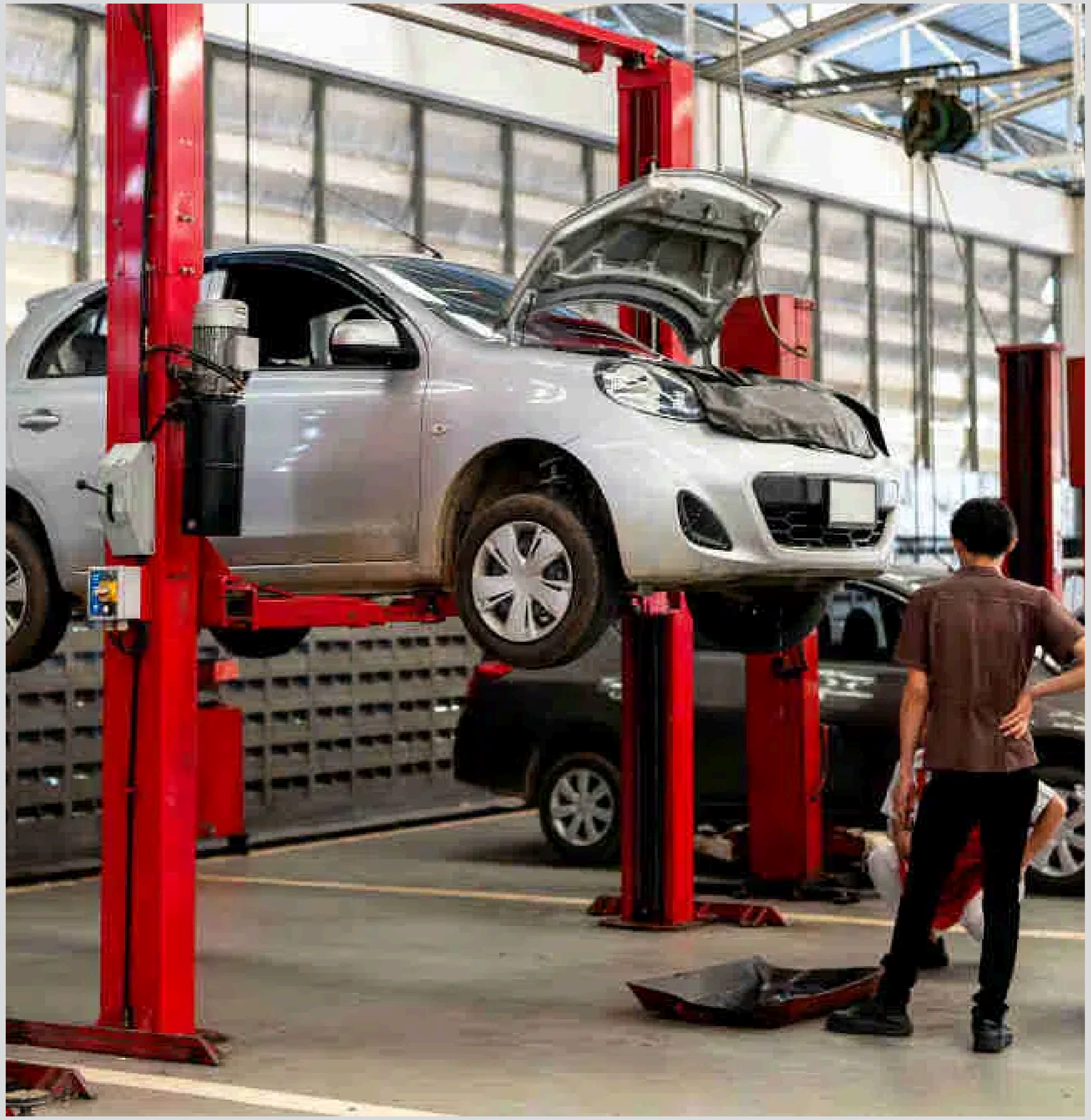
PROJECT AKHIR PEMOGRAMAN LANJUT

SISTEM ANTRIAN SERVICE MOBIL SHOP & DRIVE

RAISA SHAHLA KHALISA
235150701111022



TUJUAN

- Membuat sistem yang mengelola antrian pelayanan pelanggan dengan efisien.
- Membantu pusat pelayanan mengelola pelanggan dan memproses mereka dengan cara yang terorganisir.

