MODUL 2

STACK AND QUEUE

1.1 PERMASALAHAN

1.3.1 Rumah Sakit Bahagia

Suatu hari, seorang programmer bernama Divo ditugaskan untuk mengembangkan sistem manajemen rumah sakit. Divo ingin membuat sebuah aplikasi yang dapat mengelola daftar pasien dan tim medis dengan efisien. Ia memutuskan untuk menggunakan struktur data Queue untuk mengatur antrian pasien yang masuk ke rumah sakit. Divo mulai dengan mendefinisikan kelas Pasien yang menyimpan informasi penting tentang setiap pasien, termasuk nama, umur, jenis kelamin, penanganan, dan status. Ia memastikan setiap objek Pasien dapat dengan mudah diubah menjadi format string untuk ditampilkan dalam daftar. Setelah itu, Divo membuat kelas Node dan SingleLinkedList untuk menangani daftar pasien yang akan dikelola. Kelas Node berfungsi sebagai elemen dasar dari daftar, sementara SingleLinkedList mengelola semua operasi seperti menambah, menghapus, dan menampilkan pasien. Divo melanjutkan dengan mendefinisikan kelas Queue yang menggunakan SingleLinkedList untuk menampung antrian pasien. Ia menambahkan metode untuk menambahkan pasien baru ke antrian (enqueue) dan mengambil pasien yang sedang dilayani (dequeue). Setiap kali pasien baru datang, Divo menambahkannya ke dalam antrian dengan mudah. Untuk menggambarkan alur kerja sistem, Divo mengisi antrian dengan beberapa pasien, termasuk Bruno Jigola, Mitrime Ganthi, dan Liam Wayne. Ia mencetak daftar pasien yang ada di dalam antrian dengan rapi dan teratur, menampilkan informasi penting seperti nama dan status setiap pasien. Setelah itu, Divo menyiapkan dua ruang operasi untuk menangani pasien darurat. Ia membuat antrian baru untuk pasien darurat dan memasukkan pasien seperti Megan Foz dan Poor Didi. Divo juga menampilkan detail tentang tim medis yang akan menangani operasi, termasuk nama dokter dan jenis penanganan yang akan diberikan. Divo memastikan sistem ini dapat menangani berbagai kategori pasien, baik yang darurat maupun yang reguler, dengan menampilkan semua informasi yang diperlukan secara jelas. Ia juga menambahkan metode untuk menampilkan daftar tim dokter yang bertanggung jawab atas penanganan pasien, termasuk detail mengenai dokter operasi, dokter anestesi, dan perawat yang terlibat.

