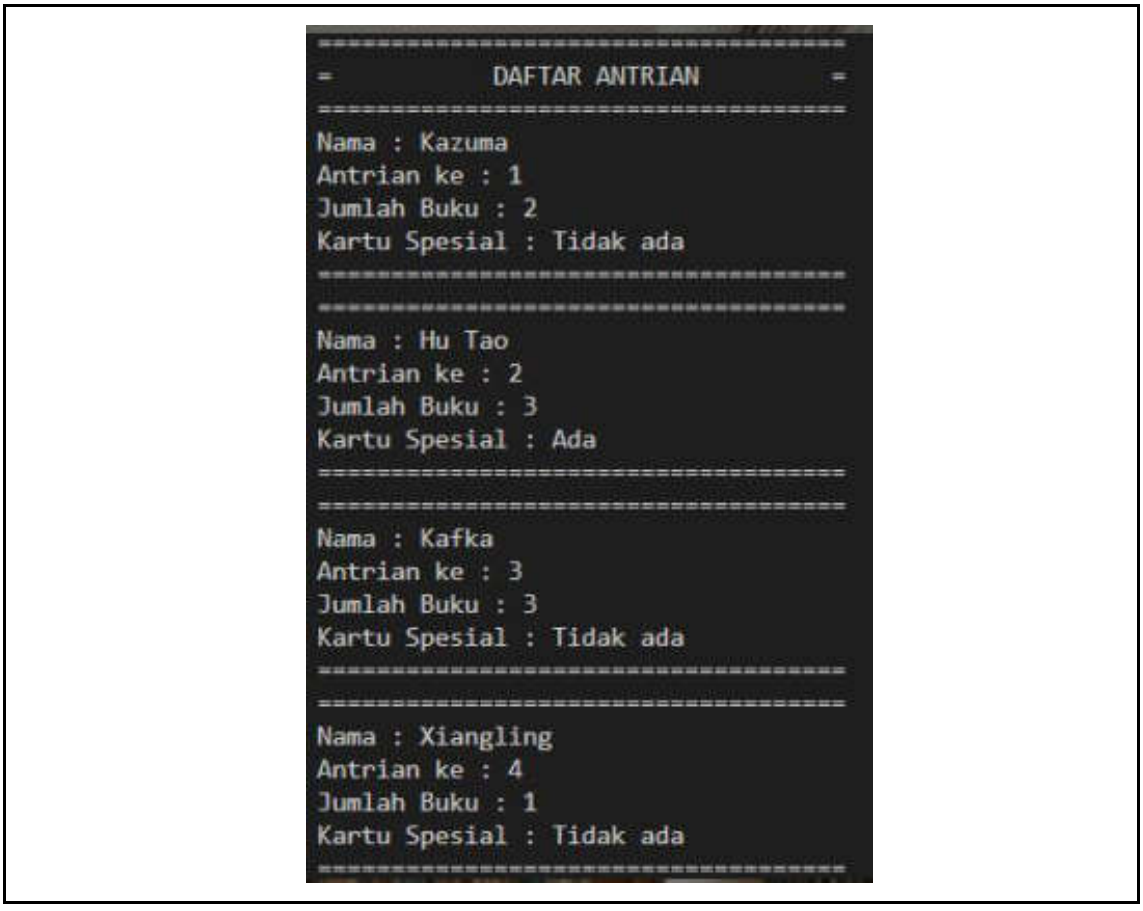
MODUL II
PEMROGRAMAN DASAR

2.1 PERMASALAHAN

2.1.1 Antrian Pemusing Kepala

Dihari berikutnya Yanto kembali menjadi volunteer penjaga perpustakaan sedirian dikarenakan yanti tidak dapat hadir, tidak lama setelah Yanto memulai shift-nya beberapa orang mulai muncul untuk meminjam buku. Sesudah orang orang itu memilih buku yang ingin mereka pinjam, merekapun berbaris di depan meja Yanto dengan barisan seperti gambar disamping.



