LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA

Oleh:
DZULFIKAR MUHAMMAD AL GHIFARI
NIM. 2341760071
SIB-1F / 08
D-IV SISTEM INFORMASI BISNIS
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

PRAKTIKUM 8

8.2 Percobaan 1

8.2.1 Langkah Langkah percobaan

1. Membuat package dan class

2. Deklarasi variable dan constructor

```
int[] data;
int front;
int rear;
int size;
int max;

public Queue08(int n) {
    max = n;
    data = new int[max];
    size = 0;
    front = rear = -1;
}
```

3. Menambahkan method isEmpty

```
public boolean IsEmpty() {
    if (size == 0) {
        return true;
    } else {
        return false;
    }
}
```

4. Menambahkan method isFull

```
public boolean IsFull() {
    if (size == max) {
        return true;
    } else {
        return false;
    }
}
```

5. Menambahkan method peek

```
public void peek() {
    if (!IsEmpty()) {
        System.out.println("Elemen terdepan: " + data[front]);
    } else {
        System.out.println(x:"Queue masih kosong");
    }
}
```

6. Menambahkan method print

```
public void print() {
    if (IsEmpty()) {
        System.out.println(x:"Queue masih kosong");
    } else {
        int i = front;
        while (i != rear) {
            System.out.print(data[i] + " ");
            i = (i + 1) % max;
        }
        System.out.println(data[i] + " ");
        System.out.println("Jumlah elemen = " + size);
    }
}
```

7. Menambahkan method clear

```
public void clear() {
    if (!IsEmpty()) {
        front = rear = -1;
        size = 0;
        System.out.println(x:"Queue behasil dikosongkan");
    } else {
        System.out.println(x:"Queue masih kosong");

    }
}
```

8. Menambahkan method Enqueue

```
public void Enqueue(int dt) {
    if (IsFull()) {
        System.out.println(x:"Queue sudah penuh");
    } else {
        if (IsEmpty()) {
            front = rear = 0;
        } else {
            if (rear == max - 1) {
                rear = 0;
            } else {
                rear++;
            }
        }
        data[rear] = dt;
        size++;
    }
}
```

9. Membuat method Dequeue

```
public int Dequeue() {
    int dt = 0;
    if (IsEmpty()) {
        System.out.println(x:"Queue masih kosong");
    } else {
        dt = data[front];
        size--;
        if (IsEmpty()) {
            front = rear = -1;
        } else {
            if (front == max - 1) {
                front = 0;
            } else {
                front++;
            }
        }
    }
    return dt;
}
```

10. Membuat class QueueMain

```
public class QueueMain {
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void menu() {
        System.out.println(x:"Masukkan operasi yang diinginkan:") ;
        System.out.println(x:"1. Enqueue") ;
        System.out.println(x:"2. Dequeue") ;
        System.out.println(x:"3. Print");
        System.out.println(x:"4, Peek");
        System.out.println(x:"5. Clear");
        System.out.println(x:"----------------------------------");
    }
}
```

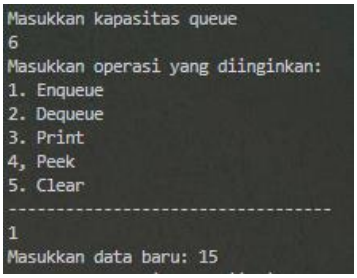
11. Membuat variabel n untuk menampung masukan

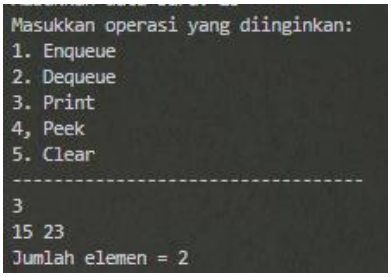
```
System.out.println(x:"Masukkan kapasistas queue");
int n = sc.nextInt();
```

