MODUL III
SORTING DAN SEARCHING

3.1 PERMASALAHAN

3.1.1 LOL *Playlist*

1. Buat *playlist* lagu menggunakan *double linked list*. Atributnya ada judul lagu, nama artis, tahun rilis, durasi (dalam menit dan detik), dan *genre*.
2. Tambahkan 10 lagu terlebih dahulu, lalu tampilkan sesudah lagu diurutkan berdasarkan menit dan detik menggunakan algoritma *bubble sort* (asc) berdasarkan durasi menit dan detik dan juga tampilkan sesudah lagu diurutkan berdasarkan menit dan detik menggunakan algoritma *selection sort* (desc) berdasarkan tahun rilis.
3. Tambahkan 5 lagu lagi sehingga terdapat 15 lagu di dalam *playlist*.
4. Tampilan hasil pencarian berdasarkan judul lagu menggunakan algoritman *linear search*

3.1.2 *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 satusama lain, lakukan proses pengurutan dengan algoritma berbeda dan buktikan kenapa algoritma tersebut lebih baik dari sebelumnya.

3.2 HASIL PERCOBAAN

3.2.1 LOL Playlist

1. Algoritma
   a. Mulai program.
   b. Buat kelas *Node* yang berfungsi untuk menampung *node-node* lagu dan *pointer next* serta *prev* untuk menghubungkan ke *node-node* berikutnya atau sebelumnya. *Node* yang disimpan dalam kelas lagu berupa data dari lagu.
   c. Buat kelas *double linked list* yang berfungsi untuk mengelola daftar lagu. *double linked list* memiliki atribut *head* dan *tail* yang berguna untuk menunjuk ke *node* lagu yang paling depan dan belakang. *double linked list* juga terdapat beberapa *method* seperti, *add*, *addfirst*, display, *bubble* sort, *selection* sort, *linear serach* dan *swap node*.
   d. Buta kelas Main untuk mengeksekusi program. Di dalam *main* dibuat objek *double linked list* untuk mengelola daftar lagu dan objek. Dan kemudian implementasikan *method* yang sudah dibuat di *double linked list* sebelumnya.
   e. Program selesai.
2. *Source Code*

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

    public Node(String namaArtis, String judul, String genre, int
tahunRilis, int menit, int detik) {
        this.namaArtis = namaArtis;
        this.judul = judul;
        this.genre = genre;
        this.tahunRilis = tahunRilis;
        this.menit = menit;
        this.detik = detik;
        this.next = null;
        this.prev = null;
    }
}

public class LinkedListDouble {
    Node head;
    Node tail;

    public void add(String namaArtis, String judul, String genre,
int tahunRilis, int menit, int detik) {
        Node lagu = new Node(namaArtis, judul, genre, tahunRilis,
menit, detik);
        if (head == null) {
            head = tail = lagu;
        } else {
```

```
            tail.next = lagu;
            lagu.prev = tail;
            tail = lagu;
        }
    }

    public void display() {
        int count = 1;
        Node current = head;


System.out.println("==========================================
====================================");
        System.out.println("||                                        LOL
Playlist                                ||");
        if (current == null) {
            System.out.println("\t\t\tThere is no song in the
playlist yet.");
            return;
        }
        while (current != null) {

System.out.println("==========================================
====================================");
            System.out.printf("%15d | %s (%d)\n", count,
current.namaArtis + " - " + current.judul, current.tahunRilis);
            System.out.printf("%15d:%02d %25s\n", current.menit,
current.detik, current.genre);
            count++;
            current = current.next;
        }

System.out.println("==========================================
====================================");
    }

    public void bubbleSort() {
        boolean swapped;
        Node temp;

        do {
            swapped = false;
            temp = this.head;

            while (temp != null && temp.next != null) {

                if (temp.menit > temp.next.menit || (temp.menit ==
temp.next.menit && temp.detik > temp.next.detik)) {

                    swapNodeData(temp, temp.next);
                    swapped = true;
                }
                temp = temp.next;
            }
        } while (swapped);
    }

    public void displaysearch(String targetTitle) {
        int count = 1;
        Node curr = this.head;
```

```
System.out.println("=============================================
===================================");
        System.out.println("||                                        LOL
Playlist                                ||");
        while (curr != null) {
            if (curr.judul.equals(targetTitle)) {
                System.out

.println("=============================================================
==========================");
                System.out.println("Judul lagu yang di cari : " +
curr.judul);
                System.out.println(
                        "Lagu yang sesuai dengan judul '" +
curr.judul + "' ditemukan pada posisi ke - " + count);
                System.out

.println("=============================================================
==========================");
                System.out.printf("%15d |  %s (%d)\n", count,
curr.namaArtis + " - " + curr.judul, curr.tahunRilis);
                System.out.printf("%15d:%02d %25s\n", curr.menit,
curr.detik, curr.genre);
            }
            curr = curr.next;
            count++;
        }

System.out.println("=============================================
===================================");
    }

    public void selectionSort() {
        Node start = head;

        while (start != null) {
            Node maxNode = start;
            Node curr = start.next;

            while (curr != null) {
                if (curr.tahunRilis > maxNode.tahunRilis) {
                    maxNode = curr;
                }
                curr = curr.next;
            }

            if (maxNode != start) {
                swapNodeData(maxNode, start);
            }
            start = start.next;
        }
    }

    public void swapNodeData(Node node1, Node node2) {
        int tempYear = node1.tahunRilis;
        node1.tahunRilis = node2.tahunRilis;
        node2.tahunRilis = tempYear;

        String tempNamaArtis = node1.namaArtis;
        node1.namaArtis = node2.namaArtis;
```

```
        node2.namaArtis = tempNamaArtis;

        String tempJudul = node1.judul;
        node1.judul = node2.judul;
        node2.judul = tempJudul;

        String tempGenre = node1.genre;
        node1.genre = node2.genre;
        node2.genre = tempGenre;

        int tempMenit = node1.menit;
        node1.menit = node2.menit;
        node2.menit = tempMenit;

        int tempDetik = node1.detik;
        node1.detik = node2.detik;
        node2.detik = tempDetik;
    }
}