Kazuma maju dan menunjukkan buku buku yang ingin dia pinjam ke pada Yanto, Kazuma meinjam 2 buku yang berjudul "Belajar Java" dan "Cara Menjadi Orang Kaya", Atribut lengkap bukunnya dapat dilihat pada gambar disamping. Setelah Kazuma menyelesaikan peminjamannya, ia pun pergi. Sekarang tinggal tersisa 3 orang dalam antrian, dan antrian setelah Kazuma menjadi antrian pertama.

```
==================================
=            BUKU KAZUMA         =
==================================
Judul Buku : Cara Menjadi Orang Kaya
Pengarang : Teguh
Genre : Fantasi
Status Buku : Buku Biasa
==================================
==================================
Judul Buku : Belajar Java
Pengarang : Raysen
Genre : Edukasi
Status Buku : Buku Biasa
```

```
==================================
=           DAFTAR ANTRIAN       =
==================================
Nama : Hu Tao
Antrian ke : 1
Jumlah Buku : 3
Kartu Spesial : Ada
==================================
==================================
Nama : Kafka
Antrian ke : 2
Jumlah Buku : 3
Kartu Spesial : Tidak ada
==================================
==================================
Nama : Xiangling
Antrian ke : 3
Jumlah Buku : 1
Kartu Spesial : Tidak ada
==================================
```

Sekarang giliran Hu Tao untuk menunjukkan buku yang ingin dia pinjam kepada Yanto, Hu Tao meminjam 3 buku yang berjudul "Cara Tidur Cepat", "Belajar C++" dan "Belajar Ilmu Hitam", Atribut lengkap bukunnya dapat dilihat pada gambar disamping.

```
==================================
=            BUKU HU TAO         =
==================================
Judul Buku : Belajar Ilmu Hitam
Pengarang : Megumin
Genre : Unknown
Status Buku : Cursed
==================================
==================================
Judul Buku : Belajar C++
Pengarang : Raysen
Genre : Edukasi
Status Buku : Buku Biasa
==================================
==================================
Judul Buku : Cara Tidur Cepat
Pengarang : Teguh
Genre : Edukasi Kayaknya
Status Buku : Cursed
==================================
```

Di perpustkaan ini terdapat dua jenis buku, buku pertama adalah buku biasa dan buku kedua adalah buku terkutuk. Buku terkutuk adalah buku yang memiliki informasi tentang pengetahuanpengetahuan terlarang jadi untuk meminjam buku terkutuk, peminjam harus memiliki kartu spesial. Karena Hu Tao memiliki kartu spesial, dia dapat meminjam buku terkutuk dan setelah menyelesaikan peminjamannya, Hu Tao langsung pergi. Di antrian sekarang tertinggal 2 orang.

```
==================================
=           DAFTAR ANTRIAN       =
==================================
Nama : Kafka
Antrian ke : 1
Jumlah Buku : 3
Kartu Spesial : Tidak ada
==================================
==================================
Nama : Xiangling
Antrian ke : 2
Jumlah Buku : 1
Kartu Spesial : Tidak ada
==================================
```

Setelah Hu Tao keluar, sesorang datang dengan terburu buru masuk ke dalam antrian. Orang itu adalah Sucrose, ia tampaknya ingin meminjam buku juga. di tangannya terdapat 3 tumpukkan buku.

Xiangling tiba tiba mendapat panggilan dari Zhongli, Yanto bisa mendengar samar samar percakapan xiangling di telefonnya, Yanto mendengar bahwa Gouba sedang mengamuk di Liyue dan butuh bantuan xiangling secepat mungkin. Karena urusan mendesak itu xiangling menyimpan Kembali bukunya dan langsung pergi. Tinggal tersisa dua orang di antrian.

Sucrose juga nampaknya sedang terburu buru, ia pun menanyakan kepada kafka apakah dia boleh bertukar tempat dengan kafka. Kafka setuju, Sucrose dan Kafka pun bertukar tempat.

Sucrose menunjukkan buku-buku yang ingin ia pinjam ke Yanto, Sucrose meminjam 3 buku dengan judul "Resurection", "Alhcemy" dan "Durin The Forgotten Dragon", Atribut lengkap bukunya dapat dilihat pada gambar disamping.