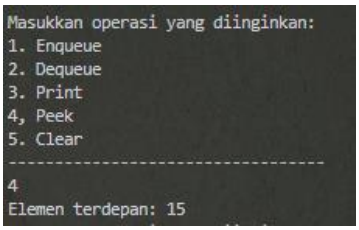
12. instansiasi object Q

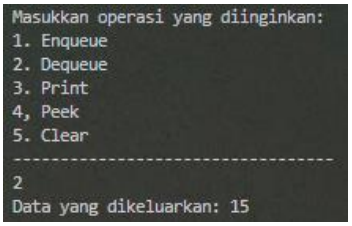
```
Queue08 Q = new Queue08(n);
```

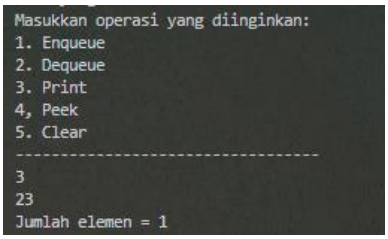
VERIFIKASI HASIL PERCOBAAN 8.2.2











PERTANYAAN 3.2.3

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?
2. Pada method Enqueue, jelaskan maksud dan kegunaan dari potongan kode berikut!

```
if (rear == max - 1) {
    rear = 0;
```

3. Pada method Dequeue, jelaskan maksud dan kegunaan dari potongan kode berikut!

```
if (front == max - 1) {
    front = 0;
```

4. Pada method print, mengapa pada proses perulangan variabel i tidak dimulai dari 0 (int i=0), melainkan int i=front?
5. Perhatikan kembali method print, jelaskan maksud dari potongan kode berikut!

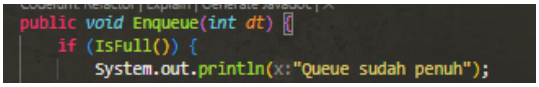
```
i = (i + 1) % max;
```

6. Tunjukkan potongan kode program yang merupakan queue overflow!
7. Pada saat terjadi queue overflow dan queue underflow, program tersebut tetap dapat berjalan dan hanya menampilkan teks informasi. Lakukan modifikasi program sehingga pada saat terjadi queue overflow dan queue underflow, program dihentikan!

Jawaban

1. Atribut front dan rear bernilai -1 untuk menyatakan bahwa kondisi masih dalam keadaan kosong. Attribut size bernilai 0, karena indexs dimulai dari ke 0.
2. Jika rear berada pada posisi indexs terakhir, penambahan data baru akan di tempatkan pada index ke 0
3. Jika front berada pada posisi indexs terakhir, penambahan data baru akan di tempatkan pada index ke -0.

4. Karena front tidak selalu pada indeks ke-0, sedangkan perulangan dimulai dengan posisi front.
5. Jika nilai i tidak berposisi sebagai rear, maka akan dilakukan increment dan dimodulus dengan nilai max atau kapasitas dari Queue tersebut.
6. Potongan kode queue overflow



```
public void Enqueue(int dt) {
    if (IsFull()) {
        System.out.println(x:"Queue sudah penuh");
```

7. Dengan melakukan perubahan seprti berikut

```
public void Enqueue(int dt) {
    if (IsFull()) {
        System.out.println(x:"Queue sudah penuh");
        System.exit(status:0);
    } else {
        if (IsEmpty()) {
            front = rear = 0;
        } else {
            if (rear == max - 1) {
                rear = 0;
            } else {
                rear++;
            }
        }
        data[rear] = dt;
        size++;
    }
}

Codeium: Refactor | Explain | Generate Javadoc | ×
public int Dequeue() {
    int dt = 0;
    if (IsEmpty()) {
        System.out.println(x:"Queue masih kosong");
        System.exit(status:0);
    } else {
        dt = data[front];
        size--;
        if (IsEmpty()) {
            front = rear = -1;
        } else {
            if (front == max - 1) {
                front = 0;
            } else {
                front++;
            }
        }
    }
    return dt;
}
```

8.3 Percobaan 2

8.3.1 Langkah-langkah Percobaan

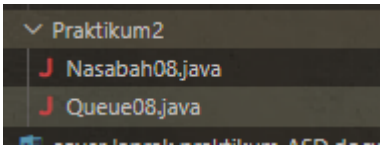
1. Menambahkan class nasabah dan instansiasi variabel

```
Codeium: Refactor | Explain
public class Nasabah08 {
    String norek, nama, alamat;
    int umur;
    double saldo;
```