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

        buatlagu.add("Billy Joel", "Piano Man", "Folk", 1973,
5,39);
        buatlagu.add("Bruno Mars dan Lady Gaga", "Die With A
SWmile", "pop", 2024, 4,11);
        buatlagu.add("Feast", "Nina", "Indonesian Rock", 2024,
4,37);
        buatlagu.add("Wave to Earth", "Bad", "Korean Rock, thai
indie", 2023, 4,24);
        buatlagu.add("Why don't we", "8 Latter", "Pop", 2018,
3,10);
        buatlagu.add("The Smiths", "There is a light that never
goes out", "Pop", 1986, 4,4);
        buatlagu.add("Pamungkas", "To The Bone", "Indonesian
Indie", 2020, 5,44);
        buatlagu.add("One Ok Rock", "We Are", "Pop Rock", 2018,
4,15);
        buatlagu.add("Le Young ji", "Small Girl", "R&B", 2024,
3,9);
        buatlagu.add("Green Day", "21 Guns", "Punk Pop", 2009,
5,21);
        buatlagu.add("Selena Gomez", "Lose You to Love Me", "Pop",
2019, 3, 27);
        buatlagu.add("Camila Cabello", "Havana", "Pop", 2017, 3,
36);
        buatlagu.add("Maroon 5", "Sugar", "Pop", 2014, 3, 55);
        buatlagu.add("The Chainsmokers", "Closer", "EDM", 2016, 4,
4);
        buatlagu.add("Avicii", "Wake Me Up", "EDM", 2013, 4, 7);
        buatlagu.add("Zedd", "Clarity", "EDM", 2012, 4, 31);
        buatlagu.add("David Guetta", "Titanium", "EDM", 2011, 4,
5);
        buatlagu.add("Daft Punk", "Get Lucky", "Funk", 2013, 6, 7);
        buatlagu.add("Sia", "Chandelier", "Pop", 2014, 3, 36);
        buatlagu.add("Katy Perry", "Firework", "Pop", 2010, 3, 47);
        buatlagu.add("Rihanna", "Diamonds", "Pop", 2012, 3, 45);
        buatlagu.add("Kendrick Lamar", "HUMBLE.", "Hip Hop", 2017,
2, 57);
```

```
        buatlagu.add("Jay-Z", "Empire State of Mind", "Hip Hop",
2009, 4, 36);

        buatlagu.add("Led Zeppelin", "Immigrant Song", "Rock",
1970, 2, 26);
        buatlagu.add("Elton John", "Tiny Dancer", "Soft Rock",
1971, 6, 12);
        buatlagu.add("The Rolling Stones", "Angie", "Rock", 1972,
4, 33);
        buatlagu.add("Stevie Wonder", "Superstition", "Funk", 1972,
4, 26);
        buatlagu.add("Queen", "Bohemian Rhapsody", "Rock", 1975, 5,
55);
        buatlagu.add("ABBA", "Dancing Queen", "Pop", 1976, 3, 52);
        buatlagu.add("Bee Gees", "Stayin' Alive", "Disco", 1977, 4,
45);
        buatlagu.add("The Clash", "London Calling", "Punk Rock",
1979, 3, 19);
        buatlagu.add("Michael Jackson", "Billie Jean", "Pop", 1982,
4, 54);
        buatlagu.add("Prince", "Purple Rain", "Rock", 1984, 8, 41);
        buatlagu.add("Bon Jovi", "Livin' on a Prayer", "Rock",
1986, 4, 11);
        buatlagu.add("U2", "With or Without You", "Rock", 1987, 4,
55);
        buatlagu.add("Guns N' Roses", "Sweet Child o' Mine",
"Rock", 1987, 5, 56);
        buatlagu.add("Metallica", "One", "Heavy Metal", 1988, 7,
24);
        buatlagu.add("Nirvana", "Smells Like Teen Spirit",
"Grunge", 1991, 5, 1);
        buatlagu.add("Red Hot Chili Peppers", "Under the Bridge",
"Alternative Rock", 1992, 4, 24);
        buatlagu.add("Dr. Dre", "Nuthin' but a 'G' Thang", "Hip
Hop", 1992, 3, 58);
        buatlagu.add("Radiohead", "Creep", "Alternative Rock",
1993, 3, 58);
        buatlagu.add("Oasis", "Wonderwall", "Britpop", 1995, 4,
18);
        buatlagu.add("2Pac", "California Love", "Hip Hop", 1995, 6,
25);
        buatlagu.add("The Notorious B.I.G.", "Hypnotize", "Hip
Hop", 1997, 3, 50);
        buatlagu.add("Blink-182", "All the Small Things", "Pop
Punk", 1999, 2, 48);
        buatlagu.add("Eminem", "Stan", "Hip Hop", 2000, 6, 44);
        buatlagu.add("OutKast", "Ms. Jackson", "Hip Hop", 2000, 4,
30);
        buatlagu.add("Linkin Park", "In the End", "Nu Metal", 2001,
3, 36);
        buatlagu.add("The White Stripes", "Seven Nation Army",
"Rock", 2003, 3, 51);
        buatlagu.add("Green Day", "Boulevard of Broken Dreams",
"Rock", 2004, 4, 20);
        buatlagu.add("Coldplay", "Fix You", "Alternative Rock",
2005, 4, 55);
        buatlagu.add("Arctic Monkeys", "I Bet You Look Good on the
Dancefloor", "Indie Rock", 2006, 2, 53);
        buatlagu.add("Amy Winehouse", "Rehab", "Soul", 2006, 3,
33);
        buatlagu.add("Rihanna", "Umbrella", "Pop", 2007, 4, 36);
```

```
        buatlagu.add("Kings of Leon", "Sex on Fire", "Rock", 2008,
3, 23);
        buatlagu.add("Beyoncé", "Single Ladies (Put a Ring on It)",
"Pop", 2008, 3, 13);
        buatlagu.add("Jay-Z", "Empire State of Mind", "Hip Hop",
2009, 4, 36);
        buatlagu.add("Lady Gaga", "Bad Romance", "Pop", 2009, 4,
54);
        buatlagu.add("Adele", "Rolling in the Deep", "Pop", 2010,
3, 48);
        buatlagu.add("Kanye West", "Runaway", "Hip Hop", 2010, 9,
8);
        buatlagu.add("Bruno Mars", "Just the Way You Are", "Pop",
2010, 3, 40);
        buatlagu.add("Gotye", "Somebody That I Used to Know",
"Alternative", 2011, 4, 4);
        buatlagu.add("Lana Del Rey", "Video Games", "Indie Pop",
2011, 4, 42);
        buatlagu.add("Imagine Dragons", "Radioactive", "Alternative
Rock", 2012, 3, 6);
        buatlagu.add("Macklemore & Ryan Lewis", "Thrift Shop", "Hip
Hop", 2012, 3, 55);
        buatlagu.add("Daft Punk", "Get Lucky", "Disco", 2013, 6,
9);
        buatlagu.add("Pharrell Williams", "Happy", "Pop", 2013, 3,
53);
        buatlagu.add("Ed Sheeran", "Thinking Out Loud", "Pop",
2014, 4, 41);
        buatlagu.add("Taylor Swift", "Shake It Off", "Pop", 2014,
3, 39);
        buatlagu.add("The Weeknd", "Can't Feel My Face", "R&B",
2015, 3, 35);
        buatlagu.add("Justin Bieber", "Sorry", "Pop", 2015, 3, 20);
        buatlagu.add("Kendrick Lamar", "HUMBLE.", "Hip Hop", 2017,
2, 57);
        buatlagu.add("Billie Eilish", "Bad Guy", "Pop", 2019, 3,
14);