Setelah Sucrose menyelesaikan peminjamannya, ia pun pergi dengan terburu-buru. Sekarang tinggal tersisa 1 orang dalam antrian yaitu Kafka.

Yanto memeriksa satu per-satu buku yang ingin di pinjam kafka, dan salah satu buku yang ingin kafka pinjam adalah buku terkutuk. Yanto menanyakan kepada Kafka apakah dia memiliki kartu special, dan kafka menjawab bahwa dia tidak memiliki kartu special. Yanto pun memberi tahu Kafka bahwa Kafka memerlukan kartu special untuk meminjam buku terkutuk. Yanto pun memindahkan buku yang terkutuk ke bagian paling atas tumpukkan dan meminta kafka untuk mengembalikkan buku tersebut. Setelah kafka mengembalikkan buku yang terkutuk, ia menyelesaikan proses peminjaman. Sudah tidak ada orang dalam antrian.


BUKU KAFKA
Judul Buku : Misteri Menghilangnya Nasi Puyung
Pengarang : Optimus
Genre : Misteri
Status Buku : Buku Biasa
Judul Buku : Cara Menjadi Milioner Dalam 1 Jam
Pengarang : Master Oogway
Genre : Edukasi
Status Buku : Buku Biasa


DAFTAR ANTRIAN
Antrian Kosong

2.2 HASIL PERCOBAAN

2.2.1 Antrian Pemusing Kepala

1. Algoritma
    a. Kelas *Node*
        i. *Start.*
        ii. Mendeklarasikan *Node next*.
        iii. Tambahkan fungsi void untu getNext dan SetNext..
        iv. *Stop*
    b. Kelas *Linked List*
        i. *Start.*
        ii. Mendeklarasi atribut *front, rear* dan *top* sebagai objek *Node*.
        iii. Memasukkan elemen queue dengan data void enqueue untuk menumpuk antrian.
        iv. Tambahkan void dequeue untuk mengeluarkan queue.
        v. Memasukkan elemen stack dengan data void push pada stack untuk buku.
        vi. Deklarasikan juga pop untuk mengeluarkan stack.
        vii. Tambahkan delete untuk menghapus dan swap unutk menukar.
        viii. Tambahkan data display untuk menampilkan stack dan queue.
    c. Kelas Antrian
        i. Tambahkan string untuk nama, dan jenis kartu.
        ii. Tambahkan int untuk jumlah.
        iii. Tambahkan metode untuk menambahkan data antrian pada queue.
    d. Kelas Buku
        i. Tambahkan String untuk judul, pengarang, genre, dan kartu khusus.
        ii. Tambahkan metode untuk menambahkan data buku pada stack.
    e. Kelas *Main*
        i. Buat objek *LinkedList* bernama *modul2*.
        ii. Tambahkan *Node* baru dari data “Antrian”.
        iii. Tambahkan *Node* baru dari data “Buku”.
        iv. Tampilkan Node pada semua stack and queue.
        v. Tampilkan hasil delete dan swap pada stack dan queue.

2. *Source Code*

a. Node.java

```
package PACKAGE_NAME;
abstract class Node {
    private Node next;
    pubic Node getNext() {
    return next;
    }
    public void setNext(Node next) {
        this.next = next;
    }
    public String getData() {
        return "";
    }
}
```