2. Menambahkan konstruktor

```
Nasabah08 (String norek, String nama, String alamat, int umur, double saldo) {
    this.norek = norek;
    this.nama = nama;
    this.alamat = alamat;
    this. umur = umur;
    this.saldo = saldo;
}
```

3. Menyalin kode program queue pada praktikum 1



4. Melakukan perumahan pada queue menyesuaikan menggunakan object

```
public class Queue08 {
    Nasabah08[] data;
    int front;
    int rear;
    int size;
```

```
public Queue08(int n) {
    max = n;
    data = new Nasabah08[max];
    size = 0;
    front = rear = -1;
}
```

```
Codeium: Refactor | Explain | Generate Javadoc |
public void Enqueue(Nasabah08 dt) {
    if (IsFull()) {
        System.out.println(x:"Queue suc
        // System.exit(0);
```

```
public Nasabah08 Dequeue() {
    Nasabah08 dt = new Nasabah08();
    if (IsEmpty()) {
        System.out.println(x:"Queue ma
```

5. Mengubah method peek dan print

```
public void peek() {
    if (!IsEmpty()) {
        System.out.println("Elemen terdepan: " + data[front].norek + " " + data[front].
        nama + " " + data[front].alamat + " " + data[front].umur + " " + data[front].
        saldo);
    } else {
        System.out.println(x:"Queue masih kosong");
    }
}
```

```
public void print() {
    if (IsEmpty()) {
        System.out.println(x:"Queue masih kosong");
    } else {
        int i = front;
        while (i != rear) {
            System.out.println(data[i].norek + " " + data[i].nama + " " + data[i].
            alamat + " " + data[i].umur + " " + data[i].saldo);
            i = (i + 1) % max;
        }
        System.out.println(data[i].norek + " " + data[i].nama + " " + data[i].alamat +
        " " + data[i].umur + " " + data[i].saldo);
        System.out.println("Jumlah elemen = " + size);
    }
}
```

6. Menambahkan class queue main

```
Codeium: Refactor | Explain
public class QueueMain {
    Codeium: Refactor | Explain | Generate Javadoc |
    public static void menu () {
        System. out.println(x:"Pilih menu: ");
        System. out.println(x:"1. Antrian baru") ;
        System. out.println(x:"2. Antrian keluar");
        System. out.println(x:"3. Cek Antrian terdepan");
        System. out.println(x:"4. Cek Semua Antrian");
        System. out. println(x:"---------------------------- ");
    }
}
```

7. Menambahkan class main

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
}
```

8. Menambahkan input dan instansiasi objek

```
System.out.print(s:"Masukkan kapasitas queue: ");
int jumlah = sc.nextInt();
Queue08 antri = new Queue08(jumlah);
```

9. Menambahkan deklarasi variable pilih

```
int pilih;
```

VERIFIKASI HASIL PERCOBAAN 7.3.2

```
Masukkan kapasitas queue: 4
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
----------------------------
1
No Rekening: 12000123
Nama: Arif
Alamat: Malang
Umur: 12
Saldo: 1200000
```

```
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
----------------------------
4
12000123 Arif Malang 12 1200000.0
12000124 Dewi Surabaya 30 860000.0
Jumlah elemen = 2
```

```
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
----------------------------
3
Elemen terdepan: 12000123 Arif Malang 12 1200000.0
```

```
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
----------------------------
2
Antrian yang keluar: 12000123 Arif Malang 12 1200000.0
```

```
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
----------------------------
4
12000124 Dewi Surabaya 30 860000.0
Jumlah elemen = 1
```

PERTANYAAN 8.3.3

1. Pada class QueueMain, jelaskan fungsi IF pada potongan kode program berikut!

```
if (!"".equals(data.norek) && !"".equals(data.nama) && !"".equals(data.alamat)
        && data.umur != 0 && data.saldo != 0) {
    System.out.println("Antrian yang keluar: " + data.norek + " " + data.nama + " "
            + data.alamat + " " + data.umur + " " + data.saldo);
    break;
}
```

2. Lakukan modifikasi program dengan menambahkan method baru bernama peekRear pada class Queue yang digunakan untuk mengecek antrian yang berada di posisi belakang! Tambahkan pula daftar menu 5. Cek Antrian paling belakang pada class QueueMain sehingga method peekRear dapat dipanggil!

Jawaban

1. .equals adalah perintah untuk membandingkan dua string. Pada kode program tersebut di bandingkan apakah list data pada object data berisikan string kosong jika tidak berisi string kosong maka akan print antrian yang keluar. Jika tidak maka tidak akan menghasilkan output apapun

2.

