setelah dikonversi ke menit
    public int convertKeDetik() {
        return durasi % 60;
    }
}

public class PlaylistMusik {
    Node head = null;
    Node tail = null;
    int index;

    //menambahkan lagu ke dalam playlist
    public void addLagu(String judul, String namaArtis, String genre,
int tahunRilis, int durasi) {
         Node lagu = new Node(judul, namaArtis, genre, tahunRilis,
durasi);
        if (head == null) {
            head = lagu;
            tail = lagu;
        } else {
            tail.next = lagu;
            lagu.prev = tail;
            tail = lagu;
        }
    }

    //menukar data antara dua node
    private void swapData(Node x, Node y) {
        String tempJudul = x.judul;
        x.judul = y.judul;
        y.judul = tempJudul;

        String tempNamaArtis = x.namaArtis;
        x.namaArtis = y.namaArtis;
        y.namaArtis = tempNamaArtis;

        String tempGenre = x.genre;
        x.genre = y.genre;
        y.genre = tempGenre;

        int tempTahunRilis = x.tahunRilis;
        x.tahunRilis = y.tahunRilis;
        y.tahunRilis = tempTahunRilis;

        int tempDurasi = x.durasi;
        x.durasi = y.durasi;
        y.durasi = tempDurasi;
    }

    //pengurutan Bubble Sort berdasarkan durasi
    public void bubbleByDurasi() {
        if (head == null) // Tidak ada lagu jika head kosong
        return;

        boolean statusSwap = true;
        while (statusSwap) {
```

```
            statusSwap = false;
            Node current = head;

            //memeriksa setiap node dan tukar jika urutannya salah
            while (current.next != null) {
                if (current.durasi > current.next.durasi) {
                    swapData(current, current.next);
                    statusSwap = true;
                }
                current = current.next;
            }
        }
    }

    // Pengurutan Selection Sort berdasarkan tahun rilis
    public void selectionByTahunRilis() {
        if (head == null) return;

          for (Node current = head; current != null; current =
current.next) {
            Node max = current;
             for (Node search = current.next; search != null; search
= search.next) {
                if (search.tahunRilis > max.tahunRilis) {
                    max = search;
                }
            }
            if (max != current) {
                swapData(current, max);
            }
        }
    }

    //Mencari lagu berdasarkan judul
    public void searchJudul(String text) {
        String search = text;
        index = 1;
        Node temp = head;

         System.out.println("\n\n==================================
================================================");
        System.out.println("||                                          LOL
PLAYLIST                                  ||");
         System.out.println("==========================================
================================================");
        if (head == null) {
             System.out.println("                          There is no
song in the playlist yet.               ");
              System.out.println("==================================
================================================");
        } else {
            while (temp != null) {
                if (temp.judul.equalsIgnoreCase(search)) {
                    System.out.println("Judul lagu yang ingin dicari
: " + search);
                   System.out.println("Lagu yang sesuai dengan judul
'" + search + "' ditemukan pada posisi ke - " + index);
                     System.out.println("========================
============================================================");
                     System.out.printf("       %d | %s - %s (%d)\n",
index, temp.namaArtis, temp.judul, temp.tahunRilis);
```

```
                    System.out.printf("       %d:%02d           %s\n",
temp.convertKeMenit(), temp.convertKeDetik(), temp.genre);
                     System.out.println("============================
============================================================");
                    return;
                } else if (temp.next == null) {
                     System.out.println("                    There is
no song that you search in the playlist.                    ");
                     System.out.println("============================
============================================================");
                }
                index++;
                temp = temp.next;
            }
        }
    }

    //Menampilkan semua lagu di dalam playlist
    public void display() {
        Node temp = head;
        index = 1;
         System.out.println("\n\n================================
================================================");
        System.out.println("||                                        LOL
PLAYLIST                                ||");
         System.out.println("========================================
================================================");
        if (head == null) {
            System.out.println("                          There is no
song in the playlist yet.         ");
             System.out.println("================================
================================================");
            return;
        }
        while (temp != null) {
             System.out.printf("       %d | %s - %s (%d)\n", index,
temp.namaArtis, temp.judul, temp.tahunRilis);
              System.out.printf("        %d:%02d             %s\n",
temp.convertKeMenit(), temp.convertKeDetik(), temp.genre);
             System.out.println("================================
================================================");
            temp = temp.next;
            index++;
        }
    }
}