b. LinkedList.java

```
package PACKAGE_NAME;

class LinkedList {
       private Node front;
       private Node rear;
       private Node top;

       public void enqueue(Antrian antrian) {
       if (rear == null) {
            front = rear = antrian;
       } else {
            rear.setNext(antrian);
            rear = antrian;
       }
       }

       public Antrian dequeue() {
       if (front == null) {
            return null;
       }
       Antrian dequeueAntrian = (Antrian) front;
       front = front.getNext();

       if (front == null) {
            rear = null;
       }
       return dequeueAntrian;
       }

       public void push(Buku buku) {
       buku.setNext(top);
       top = buku;
       }

       public Buku pop() {
       if (top == null) {
            return null;
       }
       Buku popBuku = (Buku) top;
       top = top.getNext();
       return popBuku;
       }
```

```
        public void displayQueue() {
System.out.println("====================================\n" +
                   "=          DAFTAR  ANTRIAN         =");
        if (front == null) {
             System.out.println("Antrian Kosong");
             return;
        }

        Node current = front;
        int posisi = 1;
        while (current != null) {
System.out.println("====================================");
             Antrian antrian = (Antrian) current;
             antrian.setPosisi(posisi);
             System.out.println(antrian.getData());
System.out.println("====================================");

             posisi++;
             current = current.getNext();
        }
        }

        public void displayStack() {
        Node temp = front;
        Antrian antrian = (Antrian) temp;
System.out.println("====================================\n" +
                   "=                BUKU  "  +  antrian.getNama()  +  "
                            =");

        if (top == null) {
             return;
        }

        Node current = top;
        while (current != null) {
System.out.println("====================================");
             Buku buku = (Buku) current;
             System.out.println(buku.getData());
System.out.println("====================================");
             current = current.getNext();
        }
        }

        public void swapQueue(int pos1, int pos2) {
        if (pos1 == pos2 || front == null) return;

        Node prev1 = null, prev2 = null;
        Node node1 = front, node2 = front;
        int currentPos = 1;

        while (node1 != null && currentPos != pos1) {
             prev1 = node1;
             node1 = node1.getNext();
             currentPos++;
        }

        currentPos = 1;
        while (node2 != null && currentPos != pos2) {
             prev2 = node2;
             node2 = node2.getNext();
```

```
        currentPos++;
    }

    if (node1 == null || node2 == null) return;

    if (prev1 != null) prev1.setNext(node2);
    else front = node2;

    if (prev2 != null) prev2.setNext(node1);
    else front = node1;

    Node temp = node1.getNext();
    node1.setNext(node2.getNext());
    node2.setNext(temp);
    }

    public void deleteQueue(int pos) {
    if (front == null) return;

    if (pos == 1) {
        front = front.getNext();
        if (front == null) rear = null;
        return;
    }

    Node current = front;
    Node prev = null;
    int currentPos = 1;

    while (current != null && currentPos != pos) {
        prev = current;
        current = current.getNext();
        currentPos++;
    }

    if (current == null) return;

    prev.setNext(current.getNext());
    if (current == rear) rear = prev;
    }

    public void swapStack(int pos1, int pos2) {
    if (pos1 == pos2 || top == null) return;

    Node prev1 = null, prev2 = null;
    Node node1 = top, node2 = top;
    int currentPos = 1;

    while (node1 != null && currentPos != pos1) {
        prev1 = node1;
        node1 = node1.getNext();
        currentPos++;
    }

    currentPos = 1;
    while (node2 != null && currentPos != pos2) {
        prev2 = node2;
        node2 = node2.getNext();
        currentPos++;
    }
```

```
        if (node1 == null || node2 == null) return;

        if (prev1 != null) prev1.setNext(node2);
        else top = node2;

        if (prev2 != null) prev2.setNext(node1);
        else top = node1;

        Node temp = node1.getNext();
        node1.setNext(node2.getNext());
        node2.setNext(temp);
        }
}
```

c. Buku.java

```
package PACKAGE_NAME;

class Buku extends Node {
    private String buku = "";
    private String author = "";
    private String genre = "";
    private String status = "";

    public Buku(String buku, String author, String genre, String
          status) {
        this.buku = buku;
        this.author = author;
        this.genre = genre;
        this.status = status;
    }

    // Getters and Setters
    public void setBuku(String buku) {
        this.buku = buku;
    }

    public String getBuku() {
        return this.buku;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getAuthor() {
        return this.author;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }

    public String getGenre() {
        return this.genre;
    }

    public void setStatus(String status) {
        this.status = status;
    }
```

```
    public String getStatus() {
        return this.status;
    }

    // Override getData to display all information about the book
    @Override
    public String getData() {
        return "Judul Buku: " + this.buku + "\n" +
                "Pengarang: " + this.author + "\n" +
                "Genre: " + this.genre + "\n" +
                "Status Buku: " + this.status;
    }
}
```