```
public void peekRear(){
    if(!IsEmpty()){
        System.out.println("Elemen terdepan : "+data[rear].norek+ " "+data[rear].nama
        +" "+data[rear].alamat+" "+data[rear].umur+" "+data[rear].saldo);
    }else{
        System.out.println("Queue masih kosong");
    }
}
```

```
public static void menu() {
    System.out.println("Pilih menu: ");
    System.out.println("1. Antrian baru");
    System.out.println("2. Antrian keluar");
    System.out.println("3. Cek Antrian terdepan");
    System.out.println("4. Cek Semua Antrian");
    System.out.println("5. Cek Antrian belakang");
    System.out.println("---------------------------- ");
}
```

```
case 5:
    antri.peekRear();
    break;
```

```
Masukkan kapasitas queue: 4
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
5. Cek Antrian belakang
----------------------------
1
No Rekening: 123123
Nama: Agus
Alamat: Malang
Umur: 34
Saldo: 76883400
```

```
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
5. Cek Antrian belakang
----------------------------
1
No Rekening: 123234
Nama: Budi
Alamat: Surabaya
Umur: 14
Saldo: 99999992
```

```
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
5. Cek Antrian belakang
----------------------------
5
Elemen terdepan : 123234 Budi Surabaya 14 9.9999992E7
```

8.4 TUGAS

1. Buatlah program antrian untuk mengilustrasikan antrian pasien di sebuah klinik. Ketika seorang pasien akan mengantri, maka dia harus mendaftarkan nama, nomor identitas, jenis kelamin dan umur seperti yang digambarkan pada Class diagram berikut:

Pembeli
nama: String noID: int jenisKelamin: char umur: int
Pasien (nama: String, noID: int, jenisKelamin: char, umur: int)

Class diagram Queue digambarkan sebagai berikut:

Queue
antrian: Pasien[] front: int rear: int size: int max: int
Queue(n: int) isEmpty(): boolean isFull(): boolean enqueue(antri: Pasien): void dequeue(): int print(): void peek(): void peekRear(): void peekPosition(nama: String): void daftarPasien(): void

Keterangan method:

• Method create(), isEmpty(), isFull(), enqueue(), dequeue() dan print(), kegunaannya sama seperti yang telah dibuat pada Praktikum

• Method peek(): digunakan untuk menampilkan data Pasien yang berada di posisi antrian paling depan

• Method peekRear(): digunakan untuk menampilkan data Pasien yang berada di posisi antrian paling belakang

• Method peekPosition(): digunakan untuk menampilkan seorang pasien (berdasarkan nama) posisi antrian ke berapa

• Method daftarPasien(): digunakan untuk menampilkan data seluruh pasien

Jawaban

Membuat class pembeli sebagai berikut

```
Tugas > J Pembeli08.java > ...
1   package Tugas;
2
    Codeium: Refactor | Explain
3   public class Pembeli08 {
4       String nama;
5       int noId, umur;
6       char jk;
7
8       Pembeli08(){}
9
10      Pembeli08 (String nama, int noId, char jk, int umur) {
11          this.nama = nama;
12          this.noId = noId;
13          this. jk = jk;
14          this.umur = umur;
15      }
16  }
```

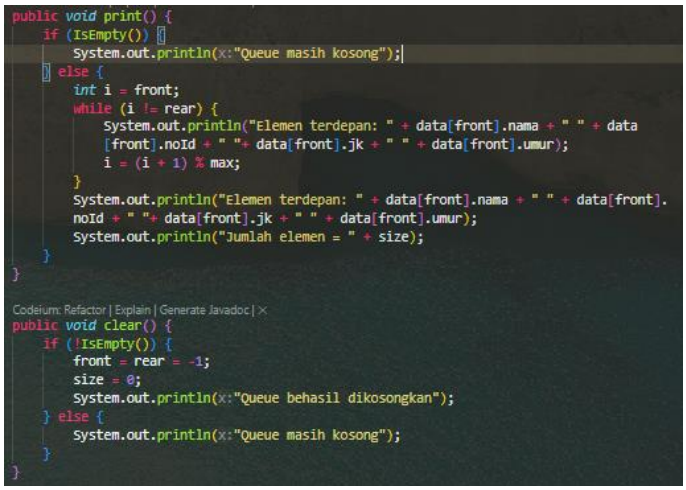
Membuat class queue sebagai berikut