1.2 HASIL PERCOBAAN

1.4.1 Program Rumah Sakit Bahagia

1. Algoritma
   a. Inisialisasi Daftar Pasien.
   b. Menambahkan Pasien.
   c. Menggabungkan Daftar Pasien.
   d. Menampilkan Daftar Pasien.
   e. Menangani Pasien.
   f. Menampilkan Daftar Tim Dokter
2. *Source Code*

```
package Medium;

public class TimDokter {
    String namaTim;
    String dokterOperasi;
    String dokterAnestesi;
    String perawat1;
    String perawat2;

    public TimDokter(String namaTim, String dokterOperasi, String
dokterAnestesi, String perawat1, String perawat2) {
        this.namaTim = namaTim;
        this.dokterOperasi = dokterOperasi;
        this.dokterAnestesi = dokterAnestesi;
        this.perawat1 = perawat1;
        this.perawat2 = perawat2;
    }

    @Override
    public String toString() {
        return "Nama Tim: " + namaTim + "\nDokter Operasi: " +
dokterOperasi +
               "\nDokter Anestesi: " + dokterAnestesi +
               "\nPerawat 1: " + perawat1 +
               "\nPerawat 2: " + perawat2 + "\n";
    }
}
package Medium;

public class Stack {
    private SingleLinkedList list;

    public Stack() {
        list = new SingleLinkedList();
    }

    public void push(Pasien pasien) {
        list.add(pasien);
    }

    public Pasien pop() {
        if (list.size() == 0) return null;
        Pasien data = (Pasien) list.getFirst();
```

```
        list.removeFirst();
        return data;
    }

    public void display() {
        list.display();
    }
}

package Medium;

public class SingleLinkedList {
    private Node head;

    public SingleLinkedList() {
        head = null;
    }

    public void add(Object data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    public void display() {
        Node current = head;
        while (current != null) {
             System.out.print(current.data + (current.next != null ?
"\n" : ""));
            current = current.next;
        }
    }

    public int size() {
        int size = 0;
        Node current = head;
        while (current != null) {
            size++;
            current = current.next;
        }
        return size;
    }

    public void removeFirst() {
        if (head != null) {
            head = head.next;
        }
    }

    public Object getFirst() {
        return head != null ? head.data : null;
    }
}
```

```
package Medium;
public class Queue {
    private SingleLinkedList list;

    public Queue() {
        list = new SingleLinkedList();
    }

    public void enqueue(Pasien pasien) {
        list.add(pasien);
    }

    public Pasien dequeue() {
        if (list.size() == 0) return null;
        Pasien data = (Pasien) list.getFirst();
        list.removeFirst();
        return data;
    }

    public void display() {
        list.display();
    }
}

package Medium;

public class Pasien {
    private String nama;
    private int umur;
    private String jenisKelamin;
    private String penanganan;
    private String status;

    public Pasien(String nama, int umur, String jenisKelamin, String
penanganan, String status) {
        this.nama = nama;
        this.umur = umur;
        this.jenisKelamin = jenisKelamin;
        this.penanganan = penanganan;
        this.status = status;
    }

    @Override
    public String toString() {
         return nama + ", " + umur + ", " + jenisKelamin + ", " +
penanganan + ", " + status;
    }
}

package Medium;

public class Node {
    Object data;
    Node next;

    public Node(Object data) {
        this.data = data;
        this.next = null;
    }
}
package Medium;
```

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

         antrian.enqueue(new Pasien("Bruno Jigola", 25, "Laki-Laki",
"Berat", "VIP"));
       antrian.enqueue(new Pasien("Mitrime Ganthi", 37, "Perempuan",
"Berat", "VIP"));
         antrian.enqueue(new Pasien("Liam Wayne", 31, "Laki-Laki",
"Sedang", "Reguler"));
         antrian.enqueue(new Pasien("Anceloti Ferguso", 41, "Laki-
Laki", "Ringan", "VIP"));
         antrian.enqueue(new Pasien("Ali Dakota", 20, "Laki-Laki",
"Ringan", "Reguler"));

         System.out.println("=====================================
=");
                  System.out.println("=                          Daftar
Pasien               =\n");
        antrian.display();
         System.out.println("=====================================
=");

        Queue antrianBaru = new Queue();
        antrianBaru.enqueue(new Pasien("Megan Foz", 32, "Perempuan",
"Darurat", "VIP"));
        antrianBaru.enqueue(new Pasien("Poor didi", 54, "Laki-Laki",
"Darurat", "VIP"));
        antrianBaru.enqueue(new Pasien("Anakin Gonzales", 29, "Laki-
Laki", "Darurat", "Reguler"));
          antrianBaru.enqueue(new Pasien("Fander Liot", 20, "Laki-
Laki", "Sedang", "VIP"));
             antrianBaru.enqueue(new  Pasien("Picanta  Ether",  29,
"Perempuan", "Sedang", "Reguler"));

         System.out.println("=====================================
=");
                System.out.println("=                          Daftar
Pasien              =\n");
        antrianBaru.display();
        antrianBaru.display();
        antrianBaru.enqueue(antrian.dequeue());
         System.out.println("=====================================
=");

         System.out.println("=====================================
==");
        System.out.println("Operasi Sedang Berlangsung");
         System.out.println("=====================================
==");
        System.out.println("\n=== Ruang Operasi 1 ===");
        System.out.println("Tim Penanganan: Tim Dr. Purwanto");
                System.out.println("Nama   Pasien        :   "   +
antrianBaru.dequeue().toString());
        System.out.println("Penanganan    : Darurat");