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

        // Menampilkan playlist kosong
        songList.display();

         songList.addLagu("A Sky Full of Stars", "Coldplay", "Pop",
2014, 264);
         songList.addLagu("Thinking Out Loud", "Ed Sheeran", "Pop",
2014, 281);
        songList.addLagu("Talking to the Moon", "Bruno Mars", "Pop",
2010, 228);
```

```
        songList.addLagu("Cinta Luar Biasa", "Andmesh Kamaleng",
"Pop", 2018, 240);
        songList.addLagu("Laskar Pelangi", "Nidji", "Pop", 2008,
215);
       songList.addLagu("Kemesraan", "Iwan Fals", "Pop", 1988, 230);
       songList.addLagu("Akad", "Payung Teduh", "Pop", 2017, 265);
        songList.addLagu("Blinding Lights", "The Weeknd", "Pop",
2020, 200);
       songList.addLagu("Watermelon Sugar", "Harry Styles", "Pop",
2019, 174);
       songList.addLagu("Separuh Aku", "Noah", "Pop", 2012, 243);
       songList.addLagu("Meraih Bintang", "Via Vallen", "Dangdut",
2018, 202);
        songList.addLagu("Peaches", "Justin Bieber", "R&B", 2021,
198);
       songList.addLagu("Drivers License", "Olivia Rodrigo", "Pop",
2021, 242);
        songList.addLagu("Rumah Kita", "God Bless", "Rock", 1988,
276);
        songList.addLagu("Tinggal Kenangan", "Gaby", "Pop", 2007,
244);
       songList.addLagu("Good 4 U", "Olivia Rodrigo", "Pop", 2021,
178);
      songList.addLagu("Levitating", "Dua Lipa feat. DaBaby", "Pop",
2020, 203);
       songList.addLagu("Bintang Kehidupan", "Nike Ardilla", "Pop",
1990, 270);
       songList.addLagu("Pupus", "Dewa 19", "Pop", 2000, 298);
        songList.addLagu("Stay", "The Kid LAROI & Justin Bieber",
"Pop", 2021, 142);
       songList.addLagu("Kangen", "Dewa 19", "Pop", 1992, 300);
       songList.addLagu("As It Was", "Harry Styles", "Pop", 2022,
167);
        songList.addLagu("Hanya Rindu", "Andmesh Kamaleng", "Pop",
2019, 210);
       songList.addLagu("Butter", "BTS", "Pop", 2021, 164);
       songList.addLagu("Easy On Me", "Adele", "Pop", 2021, 224);
        songList.addLagu("Januari", "Glenn Fredly", "Pop", 2003,
276);
        songList.addLagu("Save Your Tears", "The Weeknd", "Pop",
2020, 215);
       songList.addLagu("Sempurna", "Andra and the Backbone", "Pop",
2007, 250);
       songList.addLagu("Love Story", "Taylor Swift", "Country Pop",
2008, 235);
       songList.addLagu("All Too Well (10 Minute Version)", "Taylor
Swift", "Pop", 2021, 600);
       songList.addLagu("Happier Than Ever", "Billie Eilish", "Pop",
2021, 298);
        songList.addLagu("Deen Assalam", "Sabyan", "Religi", 2018,
300);
        songList.addLagu("Bad Habits", "Ed Sheeran", "Pop", 2021,
231);
       songList.addLagu("Dance Monkey", "Tones and I", "Pop", 2019,
209);
        songList.addLagu("Shape of You", "Ed Sheeran", "Pop", 2017,
233);
       songList.addLagu("Rolling in the Deep", "Adele", "Pop", 2010,
228);
       songList.addLagu("Senorita", "Shawn Mendes & Camila Cabello",
"Pop", 2019, 191);
```

```
        songList.addLagu("Don't Start Now", "Dua Lipa", "Pop", 2019,
183);
         songList.addLagu("Shallow", "Lady Gaga & Bradley Cooper",
"Pop", 2018, 215);
         songList.addLagu("Someone Like You", "Adele", "Pop", 2011,
285);
        songList.addLagu("Rockstar", "Post Malone feat. 21 Savage",
"Hip Hop", 2017, 218);
        songList.addLagu("Sunflower", "Post Malone & Swae Lee", "Hip
Hop", 2018, 158);
        songList.addLagu("Perfect", "Ed Sheeran", "Pop", 2017, 263);
        songList.addLagu("Memories", "Maroon 5", "Pop", 2019, 189);
       songList.addLagu("Believer", "Imagine Dragons", "Rock", 2017,
204);
      songList.addLagu("Despacito", "Luis Fonsi feat. Daddy Yankee",
"Reggaeton", 2017, 229);
        songList.addLagu("Kasih Putih", "Glenn Fredly", "Pop", 2001,
278);
        songList.addLagu("Bento", "Iwan Fals", "Rock", 1989, 320);
          songList.addLagu("Somewhere Over The Rainbow", "Israel
Kamakawiwo'ole", "Hawaiian", 1993, 210);
         songList.addLagu("Photograph", "Ed Sheeran", "Pop", 2014,
258);
        songList.addLagu("See You Again", "Wiz Khalifa feat. Charlie
Puth", "Hip Hop", 2015, 229);
        // Menampilkan playlist setelah penambahan lagu
        songList.display();