```
package Tugas;

Codeium: Refactor | Explain
public class Queue08 {
    Pembeli08[] data;
    int front;
    int rear;
    int size;
    int max;

    public Queue08(int n) {
        max = n;
        data = new Pembeli08[max];
        size = 0;
        front = rear = -1;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public boolean IsEmpty() {
        if (size == 0) {
            return true;
        } else {
            return false;
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public boolean IsFull() {
        if (size == max) {
            return true;
        } else {
            return false;
        }
    }
```

Method peek dan peek rear

```
Codeium: Refactor | Explain | Generate Javadoc | ×
public void peek() {
    if (!IsEmpty()) {
        System.out.println("Elemen terdepan: " + data[front].nama + " " + data[front].
        noId + " "+ data[front].jk + " " + data[front].umur);
    } else {
        System.out.println(x:"Queue masih kosong");
    }
}

Codeium: Refactor | Explain | Generate Javadoc | ×
public void peekRear(){
    if(!IsEmpty()){
        System.out.println("Elemen terbelakang: " + data[rear].nama + " " + data[rear].
        noId + " "+ data[rear].jk + " " + data[rear].umur);
    }else{
        System.out.println(x:"Queue masih kosong");
    }
}
```

Method print & clear



```
public void print() {
    if (IsEmpty()) {
        System.out.println(x:"Queue masih kosong");
    } else {
        int i = front;
        while (i != rear) {
            System.out.println("Elemen terdepan: " + data[front].nama + " " + data
            [front].noId + " "+ data[front].jk + " " + data[front].umur);
            i = (i + 1) % max;
        }
        System.out.println("Elemen terdepan: " + data[front].nama + " " + data[front].
        noId + " "+ data[front].jk + " " + data[front].umur);
        System.out.println("Jumlah elemen = " + size);
    }
}

Codeium: Refactor | Explain | Generate Javadoc | ×
public void clear() {
    if (!IsEmpty()) {
        front = rear = -1;
        size = 0;
        System.out.println(x:"Queue behasil dikosongkan");
    } else {
        System.out.println(x:"Queue masih kosong");
    }
}
```

Method Enqueue & Dequeue

```
public void Enqueue(Pembeli08 dt) {
    if (IsFull()) {
        System.out.println(x:"Queue sudah penuh");
        // System.exit(0);
    } else {
        if (IsEmpty()) {
            front = rear = 0;
        } else {
            if (rear == max - 1) {
                rear = 0;
            } else {
                rear++;
            }
        }
        data[rear] = dt;
        size++;
    }
}

Codeium: Refactor | Explain | Generate Javadoc | ×
public Pembeli08 Dequeue() {
    Pembeli08 dt = new Pembeli08();
    if (IsEmpty()) {
        System.out.println(x:"Queue masih kosong");
        // System.exit(0);
    } else {
        dt = data[front];
        size--;
        if (IsEmpty()) {
            front = rear = -1;
        } else {
            if (front == max - 1) {
                front = 0;
            } else {
                front++;
            }
        }
    }
    return dt;
}
```

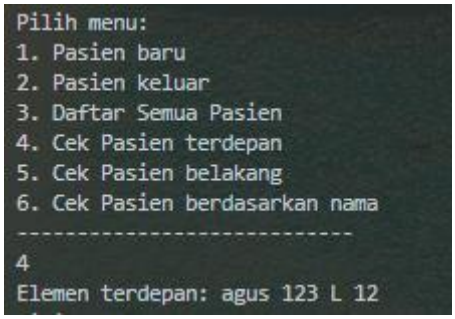
Method peekPosition & daftarPasien