        System.out.println("\n=== Ruang Operasi 2 ===");
        System.out.println("Tim Penanganan: Tim Dr. Rizal Airin");
                System.out.println("Nama   Pasien        :   "   +
antrianBaru.dequeue().toString());
        System.out.println("Penanganan    : Darurat");
```

```
                System.out.println("\n============   Dalam   Antrian
==================");
        antrianBaru.display();

        Queue ruangPerawatan = new Queue();
          ruangPerawatan.enqueue(new  Pasien("Anceloti  Ferguso",  41,
"Laki-Laki", "Ringan", "VIP"));
            ruangPerawatan.enqueue(new  Pasien("Bana  Ilyana",  29,
"Perempuan", "Ringan", "VIP"));

         System.out.println("\n=====================================
=====");
        System.out.println("Dalam Penanganan Dokter Umum");
         System.out.println("=====================================
===");

        System.out.println("\n=== Ruang Perawatan 1 ===");
        System.out.println("Tim Penanganan: Dr. Erling Haaland");
                 System.out.println("Nama   Pasien        :   "   +
ruangPerawatan.dequeue().toString());
        System.out.println("Penanganan    : Ringan");

        System.out.println("\n=== Ruang Perawatan 2 ===");
        System.out.println("Tim Penanganan: Dr. Ridwan Munir");
                 System.out.println("Nama   Pasien        :   "   +
ruangPerawatan.dequeue().toString());
        System.out.println("Penanganan    : Ringan");

                          System.out.println("\n===============Dalam
Antrian====================");
        ruangPerawatan.display();

         System.out.println("\n=====================================
=====");
        System.out.println("Daftar Tim Dokter Umum");
         System.out.println("=====================================
===");
       System.out.println("1. Dr. Erling Haaland\n   Dokter Operasi:
Dr.  Erling  Haaland\n    Dokter  Anestesi:  \n    Perawat  1:  Oscar
Az\n   Perawat 2: -");
        System.out.println("\n2. Dr. Ridwan Munir\n   Dokter Operasi:
Dr.  Ridwan  Munir\n     Dokter  Anestesi:  \n     Perawat  1:  Nia
Valentino\n   Perawat 2: -");

         System.out.println("\n=====================================
=====");
        System.out.println("Daftar Tim Operasi");
         System.out.println("=====================================
===");
         System.out.println("1. Tim Dr. Purwanto\n   Dokter Operasi:
Dr. Purwanto\n   Dokter Anestesi: Dr. Rewind HD\n   Perawat 1: Dwi
Julian\n   Perawat 2: Exelsen");
          System.out.println("\n2.  Tim  Dr.  Rizal  Airin\n    Dokter
Operasi:   Dr.   Rizal   Airin\n      Dokter   Anestesi:   Dr.   Kyky
Insania\n   Perawat 1: Abyan Tubuh\n   Perawat 2: Rahmatdian");

         System.out.println("\n=====================================
=====");
        System.out.println("Operasi Sedang Berlangsung");
```

```
         System.out.println("=====================================
===");
                  System.out.println("\n==========Ruang    Operasi
1===============");
        System.out.println("Tim Penanganan: Tim Dr. Faruk Ad-Dhar");
        System.out.println("Nama Pasien: Anakin Gonzales");
        System.out.println("Penanganan: Darurat");
                 System.out.println("\n============Ruang    Operasi
2=====================");
           System.out.println("Tim  Penanganan:  Tim  Dr.  Anastasya
Maharani");
        System.out.println("Nama Pasien: Bruno Jigola");
        System.out.println("Penanganan: Berat");
                          System.out.println("\n============Dalam
Antrian======================");

        antrian.display();

         System.out.println("\n=================================
=====");
        System.out.println("Daftar Pasien Pasca Penanganan");
         System.out.println("=====================================
===");
       System.out.println("Megan Foz, 32, Perempuan, Darurat, VIP");
       System.out.println("Poor Didi, 54, Laki-Laki, Darurat, VIP");
        System.out.println("Anakin Gonzales, 29, Laki-Laki, Darurat,
Reguler");
      System.out.println("Bruno Jigola, 25, Laki-Laki, Berat, VIP");
         System.out.println("Mitrime  Ganthi,  37,  Perempuan,  Berat,
VIP");
        System.out.println("Antoni Dos Santos, 32, Laki-Laki, Berat,
Reguler");
        System.out.println("Anceloti Ferguso, 41, Laki-Laki, Ringan,
VIP");
       System.out.println("Bana Ilyana, 29, Perempuan, Ringan, VIP");
          System.out.println("Ali  Dakota,  20,  Laki-Laki,  Ringan,
Reguler");
         System.out.println("=====================================
===");
    }
}
```

1.3 ANALISIS DATA

1.5.1 Main & Node