        buatlagu.add("Prince", "Purple Rain", "Rock", 1984, 8, 41);
        buatlagu.add("Kanye West", "Runaway", "Hip Hop", 2010, 9,
8);
        buatlagu.add("Metallica", "One", "Heavy Metal", 1988, 7,
24);
        buatlagu.add("Eminem", "Stan", "Hip Hop", 2000, 6, 44);
        buatlagu.add("Daft Punk", "Get Lucky", "Disco", 2013, 6,
9);
        buatlagu.add("2Pac", "California Love", "Hip Hop", 1995, 6,
25);
        buatlagu.add("Elton John", "Tiny Dancer", "Soft Rock",
1971, 6, 12);
        buatlagu.add("Adele", "Rolling in the Deep", "Pop", 2010,
3, 48);
        buatlagu.add("OutKast", "Ms. Jackson", "Hip Hop", 2000, 4,
30);
        buatlagu.add("Lana Del Rey", "Video Games", "Indie Pop",
2011, 4, 42);
        buatlagu.add("Beyoncé", "Single Ladies (Put a Ring on It)",
"Pop", 2008, 3, 13);
        buatlagu.add("Lady Gaga", "Bad Romance", "Pop", 2009, 4,
54);
        buatlagu.add("Bee Gees", "Stayin' Alive", "Disco", 1977, 4,
```

```
45);
        buatlagu.add("The Weeknd", "Can't Feel My Face", "R&B",
2015, 3, 35);
        buatlagu.add("Justin Bieber", "Sorry", "Pop", 2015, 3, 20);
        buatlagu.add("Rihanna", "Umbrella", "Pop", 2007, 4, 36);
        buatlagu.add("Jay-Z", "Empire State of Mind", "Hip Hop",
2009, 4, 36);
        buatlagu.add("Led Zeppelin", "Immigrant Song", "Rock",
1970, 2, 26);
        buatlagu.add("Green Day", "Boulevard of Broken Dreams",
"Rock", 2004, 4, 20);
        buatlagu.add("Coldplay", "Fix You", "Alternative Rock",
2005, 4, 55);
        buatlagu.add("Bon Jovi", "Livin' on a Prayer", "Rock",
1986, 4, 11);
        buatlagu.add("The Rolling Stones", "Angie", "Rock", 1973,
4, 33);
        buatlagu.add("The Clash", "London Calling", "Punk Rock",
1979, 3, 19);
        buatlagu.add("Michael Jackson", "Billie Jean", "Pop", 1982,
4, 54);
        buatlagu.add("Queen", "Bohemian Rhapsody", "Rock", 1975, 5,
55);
        buatlagu.add("The Notorious B.I.G.", "Hypnotize", "Hip
Hop", 1997, 3, 50);
        buatlagu.add("Arctic Monkeys", "I Bet You Look Good on the
Dancefloor", "Indie Rock", 2006, 2, 53);
        buatlagu.add("Gotye", "Somebody That I Used to Know",
"Alternative", 2011, 4, 4);
        buatlagu.add("Imagine Dragons", "Radioactive", "Alternative
Rock", 2012, 3, 6);
        buatlagu.add("U2", "With or Without You", "Rock", 1987, 4,
55);
        buatlagu.add("Bruno Mars", "Just the Way You Are", "Pop",
2010, 3, 40);
        buatlagu.add("Taylor Swift", "Shake It Off", "Pop", 2014,
3, 39);
        buatlagu.add("Guns N' Roses", "Sweet Child o' Mine",
"Rock", 1987, 5, 56);
        buatlagu.add("Radiohead", "Creep", "Alternative Rock",
1993, 3, 58);
        buatlagu.add("Oasis", "Wonderwall", "Britpop", 1995, 4,
18);
        buatlagu.add("Nirvana", "Smells Like Teen Spirit",
"Grunge", 1991, 5, 1);
        buatlagu.add("ABBA", "Dancing Queen", "Pop", 1976, 3, 52);
        buatlagu.add("Red Hot Chili Peppers", "Under the Bridge",
"Alternative Rock", 1992, 4, 24);
        buatlagu.add("Dr. Dre", "Nuthin' but a 'G' Thang", "Hip
Hop", 1992, 3, 58);
        buatlagu.add("Kings of Leon", "Sex on Fire", "Rock", 2008,
3, 23);
        buatlagu.add("Amy Winehouse", "Rehab", "Soul", 2006, 3,
33);
        buatlagu.add("Kendrick Lamar", "HUMBLE.", "Hip Hop", 2017,
2, 57);
        buatlagu.add("Blink-182", "All the Small Things", "Pop
Punk", 1999, 2, 48);
        buatlagu.add("Linkin Park", "In the End", "Nu Metal", 2001,
3, 36);
        buatlagu.add("Macklemore & Ryan Lewis", "Thrift Shop", "Hip
```