```
public int peekPosition(String nama) {
    if (IsEmpty()) {
      return -1;
    }

    int i = front;
    int position = 0;
    while (i != rear) {
      if (data[i].nama.equals(nama)) {
        System.out.println("Nama: " + data[i].nama + ", No. ID: " + data[i].noId + ",
        JK: " + data[i].jk + ", Umur: " + data[i].umur);
        return position;
      }
      position++;
      i = (i + 1) % max;
    }

    if (data[i].nama.equals(nama)) {
        System.out.println("Nama: " + data[i].nama + ", No. ID: " + data[i].noId + ",
        JK: " + data[i].jk + ", Umur: " + data[i].umur);

      return position;
    }

    return -1;
  }

Codeium: Refactor | Explain | Generate Javadoc | ×
public void daftarPasien() {
    if (IsEmpty()) {
      System.out.println(x:"Daftar pasien masih kosong");
    } else {
      int i = front;
      while (i != rear) {
        System.out.println("Nama: " + data[i].nama + ", No. ID: " + data[i].noId
                           + ", JK: " + data[i].jk + ", Umur: " + data[i].umur);
        i = (i + 1) % max;
      }

      System.out.println("Nama: " + data[i].nama + ", No. ID: " + data[i].noId
                         + ", JK: " + data[i].jk + ", Umur: " + data[i].umur);
    }
  }
```

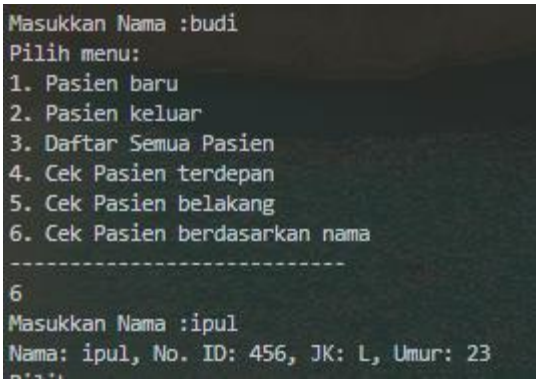
Hasil compile

```
Masukkan kapasitas queue: 5
Pilih menu:
1. Pasien baru
2. Pasien keluar
3. Daftar Semua Pasien
4. Cek Pasien terdepan
5. Cek Pasien belakang
6. Cek Pasien berdasarkan nama
----------------------------
1
Nama: agus
No ID: 123
Jenis Kelamin (L/P): L
Umur: 12
Pilih menu:
1. Pasien baru
2. Pasien keluar
3. Daftar Semua Pasien
4. Cek Pasien terdepan
5. Cek Pasien belakang
6. Cek Pasien berdasarkan nama
----------------------------
1
Nama: ipul
No ID: 456
Jenis Kelamin (L/P): L
Umur: 23
Pilih menu:
1. Pasien baru
2. Pasien keluar
3. Daftar Semua Pasien
4. Cek Pasien terdepan
5. Cek Pasien belakang
6. Cek Pasien berdasarkan nama
----------------------------
1
Nama: putput
No ID: 678
Jenis Kelamin (L/P): P
Umur: 21
```

```
Pilih menu:
1. Pasien baru
2. Pasien keluar
3. Daftar Semua Pasien
4. Cek Pasien terdepan
5. Cek Pasien belakang
6. Cek Pasien berdasarkan nama
----------------------------
3
Nama: agus, No. ID: 123, JK: L, Umur: 12
Nama: ipul, No. ID: 456, JK: L, Umur: 23
Nama: putput, No. ID: 678, JK: P, Umur: 21
```


Pilih menu:
1. Pasien baru
2. Pasien keluar
3. Daftar Semua Pasien
4. Cek Pasien terdepan
5. Cek Pasien belakang
6. Cek Pasien berdasarkan nama

4
Elemen terdepan: agus 123 L 12


Masukkan Nama :budi
Pilih menu:
1. Pasien baru
2. Pasien keluar
3. Daftar Semua Pasien
4. Cek Pasien terdepan
5. Cek Pasien belakang
6. Cek Pasien berdasarkan nama

6
Masukkan Nama :ipul
Nama: ipul, No. ID: 456, JK: L, Umur: 23