        // Mengurutkan lagu berdasarkan durasi dengan Bubble Sort
        songList.bubbleByDurasi();
        songList.display();

        // Mengurutkan lagu berdasarkan tahun rilis dengan Selection
Sort
        songList.selectionByTahunRilis();
        songList.display();

        // Mencari lagu berdasarkan judul tertentu
        songList.searchJudul("Blinding Lights");
        songList.searchJudul("Stay");
        songList.searchJudul("Akad");
        songList.searchJudul("Memories");
        songList.searchJudul("Despacito");
    }}
```

3. ANALISIS DATA

3.1 Program LOL *Playlist*

```
public class Node {
    String judul, namaArtis, genre;
    int tahunRilis, durasi;
    Node next, prev;
    // Konstruktor
     public Node(String judul, String namaArtis, String genre, int
tahunRilis, int durasi) {
        this.judul = judul;
        this.namaArtis = namaArtis;
        this.genre = genre;
        this.tahunRilis = tahunRilis;
        this.durasi = durasi;
        this.next = null;
```

```
        this.prev = null;
    }

    // Mengonversi durasi ke menit
    public int convertKeMenit() {
        return durasi / 60;
    }

     // Mendapatkan sisa detik setelah dikonversi ke menit
    public int convertKeDetik() {
        return durasi % 60;
    }}
```

Code di atas merupakan kelas "`Node`" yang merepresentasikan *node* dalam struktur data *linked list* yang menyimpan informasi tentang lagu. Setiap objek "`Node`" berisi beberapa atribut seperti "`judul`", "`Node`", "`namaArtis`", "`genre`", "`tahunRilis`" dan "`durasi`". Atribut "`next`" dan "`prev`" digunakan untuk menghubungkan node ini dengan node berikutnya "`next`" dan node sebelumnya "`prev`" dalam *linked list*, memungkinkan *list* berjalan dua arah atau *doubly linked list*. Konstruktor kelas "`Node`" menginisialisasi setiap node baru dengan "`judul`", "`Node`", "`namaArtis`", "`genre`", "`tahunRilis`" dan "`durasi`", sementara "`next`" dan "`prev`" diatur ke null. Selain itu, kelas ini memiliki dua metode, "`convertKeMenit`" dan "`convertKeDetik`" yang masing-masing mengonversi durasi lagu ke dalam menit dan menghitung sisa detik setelah dikonversi.

```
public class PlaylistMusik {
    Node head = null;
    Node tail = null;
    int index;

    //menambahkan lagu ke dalam playlist
     public void addLagu(String judul, String namaArtis, String genre,
int tahunRilis, int durasi) {
        Node lagu = new Node(judul, namaArtis, genre, tahunRilis, durasi);
         if (head == null) {
             head = lagu;
             tail = lagu;
         } else {
             tail.next = lagu;
             lagu.prev = tail;
             tail = lagu;
         }}

    //menukar data antara dua node
    private void swapData(Node x, Node y) {
        String tempJudul = x.judul;
        x.judul = y.judul;
        y.judul = tempJudul;

        String tempNamaArtis = x.namaArtis;
        x.namaArtis = y.namaArtis;
        y.namaArtis = tempNamaArtis;

        String tempGenre = x.genre;
```

```
        x.genre = y.genre;
        y.genre = tempGenre;

        int tempTahunRilis = x.tahunRilis;
        x.tahunRilis = y.tahunRilis;
        y.tahunRilis = tempTahunRilis;

        int tempDurasi = x.durasi;
        x.durasi = y.durasi;
        y.durasi = tempDurasi;
    }

    //pengurutan Bubble Sort berdasarkan durasi
    public void bubbleByDurasi() {
        if (head == null) // Tidak ada lagu jika head kosong
        return;

        boolean statusSwap = true;
        while (statusSwap) {
            statusSwap = false;
            Node current = head;