```
Hop", 2012, 3, 55);
        buatlagu.add("Pharrell Williams", "Happy", "Pop", 2013, 3,
53);
        buatlagu.add("Ed Sheeran", "Thinking Out Loud", "Pop",
2014, 4, 41);
        buatlagu.add("Billie Eilish", "Bad Guy", "Pop", 2019, 3,
14);

        buatlagu.add("Coldplay", "Fix You", "Alternative Rock",
2005, 4, 55);
        buatlagu.add("Adele", "Rolling in the Deep", "Pop", 2010,
3, 48);
        buatlagu.add("Nirvana", "Smells Like Teen Spirit",
"Grunge", 1991, 5, 1);
        buatlagu.add("Taylor Swift", "Shake It Off", "Pop", 2014,
3, 39);
        buatlagu.add("Guns N' Roses", "Sweet Child o' Mine",
"Rock", 1987, 5, 56);
        buatlagu.add("Daft Punk", "Get Lucky", "Disco", 2013, 6,
9);
        buatlagu.add("Imagine Dragons", "Radioactive", "Alternative
Rock", 2012, 3, 6);
        buatlagu.add("Beyoncé", "Single Ladies (Put a Ring on It)",
"Pop", 2008, 3, 13);
        buatlagu.add("Amy Winehouse", "Rehab", "Soul", 2006, 3,
33);
        buatlagu.add("Led Zeppelin", "Immigrant Song", "Rock",
1970, 2, 26);
        buatlagu.add("U2", "With or Without You", "Rock", 1987, 4,
55);
        buatlagu.add("OutKast", "Ms. Jackson", "Hip Hop", 2000, 4,
30);
        buatlagu.add("Arctic Monkeys", "I Bet You Look Good on the
Dancefloor", "Indie Rock", 2006, 2, 53);
        buatlagu.add("Michael Jackson", "Billie Jean", "Pop", 1982,
4, 54);
        buatlagu.add("Prince", "Purple Rain", "Rock", 1984, 8, 41);
        buatlagu.add("Lana Del Rey", "Video Games", "Indie Pop",
2011, 4, 42);
        buatlagu.add("Jay-Z", "Empire State of Mind", "Hip Hop",
2009, 4, 36);
        buatlagu.add("Radiohead", "Creep", "Alternative Rock",
1993, 3, 58);
        buatlagu.add("Blink-182", "All the Small Things", "Pop
Punk", 1999, 2, 48);
        buatlagu.add("Queen", "Bohemian Rhapsody", "Rock", 1975, 5,
55);
        buatlagu.add("Pharrell Williams", "Happy", "Pop", 2013, 3,
53);
        buatlagu.add("Eminem", "Stan", "Hip Hop", 2000, 6, 44);
        buatlagu.add("Red Hot Chili Peppers", "Under the Bridge",
"Alternative Rock", 1992, 4, 24);
        buatlagu.add("Rihanna", "Umbrella", "Pop", 2007, 4, 36);
        buatlagu.add("Gotye", "Somebody That I Used to Know",
"Alternative", 2011, 4, 4);

        buatlagu.display();

        System.out.println("\nMengurutkan berdasarkan durasi
(Bubble Sort):");
        long startTimeBubbleSort = System.nanoTime();
```

```
        buatlagu.bubbleSort();
        long endTimeBubbleSort = System.nanoTime();
        buatlagu.display();
        System.out.println("Waktu eksekusi Bubble Sort: " +
(endTimeBubbleSort - startTimeBubbleSort) + " nanoseconds");

        System.out.println("\nMengurutkan berdasarkan tahun
(Selection Sort):");
        long startTimeSelectionSort = System.nanoTime();
        buatlagu.selectionSort();
        long endTimeSelectionSort = System.nanoTime();
        buatlagu.display();
        System.out.println("Waktu eksekusi Selection Sort: " +
(endTimeSelectionSort - startTimeSelectionSort) + " nanoseconds");

        buatlagu.displaysearch("21 Guns");

    }
}
```

3.3 ANALISIS DATA

3.3.1 LOL *Playlist*

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

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

Script "`public class Lagu{`" adalah deklarasi kelas yang dapat diakses kelas lain, dimana kelas ini bernama Lagu yang digunakan untuk membuat struktur *node* dalam *double linked list*, dimana setiap *node* merepresentasikan suatu lagu dalam daftar lagu. *Script* "`String namaArti, judul, genre; int tahunRilis, menit, detik; Lagu next, prev;`" berfungsi untuk mendeklarasikan variabel data yang menyimpan informasi lagu dan *pointer* "`next, prev`" sebagai navigasi dalam *double linked list*. *Script* "`public Lagu String namaArtis, String judul, String genre, int tahunRilis, int menit, int detik)}`adalah *constructor* atau *method* khusus yang akan dipanggil saat membuat objek baru dari kelas Lagu, dengan parameter data bertipe "`String`" dan "`int`". *Script* "`this.namaArtis = namaArtis; this.judul = judul; this.genre = genre; this.tahunRilis = tahunRilis; this.menit = menit; this.detik = detik;`"

`this.next = null; this.prev = null;`" berfungsi untuk menginisialisasi variabel data dari objek yang dibuat serta menandakan bahwa *node* baru belum terhubung dengan *node* lainnya.

```
public class LinkedListDouble {
    Node head;
    Node tail;
```

Script "`public class DoubleLinkedList {`" adalah deklarasi kelas yang dapat diakses oleh kelas lain, dimana kelas ini bernama *DoubleLinkedList* yang digunakan untuk mengelola lagu. *Script* "`Lagu head, tail;`" berfungsi untuk mendeklarasikan variabel data yang berguna untuk menjadi acuan untuk *node* pertama dan terakhir dari kelas Lagu.

```
public void add(String namaArtis, String judul, String genre, int
tahunRilis, int menit, int detik) {
      Node lagu = new Node(namaArtis, judul, genre, tahunRilis, menit,
      detik);
      if (head == null) {
          head = tail = lagu;
      } else {
          tail.next = lagu;
          lagu.prev = tail;
          tail = lagu;
      }
}
```

Script "`public void add(String namaArtis, String judul, String genre, int tahunRilis, int menit, int detik)`" berfungsi untuk menambahkan data lagu baru ke dalam *list*. Jika *list* masih kosong, maka lagu baru ini akan menjadi *node* pertama, dengan *head* dan *tail* sama-sama menunjuk ke "`laguBaru`". Namun, jika *list* sudah berisi data, lagu baru ditambahkan di akhir *list*. "`tail.next`" diatur ke laguBaru untuk menghubungkan *node* terakhir yang lama dengan node baru. Selanjutnya, "`laguBaru.prev`" diatur ke *tail* untuk menghubungkan lagu baru dengan *node* sebelumnya, dan *tail* diperbarui menjadi "`laguBaru`"sebagai *node* terakhir di *list*.