d. Antrian.java

```
package PACKAGE_NAME;

class Antrian extends Node {
    private String nama = "";
    private int jumlah = 0;
    private String kartu = "";
    private int posisi = -1;

    public Antrian(String nama, int jumlah, String kartu) {
        this.nama = nama;
        this.jumlah = jumlah;
        this.kartu = kartu;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNama() {
        return this.nama;
    }

    public void setJumlah(int jumlah) {
        this.jumlah = jumlah;
    }

    public int getJumlah() {
        return this.jumlah;
    }

    public void setKartu(String kartu) {
        this.kartu = kartu;
    }

    public String getKartu() {
        return this.kartu;
    }

    public void setPosisi(int posisi) {
        this.posisi = posisi;
    }

    public int getPosisi() {
        return this.posisi;
    }
```

```
    @Override
    public String getData() {
        return "Nama: " + this.nama + "\n" +
                "Antrian ke: " + this.posisi + "\n" +
                "Jumlah Buku: " + this.jumlah + "\n" +
                "Kartu Spesial: " + this.kartu;
    }
}
```

e. Main.java

```
package PACKAGE_NAME;

public class Main {
    public static void main(String[] args) {
        LinkedList modul2 = new LinkedList();

        modul2.enqueue(new Antrian("Kazuma", 2, "ada"));
        modul2.enqueue(new Antrian("Hu Tao", 3, "ada"));
        modul2.enqueue(new Antrian("kafka", 3, "Tida Ada"));
        modul2.enqueue(new Antrian("Xiangling", 1, "Tidak Ada"));
        modul2.push(new Buku("Belajar Java", "Raysen", "Edukasi",
         "Buku Biasa"));
        modul2.push(new Buku("Cara Menjadi Orang Kaya", "Teguh",
         "Fantasy", "Buku Biasa"));
        modul2.displayQueue();
        modul2.displayStack();

        modul2.dequeue();
        modul2.pop();
        modul2.pop();

        modul2.push(new Buku("Cara Tidur Cepat", "Teguh", "Edukasi
         Kayaknya", "Cursed"));
        modul2.push(new Buku("Belajar C++", "Raysen", "Edukasi",
         "Buku Biasa"));
        modul2.push(new Buku("Belajar Ilmu Hitam", "Megumin",
         "Unknown", "Cursed"));
        modul2.displayStack();
        modul2.dequeue();
        modul2.pop();
        modul2.pop();
        modul2.pop();

        modul2.displayQueue();
        modul2.enqueue(new Antrian("Sucrose", 3, "ada"));
        modul2.displayQueue();

        modul2.deleteQueue(2);
        modul2.displayQueue();

        modul2.swapQueue(1,2);
        modul2.displayQueue();

        modul2.push(new Buku("Resurection", "Unknown", "Unknown",
         "Cursed"));
        modul2.push(new Buku("Alchemy", "Albedo", "Sience",
         "Cursed"));
        modul2.push(new Buku("Durin the Forgotten Dragon", "Gold",
         "Misteri", "Buku Biasa"));
```

```
        modul2.displayStack();
        modul2.dequeue();
        modul2.pop();
        modul2.pop();
        modul2.pop();

        modul2.displayQueue();
        modul2.push(new Buku("Raysen the Forgotten One", "Unknown",
         "Sejarah", "Cursed"));
        modul2.push(new Buku("Misteri Menghilangnya Nasi Puyung",
         "Optimus", "Misteri", "Buku Biasa"));
        modul2.push(new Buku("Cara Menjadi milioner Dalam 1 Jam",
         "Master Oogway", "Edukasi", "Buku Biasa"));
        modul2.displayStack();
        modul2.swapStack(1,3);
        modul2.displayStack();
        modul2.pop();
        modul2.displayStack();
        modul2.pop();
        modul2.pop();
        modul2.displayQueue();
    }
}
```

2.3 ANALISIS DATA

1.3.1 Nama Program

```
class Antrian extends Node {
    private String nama = "";
    private int jumlah = 0;
    private String kartu = "";
    private int posisi = -1;
```

Script "`public class Character {`" adalah sebuah model node yang digunakan untuk merepresentasikan karakter dalam sebuah struktur linked list, di mana setiap node menyimpan informasi tentang karakter dan stand-nya.

```
public Antrian(String nama, int jumlah, String kartu) {
        this.nama = nama;
        this.jumlah = jumlah;
        this.kartu = kartu;
    }
```