```
package Medium;
public class Main {
    public static void main(String[] args) {
        Queue antrian = new Queue();

        antrian.enqueue(new Pasien("Bruno Jigola", 25, "Laki-Laki",
"Berat", "VIP"));
        antrian.enqueue(new Pasien("Mitrime Ganthi", 37, "Perempuan",
"Berat", "VIP"));
        antrian.enqueue(new Pasien("Liam Wayne", 31, "Laki-Laki",
"Sedang", "Reguler"));
        antrian.enqueue(new Pasien("Anceloti Ferguso", 41, "Laki-Laki",
"Ringan", "VIP"));
        antrian.enqueue(new Pasien("Ali Dakota", 20, "Laki-Laki",
"Ringan", "Reguler"));

        System.out.println("=======================================");
        System.out.println("=        Daftar Pasien              =\n");
        antrian.display();
        System.out.println("=======================================");

        Queue antrianBaru = new Queue();
        antrianBaru.enqueue(new Pasien("Megan Foz", 32, "Perempuan",
"Darurat", "VIP"));
        antrianBaru.enqueue(new Pasien("Poor didi", 54, "Laki-Laki",
"Darurat", "VIP"));
        antrianBaru.enqueue(new Pasien("Anakin Gonzales", 29, "Laki-
Laki", "Darurat", "Reguler"));
        antrianBaru.enqueue(new Pasien("Fander Liot", 20, "Laki-Laki",
"Sedang", "VIP"));
        antrianBaru.enqueue(new Pasien("Picanta Ether", 29, "Perempuan",
"Sedang", "Reguler"));