            //memeriksa setiap node dan tukar jika urutannya salah
            while (current.next != null) {
                if (current.durasi > current.next.durasi) {
                    swapData(current, current.next);
                    statusSwap = true;
                }
                current = current.next;
            }}}

    // Pengurutan Selection Sort berdasarkan tahun rilis
    public void selectionByTahunRilis() {
        if (head == null) return;

       for (Node current = head; current != null; current = current.next)
{
            Node max = current;
             for (Node search = current.next; search != null; search =
search.next) {
                if (search.tahunRilis > max.tahunRilis) {
                    max = search;
                }
            }
            if (max != current) {
                swapData(current, max);
            }}}

    //Mencari lagu berdasarkan judul
    public void searchJudul(String text) {
        String search = text;
        index = 1;
        Node temp = head;

         System.out.println("\n\n=================================
=============================================");
          System.out.println("||                                          LOL
PLAYLIST                                  ||");
         System.out.println("=============================================
=========================================");
        if (head == null) {
```

```
            System.out.println("                              There is no song
in the playlist yet.                ");
            System.out.println("==================================
================================================");
        } else {
            while (temp != null) {
                if (temp.judul.equalsIgnoreCase(search)) {
                    System.out.println("Judul lagu yang ingin dicari : "
+ search);
                    System.out.println("Lagu yang sesuai dengan judul '"
+ search + "' ditemukan pada posisi ke - " + index);
                    System.out.println("==============================
============================================================");
                      System.out.printf("        %d | %s - %s (%d)\n",
index, temp.namaArtis, temp.judul, temp.tahunRilis);
                     System.out.printf("       %d:%02d            %s\n",
temp.convertKeMenit(), temp.convertKeDetik(), temp.genre);
                    System.out.println("==============================
============================================================");
                    return;
                } else if (temp.next == null) {
                     System.out.println("                    There is no
song that you search in the playlist.                  ");
                    System.out.println("==============================
============================================================");
                }
                index++;
                temp = temp.next;
            }}}

    //Menampilkan semua lagu di dalam playlist
    public void display() {
        Node temp = head;
        index = 1;
        System.out.println("\n\n==================================
================================================");
         System.out.println("||                                          LOL
PLAYLIST                                  ||");
        System.out.println("==================================
========================================");
        if (head == null) {
            System.out.println("                              There is no song
in the playlist yet.          ");
            System.out.println("==================================
================================================");
            return;
        }
        while (temp != null) {
              System.out.printf("        %d | %s - %s (%d)\n", index,
temp.namaArtis, temp.judul, temp.tahunRilis);
                System.out.printf("         %d:%02d              %s\n",
temp.convertKeMenit(), temp.convertKeDetik(), temp.genre);
            System.out.println("==================================
================================================");
            temp = temp.next;
            index++;
        }}}
```

Code di atas merupakan kelas "`PlaylistMusik`" yang merepresentasikan *playlist* lagu dengan menggunakan struktur data *doubly linked list*. Di mana setiap lagu disimpan sebagai

objek "Node" yang berisi informasi seperti "judul", "Node", "namaArtis", "genre", "tahunRilis" dan "durasi" dalam detik. *Playlist* memiliki metode "addLagu" untuk menambahkan lagu ke akhir *playlist*. Metode "swapData" untuk menukar data dua *node* (digunakan dalam pengurutan) dan dua metode pengurutan yaitu "bubbleByTahunRilis" (*bubble sort* yang berdasarkan durasi lagu secara *ascending*) dan "selectionByTahunRilis" (*selection sort* yang berdasarkan tahun rilis secara *descending*). Metode "searchJudul" untuk mencari lagu berdasarkan judul dan menampilkan informasi jika ditemukan, serta metode "display" menampilkan semua lagu di dalam *playlist* yang menampilkan urutan "namaArtis", "judul", "tahunRilis", "durasi" (dalam menit dan detik) dan "genre".