```
public void display() {
        int count = 1;
        Node current = head;


System.out.println("=============================================================================
        System.out.println("||                                        LOL Playlist
||");
        if (current == null) {
            System.out.println("\t\t\tThere is no song in the playlist yet.");
            return;
        }
        while (current != null) {

System.out.println("=============================================================================
            System.out.printf("%15d | %s (%d)\n", count, current.namaArtis + " - " + curr
current.tahunRilis);
```

```
            System.out.printf("%15d:%02d %25s\n", current.menit, current.detik, current.g
            count++;
            current = current.next;
        }

System.out.println("=============================================================================
```

Script "`public void display()`" berfungsi untuk menampilkan seluruh data lagu yang ada di salam *linked list* dengan format yang rapi. Seperti yang dicontohkan di soal jurnal yang ada.

```
public void bubbleSort() {
        boolean swapped;
        Node temp;

        do {
            swapped = false;
            temp = this.head;

            while (temp != null && temp.next != null) {

                if (temp.menit > temp.next.menit || (temp.menit
==temp.next.menit && temp.detik > temp.next.detik)) {

                    swapNodeData(temp, temp.next);
                    swapped = true;
                }
                temp = temp.next;
            }
        } while (swapped);
}
```

Script "`public void bubbleSort()`" berfungsi untuk mengurutkan daftar lagu di dalam *double linked list* berdasarkan durasi lagu (menit dan detik) dari yang terpendek ke terpanjang. Metode ini menggunakan algoritma *bubble sort*, di mana setiap *node* akan dibandingkan dengan *node* berikutnya dan ditukar posisinya jika durasi lebih besar.

```
public void displaysearch(String targetTitle) {
        int count = 1;
        Lagu curr = this.head;

        System.out.println("========================================
========================================");
        System.out.println("||
LOL Playlist                                ||");
        while (curr != null) {
            if (curr.judul.equals(targetTitle)) {
                System.out
                        .println("==================================
==============================================");
                System.out.println("Judul lagu yang di cari : "
+ curr.judul);
                System.out.println(
                        "Lagu yang sesuai dengan judul '" + curr.judul
+ "' ditemukan pada posisi ke - " + count);
                System.out
                        .println("==================================
==============================================");
```

```
                System.out.printf("%15d |  %s (%d)\n", count,
curr.namaArtis + " - " + curr.judul, curr.tahunRilis);
                System.out.printf("%15d:%02d %25s\n", curr.menit,
curr.detik, curr.genre);
            }
            curr = curr.next;
            count++;
        }
        System.out.println("=======================================
========================================");
    }
```

Script "`public void displaysearch(targetTitle)`" berfungsi untuk mencari dan juga menampilkan informasi lagu berdasarkan judul yang dicari di dalam *linked list*. Seperti yang dicontohkan di soal jurnal yang ada.

```
public void selectionSort() {
        Lagu start = head;

        while (start != null) {
            Lagu maxNode = start;
            Lagu curr = start.next;

            while (curr != null) {
                if (curr.tahunRilis > maxNode.tahunRilis) {
                    maxNode = curr;
                }
                curr = curr.next;
            }

            if (maxNode != start) {
                swapNodeData(maxNode, start);
            }
            start = start.next;
        }
    }
```

Script "`public void selectionSort()`" berfungsi untuk mengurutkan daftar lagu di dalam *double linked list* berdasarkan tahun rilis dari yang terbaru ke terlama. Metode ini menggunakan algoritma *selection sort*, di mana setiap *node* akan dibandingkan dengan *node-node* setelahnya untuk menemukan *node* dengan tahun rilis terbesar, lalu menukar data kedua *node* jika diperlukan.

```
public void swapNodeData(Lagu node1, Lagu node2) {
        int tempYear = node1.tahunRilis;
        node1.tahunRilis = node2.tahunRilis;
        node2.tahunRilis = tempYear;

        String tempNamaArtis = node1.namaArtis;
        node1.namaArtis = node2.namaArtis;
        node2.namaArtis = tempNamaArtis;

        String tempJudul = node1.judul;
        node1.judul = node2.judul;
        node2.judul = tempJudul;

        String tempGenre = node1.genre;
```

```
        node1.genre = node2.genre;
        node2.genre = tempGenre;

        int tempMenit = node1.menit;
        node1.menit = node2.menit;
        node2.menit = tempMenit;

        int tempDetik = node1.detik;
        node1.detik = node2.detik;
        node2.detik = tempDetik;
    }
}
```

Script "`public void swapNodeDatta(Lagu node1, Lagu node2)`" berfungsi untuk menukar data antara dua *node* dalam *linked list*. Dalam hal ini, data yang ditukar meliputi tahun rilis, nama artis, judul, *genre*, serta durasi lagu (menit dan detik).

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

Script "`public class Main{`" adalah deklarasi kelas yang dapat diakses oleh kelas lain, dimana kelas ini bernama Main yang digunakan untuk menguji dan menjalankan fitur-fitur dari kelas Lagu. *Method* "`main()`" adalah titik masuk eksekusi program, *method* ini yang pertama kali dipanggil ketika program dijalankan. *Script* "`DoubleLinkedList lagu = new DoubleLinkedList()`" merupakan *pembuatan* objek baru dari kelas DoubleLinkedList yang akan digunakan untuk menyimpan lagu dalam *list*.

```
lagu.add("Billy Joel", "Piano Man", "Folk", 1973, 5, 39);
lagu.add("Bruno Mars dan Lady Gaga", "Die With A SWmile", "pop", 2024,
4, 11);
lagu.add(".Feast", "Nina", "Indonesian Rock", 2024, 4, 37);
lagu.add("Wave to Earth", "Bad", "Korean Rock, thai indie", 2023, 4,
24);
lagu.add("Why Don't We", "8 Latter", "Pop", 2018, 3, 10);
lagu.add("The Smiths", "There Is a Light That Never Goes Out", "Indie",
1986, 4, 4);
lagu.add("Pamungkas", "To The Bone", "Indonesian Indie", 2020, 5, 44);
        lagu.add("One Ok Rock", "We Are", "Pop Rock", 2018, 4, 15);
lagu.add("Le Young ji", "Small girl", "R&B", 2024, 3, 9);
lagu.add("Green Day", "21 Guns", "Punk Pop", 2009, 5, 21);
lagu.display();
System.out.println();
```

Script di atas adalah *method* "`add()`" yang berfungsi untuk menambahkan 10 lagu ke dalam daftar lagu. Setiap kali *method* ini dipanggil, pemanggil menyertakan informasi tentang lagu yang terdiri dari nama artis, judul lagu, *genre*, tahun rilis, dan durasi (dalam menit dan detik). Data lagu yang diberikan akan disimpan dalam *node* baru yang kemudian ditambahkan ke dalam *linked list*. Dan dapat ditampilkan dengan menggunakan *method* "`display()`".