        System.out.println("=======================================");
        System.out.println("=          Daftar Pasien              =\n");
        antrianBaru.display();
        antrianBaru.display();
        antrianBaru.enqueue(antrian.dequeue());
```

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


System.out.println("=======================================");
        System.out.println("Operasi Sedang Berlangsung");

System.out.println("=======================================");
        System.out.println("\n=== Ruang Operasi 1 ===");
        System.out.println("Tim Penanganan: Tim Dr. Purwanto");
        System.out.println("Nama    Pasien                  :    "    +
antrianBaru.dequeue().toString());
        System.out.println("Penanganan    : Darurat");

        System.out.println("\n=== Ruang Operasi 2 ===");
        System.out.println("Tim Penanganan: Tim Dr. Rizal Airin");
        System.out.println("Nama    Pasien                  :    "    +
antrianBaru.dequeue().toString());
        System.out.println("Penanganan    : Darurat");

        System.out.println("\n===========         Dalam         Antrian
==================");
        antrianBaru.display();

        Queue ruangPerawatan = new Queue();
        ruangPerawatan.enqueue(new Pasien("Anceloti Ferguso", 41, "Laki-
Laki", "Ringan", "VIP"));
        ruangPerawatan.enqueue(new     Pasien("Bana     Ilyana",     29,
"Perempuan", "Ringan", "VIP"));


System.out.println("\n=======================================");
        System.out.println("Dalam Penanganan Dokter Umum");

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

        System.out.println("\n=== Ruang Perawatan 1 ===");
        System.out.println("Tim Penanganan: Dr. Erling Haaland");
        System.out.println("Nama    Pasien                  :    "    +
ruangPerawatan.dequeue().toString());
        System.out.println("Penanganan    : Ringan");
```

```
        System.out.println("\n=== Ruang Perawatan 2 ===");
        System.out.println("Tim Penanganan: Dr. Ridwan Munir");
        System.out.println("Nama    Pasien                  :    "    +
ruangPerawatan.dequeue().toString());
        System.out.println("Penanganan    : Ringan");

        System.out.println("\n================Dalam
Antrian====================");
        ruangPerawatan.display();


System.out.println("\n=========================================");
        System.out.println("Daftar Tim Dokter Umum");

System.out.println("=========================================");
        System.out.println("1. Dr. Erling Haaland\n    Dokter Operasi:
Dr. Erling Haaland\n    Dokter Anestesi: \n    Perawat 1: Oscar Az\n
Perawat 2: -");
        System.out.println("\n2. Dr. Ridwan Munir\n    Dokter Operasi:
Dr. Ridwan Munir\n    Dokter Anestesi: \n    Perawat 1: Nia Valentino\n
Perawat 2: -");


System.out.println("\n=========================================");
        System.out.println("Daftar Tim Operasi");

System.out.println("=========================================");
        System.out.println("1. Tim Dr. Purwanto\n    Dokter Operasi: Dr.
Purwanto\n    Dokter Anestesi: Dr. Rewind HD\n    Perawat 1: Dwi Julian\n
Perawat 2: Exelsen");
        System.out.println("\n2. Tim Dr. Rizal Airin\n    Dokter Operasi:
Dr. Rizal Airin\n    Dokter Anestesi: Dr. Kyky Insania\n    Perawat 1:
Abyan Tubuh\n    Perawat 2: Rahmatdian");


System.out.println("\n=========================================");
        System.out.println("Operasi Sedang Berlangsung");

System.out.println("=========================================");
        System.out.println("\n==========Ruang                  Operasi
1===============");
```

```
        System.out.println("Tim Penanganan: Tim Dr. Faruk Ad-Dhar");
        System.out.println("Nama Pasien: Anakin Gonzales");
        System.out.println("Penanganan: Darurat");
        System.out.println("\n============Ruang                Operasi
2=====================");
        System.out.println("Tim   Penanganan:   Tim   Dr.   Anastasya
Maharani");
        System.out.println("Nama Pasien: Bruno Jigola");
        System.out.println("Penanganan: Berat");
        System.out.println("\n============Dalam
Antrian=====================");

        antrian.display();


System.out.println("\n=======================================");
        System.out.println("Daftar Pasien Pasca Penanganan");

System.out.println("=======================================");
        System.out.println("Megan Foz, 32, Perempuan, Darurat, VIP");
        System.out.println("Poor Didi, 54, Laki-Laki, Darurat, VIP");
        System.out.println("Anakin  Gonzales,  29,  Laki-Laki,  Darurat,
Reguler");
        System.out.println("Bruno Jigola, 25, Laki-Laki, Berat, VIP");
        System.out.println("Mitrime Ganthi, 37, Perempuan, Berat, VIP");
        System.out.println("Antoni  Dos  Santos,  32,  Laki-Laki,  Berat,
Reguler");
        System.out.println("Anceloti  Ferguso,  41,  Laki-Laki,  Ringan,
VIP");
        System.out.println("Bana Ilyana, 29, Perempuan, Ringan, VIP");
        System.out.println("Ali   Dakota,   20,   Laki-Laki,   Ringan,
Reguler");

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

public class Node {
    Object data;
    Node next;

    public Node(Object data) {
        this.data = data;
```

```
        this.next = null;
    }
}
```

Script "`public class Main {`" adalah definisi dari kelas Main, yang berfungsi sebagai titik masuk program. Kelas ini berisi metode main, yang merupakan metode pertama yang dijalankan saat program dieksekusi. Queue antrian = new Queue(); mendeklarasikan dan menginisialisasi objek antrian dari kelas Queue, yang digunakan untuk menyimpan data pasien. Antrian.display(); memanggil metode display pada objek antrian untuk menampilkan daftar pasien yang ada di dalamnya. Queue antrianBaru = new Queue(); adalah deklarasi objek baru antrianBaru, yang juga bertipe Queue, digunakan untuk menyimpan pasien yang baru datang.

Script "`public class Node {`" adalah definisi dari kelas Node, yang berfungsi sebagai elemen dalam struktur data linked list. Kelas ini memiliki dua atribut: Object data, yang menyimpan data dari node, dan Node next, yang mereferensikan node berikutnya dalam list.