```
public class Main {
    public static void main(String[] args) {
        PlaylistMusik songList = new PlaylistMusik();

        // Menampilkan playlist kosong
        songList.display();

        songList.addLagu("A Sky Full of Stars", "Coldplay", "Pop", 2014,
264);
        songList.addLagu("Thinking Out Loud", "Ed Sheeran", "Pop", 2014,
281);
         songList.addLagu("Talking to the Moon", "Bruno Mars", "Pop",
2010, 228);
        songList.addLagu("Cinta Luar Biasa", "Andmesh Kamaleng", "Pop",
2018, 240);
        songList.addLagu("Laskar Pelangi", "Nidji", "Pop", 2008, 215);
        songList.addLagu("Kemesraan", "Iwan Fals", "Pop", 1988, 230);
        songList.addLagu("Akad", "Payung Teduh", "Pop", 2017, 265);
         songList.addLagu("Blinding Lights", "The Weeknd", "Pop", 2020,
200);
       songList.addLagu("Watermelon Sugar", "Harry Styles", "Pop", 2019,
174);
        songList.addLagu("Separuh Aku", "Noah", "Pop", 2012, 243);
       songList.addLagu("Meraih Bintang", "Via Vallen", "Dangdut", 2018,
202);
        songList.addLagu("Peaches", "Justin Bieber", "R&B", 2021, 198);
         songList.addLagu("Drivers License", "Olivia Rodrigo", "Pop",
2021, 242);
        songList.addLagu("Rumah Kita", "God Bless", "Rock", 1988, 276);
        songList.addLagu("Tinggal Kenangan", "Gaby", "Pop", 2007, 244);
       songList.addLagu("Good 4 U", "Olivia Rodrigo", "Pop", 2021, 178);
         songList.addLagu("Levitating", "Dua Lipa feat. DaBaby", "Pop",
2020, 203);
          songList.addLagu("Bintang Kehidupan", "Nike Ardilla", "Pop",
1990, 270);
        songList.addLagu("Pupus", "Dewa 19", "Pop", 2000, 298);
        songList.addLagu("Stay", "The Kid LAROI & Justin Bieber", "Pop",
2021, 142);
        songList.addLagu("Kangen", "Dewa 19", "Pop", 1992, 300);
        songList.addLagu("As It Was", "Harry Styles", "Pop", 2022, 167);
        songList.addLagu("Hanya Rindu", "Andmesh Kamaleng", "Pop", 2019,
210);
```

```
        songList.addLagu("Butter", "BTS", "Pop", 2021, 164);
        songList.addLagu("Easy On Me", "Adele", "Pop", 2021, 224);
        songList.addLagu("Januari", "Glenn Fredly", "Pop", 2003, 276);
         songList.addLagu("Save Your Tears", "The Weeknd", "Pop", 2020,
215);
         songList.addLagu("Sempurna", "Andra and the Backbone", "Pop",
2007, 250);
         songList.addLagu("Love Story", "Taylor Swift", "Country Pop",
2008, 235);
         songList.addLagu("All Too Well (10 Minute Version)", "Taylor
Swift", "Pop", 2021, 600);
         songList.addLagu("Happier Than Ever", "Billie Eilish", "Pop",
2021, 298);
        songList.addLagu("Deen Assalam", "Sabyan", "Religi", 2018, 300);
        songList.addLagu("Bad Habits", "Ed Sheeran", "Pop", 2021, 231);
         songList.addLagu("Dance Monkey", "Tones and I", "Pop", 2019,
209);
       songList.addLagu("Shape of You", "Ed Sheeran", "Pop", 2017, 233);
         songList.addLagu("Rolling in the Deep", "Adele", "Pop", 2010,
228);
         songList.addLagu("Senorita", "Shawn Mendes & Camila Cabello",
"Pop", 2019, 191);
         songList.addLagu("Don't Start Now", "Dua Lipa", "Pop", 2019,
183);
        songList.addLagu("Shallow", "Lady Gaga & Bradley Cooper", "Pop",
2018, 215);
        songList.addLagu("Someone Like You", "Adele", "Pop", 2011, 285);
        songList.addLagu("Rockstar", "Post Malone feat. 21 Savage", "Hip
Hop", 2017, 218);
         songList.addLagu("Sunflower", "Post Malone & Swae Lee", "Hip
Hop", 2018, 158);
        songList.addLagu("Perfect", "Ed Sheeran", "Pop", 2017, 263);
        songList.addLagu("Memories", "Maroon 5", "Pop", 2019, 189);
         songList.addLagu("Believer", "Imagine Dragons", "Rock", 2017,
204);
        songList.addLagu("Despacito", "Luis Fonsi feat. Daddy Yankee",
"Reggaeton", 2017, 229);
         songList.addLagu("Kasih Putih", "Glenn Fredly", "Pop", 2001,
278);
        songList.addLagu("Bento", "Iwan Fals", "Rock", 1989, 320);
            songList.addLagu("Somewhere Over The Rainbow", "Israel
Kamakawiwo'ole", "Hawaiian", 1993, 210);
        songList.addLagu("Photograph", "Ed Sheeran", "Pop", 2014, 258);
         songList.addLagu("See You Again", "Wiz Khalifa feat. Charlie
Puth", "Hip Hop", 2015, 229);
        // Menampilkan playlist setelah penambahan lagu
        songList.display();