Script "`public Character(String characterName, String standName, String standAbilities, String standPower, String standSpeed, String standRange, double standRating) {`" adalah Konstruktor yang digunakan untuk menginisialisasi objek "`CharacterNode`" dengan nilai-nilai yang diberikan untuk setiap atributnya.

```
public class LinkedList {
    private Node head;
    private Node tail;
    public LinkedList() {
        this.head = null;
        this.tail = null;
    }
```

Script "`public class LinkedList {`" merupakan implementasi dari struktur data double linked list yang berfungsi untuk menyimpan daftar karakter, dengan fokus pada kategori atau jenis tertentu yang diidentifikasi oleh atribut "`LinkedList()`".

```
public void tambahCharacter(Character character) {
        Node newNode = new Node(character);
        if (head == null) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }}
```

Script "`public void tambahCharacter (....){`" adalah metode dalam kelas "`\ LinkedList`" bertugas untuk menambahkan karakter baru ke dalam linked list. Metode ini menerima beberapa parameter yang mendeskripsikan karakter, seperti "`characterName`", "`standName`", "`standability`" serta atribut karakter seperti "`power`", "`speed`", "`range`", dan "`totalRating`".

```
public void tampilanCharacters() {
        Node current = head;
        while (current != null) {
            System.out.println(current.data);
            current = current.next;
        }
        System.out.println();
    }}
```

Script "`public void tampilanCharacters () {`" dalam kelas "`LinkedList`" berfungsi untuk menampilkan informasi tentang semua karakter yang tersimpan dalam linked list. Saat metode ini dipanggil, pertama-tama akan dicetak nama kategori karakter berdasarkan atribut "`character`". Kemudian, sebuah variabel sementara "`current`" diinisialisasi untuk menunjuk ke "`head`" dari linked list.

```
public void merge(LinkedList tipeLain) {
        if (this.head == null) {
            this.head = tipeLain.head;
            this.tail = tipeLain.tail;
        } else if (tipeLain.head != null) {
            this.tail.next = tipeLain.head;
            tipeLain.head.prev = this.tail;
            this.tail = tipeLain.tail;
        }
    }
```

Script "`public void merge {`" dalam kelas "`LinkedList`" bertujuan untuk menggabungkan dua linked list menjadi satu, dengan menambahkan semua karakter dari "`tipeList`" ke akhir linked list saat ini. Jika linked list saat ini ("`this`") kosong (ditandai dengan "`head`" yang bernilai "`null`"), maka "`head`" dan "`tail`" dari linked list saat ini akan diatur untuk merujuk ke "`head`" dan "`tail`" dari "`tipeLain`".

```
public void addmerge(DoubleLinkedList otherList, String carii) {
        if (head == null) {
            head = otherList.head;
            tail = otherList.tail;
        } else {
            CharacterNode temp = head;
            while (temp != null) {
                if (temp.characterName.equals(carii)) {
                 CharacterNode nextNode = temp.next;
                temp.next = otherList.head;
                if (otherList.head != null) {
                    otherList.head.prev = temp;
                }
                if (nextNode != null) {
                    otherList.tail.next = nextNode;
                    nextNode.prev = otherList.tail;
                } else {
                    tail = otherList.tail;
                }
                return;
                }
                temp = temp.next;
```

```
            }
        }
    }
```

Script "`public void addmerge(DoubleLinkedList otherList, String carii) {`" dalam kelas "`DoubleLinkedList`" dirancang untuk menyisipkan karakter dari linked list lain (otherList) ke dalam linked list saat ini pada posisi setelah karakter yang namanya sesuai dengan parameter "`carii`". Jika linked list saat ini kosong (dengan "`head`" bernilai "`null`"), maka "`head`" dan "`tail`" diatur untuk merujuk ke "`head`" dan "`tail`" dari "`otherList`".

```
public void addBefore(String targetName, String characterName, String
standName, String standAbility, char power, char speed, char range,
double totalRating) {
         CharacterNode newCharacter = new CharacterNode(characterName,
standName, standAbility, power, speed, range, totalRating);
        CharacterNode temp = head;
        while (temp != null) {
            if (temp.characterName.equals(targetName)) {
                newCharacter.next = temp;newCharacter.prev = temp.prev;
                if (temp.prev != null) {
                    temp.prev.next = newCharacter;
                } else {head = newCharacter;
                }
                temp.prev = newCharacter; return;
            }temp = temp.next;
        }
    }
```