1.5.2 Pasien & Queue

```
package Medium;

public class Pasien {
    private String nama;
    private int umur;
    private String jenisKelamin;
    private String penanganan;
    private String status;

    public Pasien(String nama, int umur, String jenisKelamin, String
penanganan, String status) {
        this.nama = nama;
        this.umur = umur;
        this.jenisKelamin = jenisKelamin;
        this.penanganan = penanganan;
        this.status = status;
    }

    @Override
    public String toString() {
        return nama + ", " + umur + ", " + jenisKelamin + ", " + penanganan
+ ", " + status;
    }
}
package Medium;
public class Queue {
    private SingleLinkedList list;

    public Queue() {
        list = new SingleLinkedList();
    }

    public void enqueue(Pasien pasien) {
        list.add(pasien);
    }
```

```
    public Pasien dequeue() {
        if (list.size() == 0) return null;
        Pasien data = (Pasien) list.getFirst();
        list.removeFirst();
        return data;
    }

    public void display() {
        list.display();
    }
}
```

Script "`public class Pasien{`" adalah definisi dari kelas Pasien, yang digunakan untuk merepresentasikan informasi pasien di rumah sakit. Kelas ini memiliki lima atribut privat: nama, umur, jenisKelamin, penanganan, dan status. Public Pasien(String nama, int umur, String jenisKelamin, String penanganan, String status) { adalah konstruktor yang digunakan untuk menginisialisasi objek Pasien dengan parameter yang diberikan. Parameter tersebut mengisi atribut kelas dengan informasi yang sesuai ketika objek Pasien dibuat. @Override public String toString() { adalah metode yang mengoverride metode toString dari kelas induk Object. Metode ini bertujuan untuk mengembalikan representasi string dari objek Pasien, mencetak nama, umur, jenis kelamin, penanganan, dan status pasien dalam format yang mudah dibaca. public void enqueue(Pasien pasien) { adalah metode yang digunakan untuk menambahkan objek Pasien ke dalam antrian. Metode ini memanggil metode add pada objek list untuk menambahkan pasien ke dalam linked list public Pasien dequeue() { adalah metode yang digunakan untuk mengeluarkan pasien dari antrian. Jika list kosong, metode ini mengembalikan null. Jika tidak, metode ini mendapatkan data pasien dari node pertama, menghapus node tersebut dari list, dan mengembalikan objek Pasien yang telah diambil. public void display() { adalah metode yang digunakan untuk menampilkan semua pasien dalam antrian dengan memanggil metode display pada objek list.

Script "`public class Queue{`" adalah definisi dari kelas Queue, yang digunakan untuk mengelola antrian pasien. Kelas ini memiliki atribut privat list, yang merupakan objek dari kelas SingleLinkedList. Public Queue() { adalah konstruktor yang menginisialisasi objek Queue dengan membuat instance baru dari SingleLinkedList.

1.5.3 SingleLinkedList, Stack & TimDokter

```
package Medium;

public class SingleLinkedList {
    private Node head;

    public SingleLinkedList() {
        head = null;
```

```
    }

    public void add(Object data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    public void display() {
        Node current = head;
        while (current != null) {
            System.out.print(current.data + (current.next != null ? "\n"
: ""));
            current = current.next;
        }
    }

    public int size() {
        int size = 0;
        Node current = head;
        while (current != null) {
            size++;
            current = current.next;
        }
        return size;
    }

    public void removeFirst() {
        if (head != null) {
            head = head.next;
        }
    }

    public Object getFirst() {
        return head != null ? head.data : null;
    }
}

package Medium;

public class Stack {
    private SingleLinkedList list;

    public Stack() {
        list = new SingleLinkedList();
    }

    public void push(Pasien pasien) {
        list.add(pasien);
    }

    public Pasien pop() {
        if (list.size() == 0) return null;
        Pasien data = (Pasien) list.getFirst();
```

```
        list.removeFirst();
        return data;
    }

    public void display() {
        list.display();
    }
}

package Medium;

public class TimDokter {
    String namaTim;
    String dokterOperasi;
    String dokterAnestesi;
    String perawat1;
    String perawat2;

      public TimDokter(String namaTim, String dokterOperasi, String
dokterAnestesi, String perawat1, String perawat2) {
        this.namaTim = namaTim;
        this.dokterOperasi = dokterOperasi;
        this.dokterAnestesi = dokterAnestesi;
        this.perawat1 = perawat1;
        this.perawat2 = perawat2;
    }

    @Override
    public String toString() {
          return "Nama Tim: " + namaTim + "\nDokter Operasi: " +
dokterOperasi +
               "\nDokter Anestesi: " + dokterAnestesi +
               "\nPerawat 1: " + perawat1 +
               "\nPerawat 2: " + perawat2 + "\n";
    }
}
```