        // Mengurutkan lagu berdasarkan durasi dengan Bubble Sort
        songList.bubbleByDurasi();
        songList.display();

       // Mengurutkan lagu berdasarkan tahun rilis dengan Selection Sort
        songList.selectionByTahunRilis();
        songList.display();

        // Mencari lagu berdasarkan judul tertentu
        songList.searchJudul("Blinding Lights");
        songList.searchJudul("Stay");
        songList.searchJudul("Akad");
        songList.searchJudul("Memories");
```

```
        songList.searchJudul("Despacito");
    }}
```

Script di atas merupakan kelas "`Main`" yang berfungsi sebagai titik awal untuk menjalankan "`PlaylistMusik`" yang mengelola *playlist* lagu menggunakan *linked list*. Dalam metode "`main`", objek "`PlaylistMusik`" dibuat lalu metode "`display`" dijalankan untuk menampilkan pesan bahwa *playlist* masih kosong. Selanjutnya, metode "`addLagu`" digunakan untuk menambahkan sejumlah lagu ke *playlist* dengan detail seperti "`judul`", "`namaArtis`", "`genre`", "`tahunRilis`" dan "`durasi`". Setelah penambahan lagu, metode "`display`" menampilkan daftar semua lagu dalam *playlist*. *Playlist* kemudian diurutkan dua kali dengan metode pengurutan berbeda yaitu pertama, "`bubbleByDurasi`" untuk mengurutkan lagu berdasarkan durasi secara *ascending* menggunakan algoritma *Bubble Sort*. Lalu metode kedua yaitu "`selectionByTahunRilis`" untuk mengurutkan lagu berdasarkan tahun rilis secara *descending* menggunakan algoritma *Selection Sort*. Setelah setiap pengurutan, isi *playlist* ditampilkan. Akhirnya, metode "`searchJudul`" mencari dan menampilkan informasi tentang beberapa lagu tertentu berdasarkan judulnya seperti "`Blinding Lights`", "`Stay`", dan "`Despacito`" yang menunjukkan apakah lagu tersebut ada dalam playlist beserta detailnya.

3.2 *Readme*

Berdasarkan praktikum yang telah dikerjakan, analisis yang kita dapatkan dari praktikum itu adalah bahwa waktu eksekusi *ascending* selama 74680500 *nanosecond*, waktu eksekusi *descending* selama 19005200 *nanosecond*, waktu eksekusi *random* selama 29153800 *nanosecond*. Terbukti bahwa dari hasil pengukuran waktu eksekusi, *descending* adalah yang paling efisien dengan waktu eksekusi 19005200 *nanodetik*, diikuti oleh *random* dengan waktu 29153800 *nanodetik*, dan *ascending* yang paling tidak efisien dengan waktu 74680500 *nanodetik*. Hal ini menunjukkan bahwa *descending* memerlukan waktu paling sedikit untuk menyelesaikan tugas dibandingkan dua lainnya.

Descending Sort lebih efektif daripada *Ascending Sort*. Berdasarkan hasil waktu, *Descending Sort* (*Selection Sort*) lebih efisien dibandingkan dengan *Ascending Sort* (*Bubble Sort*). Meskipun keduanya menggunakan algoritma pengurutan dengan kompleksitas waktu yang sama, *Selection Sort* dapat menangani pengurutan lebih efisien karena tidak memerlukan pertukaran yang banyak.

Random Shuffle paling efisien dalam waktu. Namun, untuk pengacakan data yang tidak memerlukan urutan tertentu, *Random Shuffle* lebih efisien dalam hal waktu.

Descending Sort (*Selection Sort*) terbukti lebih efektif dibandingkan dengan *Ascending Sort* (*Bubble Sort*), karena lebih cepat dalam kasus pengurutan terbalik (*descending*).

Pengacakan (*Random Shuffle*) meskipun efisien dalam hal waktu, hanya digunakan ketika urutan data tidak penting, dan tidak menggantikan kebutuhan akan pengurutan terstruktur.