```
System.out.println("Setelah diututkan berdasarkan durasi (Bubble
Sort):");
lagu.bubbleSort();
lagu.display();
System.out.println();
```

Script di atas berfungsi untuk mengurutkan daftar lagu berdasarkan durasi menggunakan algoritma *bubble sort*. Setiap kali "`bubbleSort()`" dipanggil, proses pengurutan dilakukan dengan membandingkan durasi lagu (menit dan detik) dan menukar posisi lagu yang durasinya lebih besar dengan lagu yang durasinya lebih kecil. Dan dapat ditampilkan dengan menggunakan *method* "`display()`".

```
System.out.println("Setelah diurutkan berdasarkan tahun rilis
(Selection Sort):");
lagu.selectionSort();
lagu.display();
System.out.println();
```

Script di atas berfungsi untuk mengurutkan daftar lagu berdasarkan durasi menggunakan algoritma *selection sort*. Setiap kali "`selectionSort()`" dipanggil, proses pengurutan dilakukan dengan mencari lagu dengan tahun rilis terbesar di bagian yang belum terurut dan menukarnya dengan lagu yang ada di posisi saat ini. Proses ini diulang hingga seluruh daftar lagu terurut berdasarkan tahun rilis, dari yang terbaru ke yang terlama. Dan dapat ditampilkan dengan menggunakan *method* "`display()`".

```
lagu.add("Selena Gomez", "Lose You to Love Me", "Pop", 2019, 3, 27);
lagu.add("Camila Cabello", "Havana", "Pop", 2017, 3, 36);
lagu.add("Maroon 5", "Sugar", "Pop", 2014, 3, 55);
lagu.add("The Chainsmokers", "Closer", "EDM", 2016, 4, 4);
lagu.add("Avicii", "Wake Me Up", "EDM", 2013, 4, 7);

System.out.println("Setelah ditambahkan 5 lagu:");
lagu.display();
System.out.println();
```

Script di atas adalah *method* "`add()`" yang berfungsi untuk menambahkan 5 lagu lagi ke dalam daftar lagu. Setiap kali *method* ini dipanggil, pemanggil menyertakan informasi tentang lagu yang terdiri dari nama artis, judul lagu, *genre*, tahun rilis, dan durasi (dalam menit dan detik). Data lagu yang diberikan akan disimpan dalam *node* baru yang kemudian ditambahkan ke dalam *linked list*. Dan dapat ditampilkan dengan menggunakan *method* "`display()`".

```
System.out.println("Setelah dicari berdasarkan judul:");
lagu.displaysearch("We Are");
```

Script di atas berfungsi untuk mencari dan menampilkan lagu berdasarkan judul yang dicari, yaitu "`We Are`". Setiap kali "`displaysearch()`" dipanggil, proses pencarian dilakukan dengan memeriksa setiap *node* dalam *linked list* untuk menemukan lagu yang

judulnya sesuai dengan yang dicari. Jika lagu dengan judul yang sesuai ditemukan, maka informasi tentang lagu tersebut akan ditampilkan, termasuk nama artis, judul, tahun rilis, durasi, dan *genre*.

```
lagu.add("Radiohead", "Creep", "Alternative Rock", 1992, 3, 56);
lagu.add("Metallica", "Enter Sandman", "Metal", 1991, 5, 32);
lagu.add("Linkin Park", "In the End", "Nu Metal", 2001, 3, 36);
lagu.add("Oasis", "Wonderwall", "Britpop", 1995, 4, 18);
lagu.add("Pink Floyd", "Wish You Were Here", "Progressive Rock", 1975,
5, 17); lagu.add("The Cranberries", "Zombie", "Alternative Rock", 1994,
5, 6);lagu.add("Eminem", "Lose Yourself", "Hip Hop", 2002, 5, 26);
lagu.add("Beyoncé", "Halo", "Pop", 2008, 3, 45);
lagu.add("Lady Gaga", "Bad Romance", "Pop", 2009, 4, 54);
lagu.add("Bruno Mars", "Just the Way You Are", "Pop", 2010, 3, 41);
lagu.add("Justin Bieber", "Sorry", "Pop", 2015, 3, 20);
lagu.add("Shawn Mendes", "Stitches", "Pop", 2015, 3, 59);
lagu.add("Ed Sheeran", "Shape of You", "Pop", 2017, 3, 53);
lagu.add("Taylor Swift", "Love Story", "Country Pop", 2008, 3, 55);
lagu.add("Drake", "Hotline Bling", "Hip Hop", 2015, 4, 27);
lagu.add("Ariana Grande", "7 rings", "Pop", 2019, 2, 59);
lagu.add("Billie Eilish", "bad guy", "Alternative Pop", 2019, 3, 14);
lagu.add("Post Malone", "Circles", "Hip Hop", 2019, 3, 35);
lagu.add("Imagine Dragons", "Radioactive", "Alternative Rock", 2012, 3,
6);lagu.add("Harry Styles", "Watermelon Sugar", "Pop", 2019, 2, 54);
lagu.add("Doja Cat", "Say So", "Pop", 2019, 3, 58);
lagu.add("Lil Nas X", "Old Town Road", "Country Rap", 2018, 2, 38);
lagu.add("Lorde", "Royals", "Alternative Pop", 2013, 3, 10);
lagu.add("Kanye West", "Stronger", "Hip Hop", 2007, 5, 11);
lagu.add("Miley Cyrus", "Wrecking Ball", "Pop", 2013, 3, 43);
lagu.add("Selena Gomez", "Lose You to Love Me", "Pop", 2019, 3, 27);
lagu.add("Camila Cabello", "Havana", "Pop", 2017, 3, 36);
lagu.add("Zedd", "Clarity", "EDM", 2012, 4, 31);
lagu.add("David Guetta", "Titanium", "EDM", 2011, 4, 5);
lagu.add("Daft Punk", "Get Lucky", "Funk", 2013, 6, 7);
lagu.add("Sia", "Chandelier", "Pop", 2014, 3, 36);
lagu.add("Katy Perry", "Firework", "Pop", 2010, 3, 47);
lagu.add("Rihanna", "Diamonds", "Pop", 2012, 3, 45);
lagu.add("Kendrick Lamar", "HUMBLE.", "Hip Hop", 2017, 2, 57);
lagu.add("Jay-Z", "Empire State of Mind", "Hip Hop", 2009, 4, 36);
lagu.display();
System.out.println();
```

Script di atas adalah *method* "`add()`" yang berfungsi untuk menambahkan 35 lagu lagi ke dalam daftar lagu yang menjadikan terdapat 50 lagu di dalam *list*. Setiap kali *method* ini dipanggil, pemanggil menyertakan informasi tentang lagu yang terdiri dari nama artis, judul lagu, *genre*, tahun rilis, dan durasi (dalam menit dan detik). Data lagu yang diberikan akan disimpan dalam *node* baru yang kemudian ditambahkan ke dalam *linked list*. Dan dapat ditampilkan dengan menggunakan *method* "`display()`".