Script "`public void addBefore (....){`" dalam kelas "`DoubleLinkedList`" bertujuan untuk menyisipkan karakter baru sebelum karakter yang namanya sesuai dengan parameter "`targetName`". Pertama, metode ini membuat objek baru dari "`CharacterNode`" menggunakan informasi yang diberikan. Kemudian, dengan menggunakan *loop* "`while`", metode ini mencari node yang memiliki "`characterName`" yang sesuai dengan "`targetName`".

```
public void swapCharacters(String character1, String character2) {
        CharacterNode node1 = null, node2 = null, temp = head;
        while (temp != null) {
            if (temp.characterName.equals(character1)) {
                node1 = temp;
            } else if (temp.characterName.equals(character2)) {
                node2 = temp;
            }
            temp = temp.next;
        }

        if (node1 != null && node2 != null && node1 != node2) {
            // Swapping references
            CharacterNode node1sebelum = node1.prev;
            CharacterNode node1sesudah = node1.next;
            CharacterNode node2seebelum = node2.prev;
            CharacterNode node2sesudah = node2.next;
```

```
            // Adjust node1's connections
            if (node1sebelum != null) {
                node1sebelum.next = node2;
            } else {
                head = node2;
            }
            if (node1sesudah != null && node1sesudah != node2) {
                node1sesudah.prev = node2;
            }

            // Adjust node2's connections
            if (node2seebelum != null) {
                node2seebelum.next = node1;
            } else {
                head = node1;
            }
            if (node2sesudah != null && node2sesudah != node1) {
                node2sesudah.prev = node1;
            }

            // Swap prev and next pointers
            if (node1sesudah == node2) {
                node1.next = node2sesudah;
                node1.prev = node2;
                node2.next = node1;
                node2.prev = node1sebelum;
            } else if (node2sesudah == node1) {
                node2.next = node1sesudah;
                node2.prev = node1;
                node1.next = node2;
                node1.prev = node2seebelum;
            } else {
                node1.next = node2sesudah;
                node1.prev = node2seebelum;
                node2.next = node1sesudah;
                node2.prev = node1sebelum;
            }

            // Update head and tail if needed
            if (node1.prev == null) head = node1;
            if (node2.prev == null) head = node2;
            if (node1.next == null) tail = node1;
            if (node2.next == null) tail = node2;
        }
    }
}
```