Script "`public class SingleLinkedList {`" adalah definisi dari kelas SingleLinkedList, yang digunakan untuk merepresentasikan struktur data linked list. Kelas ini memiliki atribut privat head, yang menunjuk ke node pertama dalam linked list. public SingleLinkedList() { adalah konstruktor yang menginisialisasi objek SingleLinkedList dengan mengatur head menjadi null, menandakan bahwa linked list kosong pada saat inisialisasi. public void add(Object data) { adalah metode yang digunakan untuk menambahkan data baru ke dalam linked list. Metode ini membuat node baru dengan data yang diberikan dan menambahkan node tersebut di akhir list. Jika head adalah null, node baru menjadi node pertama. Jika tidak, metode ini menjelajahi list hingga menemukan node terakhir dan menambahkan node baru setelahnya. public void display() { adalah metode yang digunakan untuk menampilkan semua data dalam linked list. Metode ini menjelajahi setiap node dari head hingga akhir dan mencetak data setiap node, memisahkan setiap item dengan newline. public int size() { adalah metode yang menghitung dan mengembalikan jumlah total

node dalam linked list. Metode ini menggunakan loop untuk menjelajahi setiap node dan menghitungnya hingga mencapai node terakhir. public void removeFirst() { adalah metode yang digunakan untuk menghapus node pertama dalam linked list. Jika head tidak null, metode ini memperbarui head untuk menunjuk ke node berikutnya. public Object getFirst() { adalah metode yang mengembalikan data dari node pertama dalam linked list. Jika head tidak null, metode ini mengembalikan data dari node pertama, jika tidak, mengembalikan null.

Script "`public class Stack {`" adalah definisi dari kelas Stack, yang menggunakan SingleLinkedList untuk mengimplementasikan struktur data tumpukan (stack). Kelas ini memiliki atribut privat list, yang merupakan objek dari SingleLinkedList. public Stack() { adalah konstruktor yang menginisialisasi objek Stack dengan membuat instance baru dari SingleLinkedList. public void push(Pasien pasien) { adalah metode yang digunakan untuk menambahkan objek Pasien ke dalam tumpukan. Metode ini memanggil metode add pada objek list untuk menambahkan pasien. public Pasien pop() { adalah metode yang digunakan untuk mengeluarkan objek Pasien dari tumpukan. Jika list kosong, metode ini mengembalikan null. Jika tidak, metode ini mendapatkan data pasien dari node pertama, menghapus node tersebut dari list, dan mengembalikan objek Pasien. public void display() { adalah metode yang digunakan untuk menampilkan semua pasien dalam tumpukan dengan memanggil metode display pada objek list.

Script "`public class TimDokter {`" adalah definisi dari kelas TimDokter, yang digunakan untuk merepresentasikan informasi tentang tim dokter. Kelas ini memiliki lima atribut privat: namaTim, dokterOperasi, dokterAnestesi, perawat1, dan perawat2. public TimDokter(String namaTim, String dokterOperasi, String dokterAnestesi, String perawat1, String perawat2) { adalah konstruktor yang digunakan untuk menginisialisasi objek TimDokter dengan parameter yang diberikan untuk setiap atribut. @Override public String toString() { adalah metode yang mengoverride metode toString dari kelas induk Object. Metode ini mengembalikan representasi string dari objek TimDokter, mencetak nama tim, dokter operasi, dokter anestesi, dan perawat.