```
System.err.println("Setelah diututkan berdasarkan durasi (Bubble
Sort):");
lagu.bubbleSort();
lagu.display();
System.out.println();
```

Script di atas berfungsi untuk mengurutkan daftar lagu berdasarkan durasi menggunakan algoritma *bubble sort*. Setiap kali "`bubbleSort()`" dipanggil, proses pengurutan dilakukan dengan membandingkan durasi lagu (menit dan detik) dan menukar posisi lagu yang durasinya lebih besar dengan lagu yang durasinya lebih kecil. Dan dapat ditampilkan dengan menggunakan *method* "`display()`".

```
System.out.println("Setelah diurutkan berdasarkan tahun rilis
(Selection Sort):");
lagu.selectionSort();
lagu.display();
System.out.println();
```

Script di atas berfungsi untuk mengurutkan daftar lagu berdasarkan durasi menggunakan algoritma *selection sort*. Setiap kali "`selectionSort()`" dipanggil, proses pengurutan dilakukan dengan mencari lagu dengan tahun rilis terbesar di bagian yang belum terurut dan menukarnya dengan lagu yang ada di posisi saat ini. Proses ini diulang hingga seluruh daftar lagu terurut berdasarkan tahun rilis, dari yang terbaru ke yang terlama. Dan dapat ditampilkan dengan menggunakan *method* "`display()`".

```
lagu.add("Coldplay", "Viva La Vida", "Alternative Rock", 2008, 4, 2);
lagu.add("Linkin Park", "Numb", "Nu Metal", 2003, 3, 05);
lagu.add("Adele", "Rolling in the Deep", "Soul", 2010, 3, 48);
lagu.add("Kendrick Lamar", "Alright", "Hip Hop", 2015, 3, 39);
lagu.add("Lady Gaga", "Poker Face", "Pop", 2008, 3, 57);
lagu.add("Rihanna", "We Found Love", "Dance Pop", 2011, 3, 35);
lagu.add("Sia", "Chandelier", "Pop", 2014, 3, 36);
lagu.add("Ed Sheeran", "Castle on the Hill", "Pop", 2017, 4, 21);
lagu.add("Billie Eilish", "Bad Guy", "Pop", 2019, 3, 14);
lagu.add("Ariana Grande", "No Tears Left To Cry", "Pop", 2018, 3, 25);
```

Script di atas adalah *method* "`add()`" yang berfungsi untuk menambahkan 10 lagu lagi secara asc ke dalam daftar lagu yang menjadikan terdapat 60 lagu di dalam *list*. Setiap kali *method* ini dipanggil, pemanggil menyertakan informasi tentang lagu yang terdiri dari nama artis, judul lagu, *genre*, tahun rilis, dan durasi (dalam menit dan detik). Data lagu yang diberikan akan disimpan dalam *node* baru yang kemudian ditambahkan ke dalam *linked list*.

```
lagu.addfirst("Glenn Fredly", "Terserah", "R&B", 2007, 4, 5);
lagu.addfirst("Noah", "Separuh Aku", "Rock", 2012, 4, 30);
lagu.addfirst("Ariel Noah", "Mungkin Nanti", "Pop Rock", 2007, 5, 20);
lagu.addfirst("Kahitna", "Cantik", "Pop", 1994, 4, 10);
lagu.addfirst("Dewa 19", "Kangen", "Rock", 1992, 5, 45);
lagu.addfirst("Raisa", "Terjebak Nostalgia", "Pop", 2017, 4, 35);
lagu.addfirst("Melly Goeslaw", "Gantung", "Pop", 2003, 4, 10);
lagu.addfirst("Cakra Khan", "Kekasih Bayangan", "Pop", 2012, 3, 59);
lagu.addfirst("Padi", "Kasih Tak Sampai", "Rock", 2001, 5, 40);
lagu.addfirst("BCL", "Melukis Hari", "Pop", 2017, 4, 30);
```

Script di atas adalah *method* "`addFirst()`" yang berfungsi untuk menambahkan 10 lagu lagi secara dsc ke dalam daftar lagu yang menjadikan terdapat 70 lagu di dalam *list*. Setiap kali *method* ini dipanggil, pemanggil menyertakan informasi tentang lagu yang terdiri dari

nama artis, judul lagu, *genre*, tahun rilis, dan durasi (dalam menit dan detik). Data lagu yang diberikan akan disimpan dalam *node* baru yang kemudian ditambahkan ke dalam *linked list*.

```
lagu.add("Eagles", "Hotel California", "Rock", 1976, 6, 30);
lagu.add("Whitney Houston", "I Will Always Love You", "Pop", 1992, 4,
31);
lagu.add("BTS", "Dynamite", "K-pop", 2020, 3, 19);
lagu.add("Journey", "Don't Stop Believin'", "Rock", 1981, 4, 10);
lagu.add("Taylor Swift", "Blank Space", "Pop", 2014, 3, 51);
lagu.add("The Rolling Stones", "Paint It Black", "Rock", 1966, 3, 45);
lagu.add("Shakira", "Hips Don't Lie", "Pop", 2006, 3, 38);
lagu.add("U2", "With or Without You", "Rock", 1987, 4, 55);
lagu.add("Luis Fonsi ft. Daddy Yankee", "Despacito", "Reggaeton", 2017,
3, 48);
lagu.add("Rihanna", "Umbrella", "Pop", 2007, 4, 35);
```

Script di atas adalah *method* "`add()`" yang berfungsi untuk menambahkan 10 lagu lagi secara *random* ke dalam daftar lagu yang menjadikan terdapat 800 lagu di dalam *list*. Setiap kali *method* ini dipanggil, pemanggil menyertakan informasi tentang lagu yang terdiri dari nama artis, judul lagu, *genre*, tahun rilis, dan durasi (dalam menit dan detik). Data lagu yang diberikan akan disimpan dalam *node* baru yang kemudian ditambahkan ke dalam *linked list*.