Script "`public void swapCharacters(String character1, String character2) {`" dalam kelas "`DoubleLinkedList`" dirancang untuk menukar posisi dua karakter dalam linked list berdasarkan nama karakter yang diberikan sebagai parameter "`character1`" dan "`character2`". Pertama, metode ini mencari kedua node yang sesuai dengan nama karakter tersebut dalam linked list, dan jika keduanya ditemukan dan tidak sama, proses pertukaran dimulai.

```
public class Main {
    public static void main(String[] args) {
//task 1
```

```
System.out.println("TASK 1");
        DoubleLinkedList heroes = new DoubleLinkedList("Heros");
        heroes.addCharacter("Jotaro Kujo", "Star Platinum", "Time Stop",
'A', 'A', 'C', 9.0);
          heroes.addCharacter("Josuke  Higashikata",  "Crazy  Diomond",
"Restoration", 'A', 'A', 'D', 8.0);
         heroes.addCharacter("Shigechi", "Harvest", "Collecting", 'E',
'B', 'A', 6.5);
        heroes.addCharacter("Giorno Giovanna","Gold Experience" , "Life
Giver", 'C', 'A', 'C', 7.0);
          heroes.addCharacter("Jolyne  Cujoh",  "Stone  Free",  "String
Manipulation", 'B', 'A', 'B', 8.0);
        // Adding characters to each list
        DoubleLinkedList villains = new DoubleLinkedList("vilain");
        villains.addCharacter("Kenny G", "tenore sax", "Illusions", 'E',
'E', 'D', 2.5);
       villains.addCharacter("Dio", "The World", "Time Stop", 'A', 'A',
'B', 8.5);
        villains.addCharacter("Yoshikage Kira", "Killaer Queen", "Bomb
Transmutation", 'A', 'B', 'D', 7.0);
        villains.addCharacter("Diavolo Vinegar Doppio", "King crimson",
"Time Ereasure", 'A', 'A', 'E', 7.0);
               villains.addCharacter("Pucci",   "Whitesnake",   "Memory
Manipulation", 'B', 'C', 'B', 7.5);
         // villains.addCharacter("Emporio", "Weather Report", "Oxygen
Manipulation", 'B', 'B', 'C', 7.0);
        DoubleLinkedList animals = new DoubleLinkedList("Animal");
         animals.addCharacter("Iggy", "The Fool", "Sand Manipulation",
'B', 'C', 'B', 7.5);
        animals.addCharacter("Bug-Eten", "Ratt", "Melting Darts", 'C',
'C', 'A', 7.0);
         animals.addCharacter("Pet Shop", "Horus", "Ice Manipulation",
'A', 'B', 'C', 8.0);
        heroes.displayCharacters();villains.displayCharacters();
        animals.displayCharacters();
        // Deleting characters
        //task 2
        System.out.println("TASK 2");
        villains.deleteCharacter("Kenny G");
        animals.deleteCharacter("Pet Shop");
        heroes.deleteCharacter("Shigechi");
        heroes.displayCharacters();
        villains.displayCharacters();animals.displayCharacters();

         // Merging Villains and Animals lists, and appending them to
Heroes
        //task 3
        System.out.println("TASK 3");
        DoubleLinkedList merge = new DoubleLinkedList("Merge");
        villains.merge(animals); villains.displayCharacters();
        heroes.addmerge(villains, "Josuke Higashikata");
        merge.merge(heroes);
        merge.displayCharacters();
        //Task 4
        System.out.println("TASK 4");
         heroes.addBefore("Pucci", "Emporio", "Weather Report", "Oxygen
Manipulation", 'B', 'B', 'c', 7.0);
        merge.displayCharacters();
        villains.deleteCharacter("Pucci");merge.displayCharacters();

        // task 5
```

```
        System.out.println("TASK 5");
        merge.swapCharacters("Dio", "Iggy");
      merge.swapCharacters("Jotaro Kujo", "Jolyne Cujoh");
        merge.displayCharacters();
    }
}
```

Script "`public static void main(String[] args) {`" dalam Kelas Main ini berfungsi sebagai titik masuk untuk menjalankan berbagai operasi pada objek dari kelas DoubleLinkedList, yang menyimpan karakter dalam kategori "Heros", "villain", dan "Animal". Pada TASK 1, beberapa karakter ditambahkan ke masing-masing daftar menggunakan metode "`addCharacter`", dan kemudian semua karakter dari setiap daftar ditampilkan. Pada TASK 2, beberapa karakter dihapus dari daftar menggunakan metode "`deleteCharacter`", di mana setelah penghapusan, karakter yang tersisa ditampilkan lagi. TASK 3 menggabungkan daftar villain dan animals menggunakan metode "`merge`", lalu menambahkan hasil gabungan tersebut ke daftar heroes dengan "`addmerge`" dan menampilkan karakter yang ada di dalam daftar "`merge`". Di TASK 4, metode "`addBefore`" digunakan untuk menyisipkan karakter baru sebelum karakter yang ditentukan, diikuti dengan penghapusan karakter dari daftar villain dan menampilkan hasilnya. Terakhir, TASK 5 menggunakan metode "`swapCharacters`" untuk menukar posisi karakter tertentu dalam daftar, dan kemudian menampilkan daftar karakter yang diperbarui. Kelas ini mengilustrasikan berbagai manipulasi linked list yang memungkinkan untuk mengelola karakter dengan cara yang terstruktur dan dinamis.