```
System.out.println("\nMengurutkan berdasarkan durasi (Bubble Sort):");
long startTimeBubbleSort = System.nanoTime();
lagu.bubbleSort();
long endTimeBubbleSort = System.nanoTime();
lagu.display();
System.out.println("Waktu eksekusi Bubble Sort: " + (endTimeBubbleSort
- startTimeBubbleSort) + " nanoseconds");
```

Script di atas berfungsi untuk mengurutkan daftar lagu berdasarkan durasi menggunakan algoritma *bubble sort*. Setiap kali "`bubbleSort()`" dipanggil, proses pengurutan dilakukan dengan membandingkan durasi lagu (menit dan detik) dan menukar posisi lagu yang durasinya lebih besar dengan lagu yang durasinya lebih kecil. Dan dapat ditampilkan dengan menggunakan *method* "`display()`". Selain itu, waktu eksekusi dari proses pengurutan dihitung menggunakan "`System.nanoTime()`"dan ditampilkan dalam satuan nanodetik, untuk menunjukkan seberapa cepat algoritma *bubble sort* dalam mengurutkan daftar lagu.

```
System.out.println("\nMengurutkan berdasarkan tahun (Selection
Sort):");
long startTimeSelectionSort = System.nanoTime();
lagu.selectionSort();
long endTimeSelectionSort = System.nanoTime();
lagu.display();
System.out.println("Waktu eksekusi Selection Sort: " +
(endTimeSelectionSort - startTimeSelectionSort) + " nanoseconds");
```

Script di atas berfungsi untuk mengurutkan daftar lagu berdasarkan durasi menggunakan algoritma *selection sort*. Setiap kali "`selectionSort()`" dipanggil, proses pengurutan dilakukan dengan mencari lagu dengan tahun rilis terbesar di bagian yang belum

terurut dan menukarnya dengan lagu yang ada di posisi saat ini. Proses ini diulang hingga seluruh daftar lagu terurut berdasarkan tahun rilis, dari yang terbaru ke yang terlama. Dan dapat ditampilkan dengan menggunakan *method* "`display()`". Selain itu, waktu eksekusi dari proses pengurutan dihitung menggunakan "`System.nanoTime()`"dan ditampilkan dalam satuan nanodetik, untuk menunjukkan seberapa cepat algoritma *selection sort* dalam mengurutkan daftar lagu.

3.4 Hasil Analisi *Readme*

1. Penjelasan permasalahan nomor 2

```
================================================================================
             78 | Dewa 19 - Kangen (1992)
              5:45                      Rock
================================================================================
             79 | Daft Punk - Get Lucky (2013)
              6:07                      Funk
================================================================================
             80 | Eagles - Hotel California (1976)
              6:30                      Rock
================================================================================
Waktu eksekusi Bubble Sort: 223799 nanoseconds
```

Gambar 3.1 Hasil dari *Bubble Sort* Permasalahan NO 2 *Readme*

```
================================================================================
             79 | Billy Joel - Piano Man (1973)
              5:39                      Folk
================================================================================
              5:39                      Folk
================================================================================
             80 | The Rolling Stones - Paint It Black (1966)
              3:45                      Rock
              3:45                      Rock
================================================================================
Waktu eksekusi Selection Sort: 74901 nanoseconds
```

Gambar 3.2 Hasil dari *Selection Sort* Permasalahan NO 2 *Readme*

Gambar 3.1 merupakan hasil dari pengurutan data menggunakan *method bubble sort* dengan data lagu sebanyak 80 lagu, menghasilkan waktu eksekusinya yaitu, 223799 *nanoseconds*. Sedangkan **Gambar 3.2** merupakan hasil dari pengurutan data menggunakan *method selection sort* dengan data lagu sebanyak 80 lagu, menghasilkan waktu eksekusinya yaitu, 74901 *nanoseconds*. Dari hasil kedua perbandingan tersebut dapat dilihat bahwa *method selection sort* lebih cepat dalam melakukan pengurutan data dibandingkan dengan *method bubble sort*, dimana selisih waktu kedua *method* tersebut adalah 148898 *nanoseconds*. Hal ini terjadi karena *selection sort* melakukan pertukaran data yang lebih sedikit dibandingkan dengan *bubble sort*, meskipun keduanya memiliki kompleksitas waktu yang sama yaitu $O(n^2)$.

2. Penjelasan permasalahan nomor 3

   Bubble Sort melakukan pengurutan dengan cara membandingkan dua elemen data yang berdekatan secara berulang-ulang sampai seluruh data terurut. Pada setiap iterasi, elemen terbesar akan mengapung ke posisi akhir. Proses ini membutuhkan banyak pertukaran data karena setiap perbandingan yang tidak sesuai urutan akan menghasilkan pertukaran. Dalam kasus terburuk, *bubble sort* harus melakukan perbandingan dan pertukaran sebanyak n × (n-1) kali, dimana n adalah jumlah data. Hal inilah yang menyebabkan waktu eksekusi *bubble sort* menjadi lebih lama, seperti yang terlihat pada hasil pengujian yaitu 223799 *nanoseconds*. Sedangkan *Selection Sort* bekerja dengan cara mencari elemen minimum dalam data yang belum terurut, kemudian menukarnya dengan elemen pertama dari bagian yang belum terurut. Meskipun *selection sort* tetap harus melakukan perbandingan sebanyak n × (n-1)/2 kali, namun jumlah pertukaran datanya jauh lebih sedikit yaitu maksimal n-1 kali. Strategi ini membuat *selection sort* lebih efisien dalam hal penggunaan memori dan waktu eksekusi, terbukti dari hasil pengujian yang hanya membutuhkan waktu 74901 *nanoseconds*. Dengan demikian, pemilihan algoritma sorting yang tepat sangat mempengaruhi kinerja program, terutama dalam hal waktu eksekusi dan penggunaan memori. Meskipun kedua algoritma memiliki kompleksitas waktu $O(n^2)$, namun perbedaan jumlah operasi pertukaran data membuat *selection sort* lebih efektif untuk kasus pengurutan data yang relatif kecil seperti dalam pengujian ini.