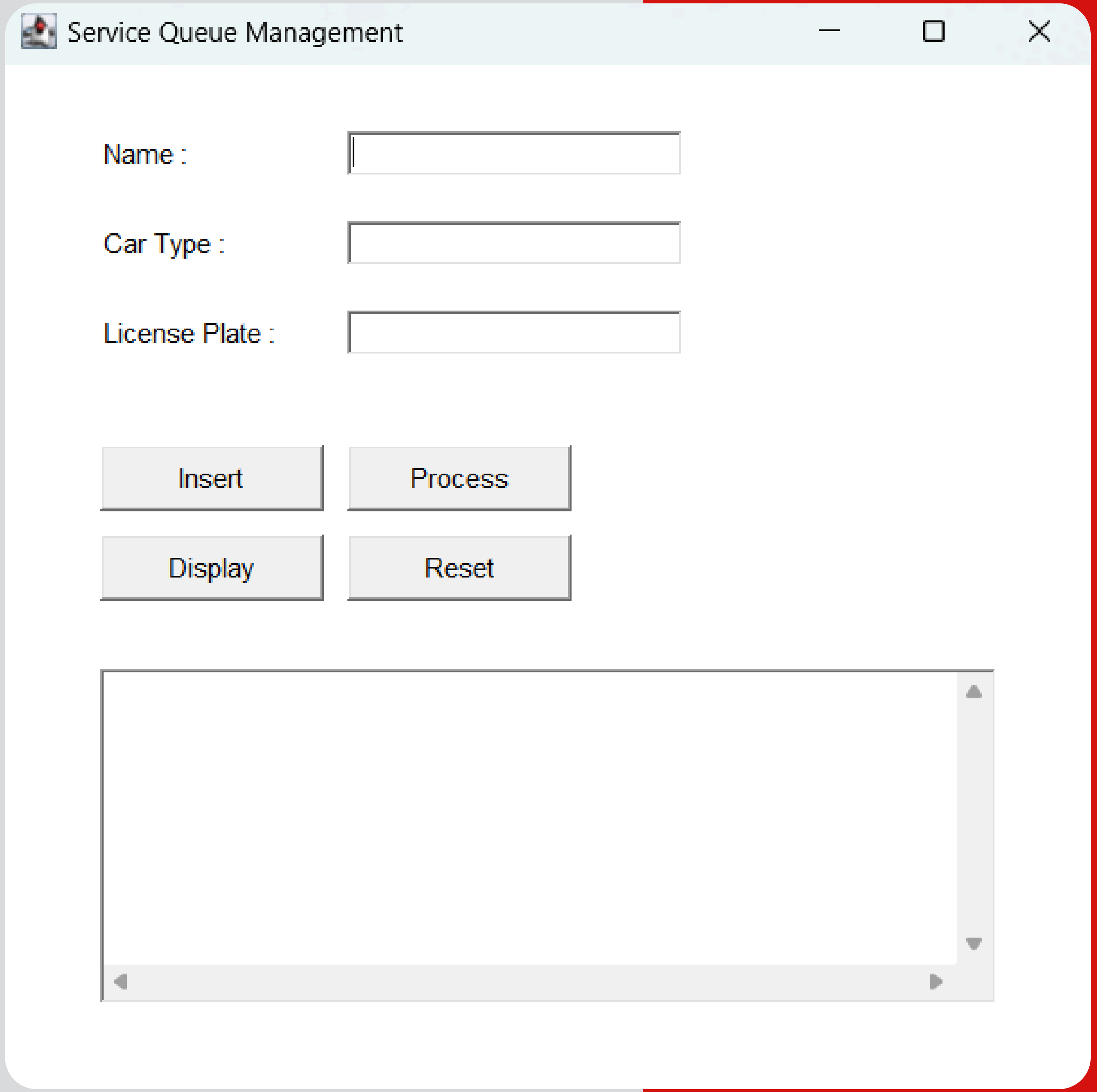
FITUR

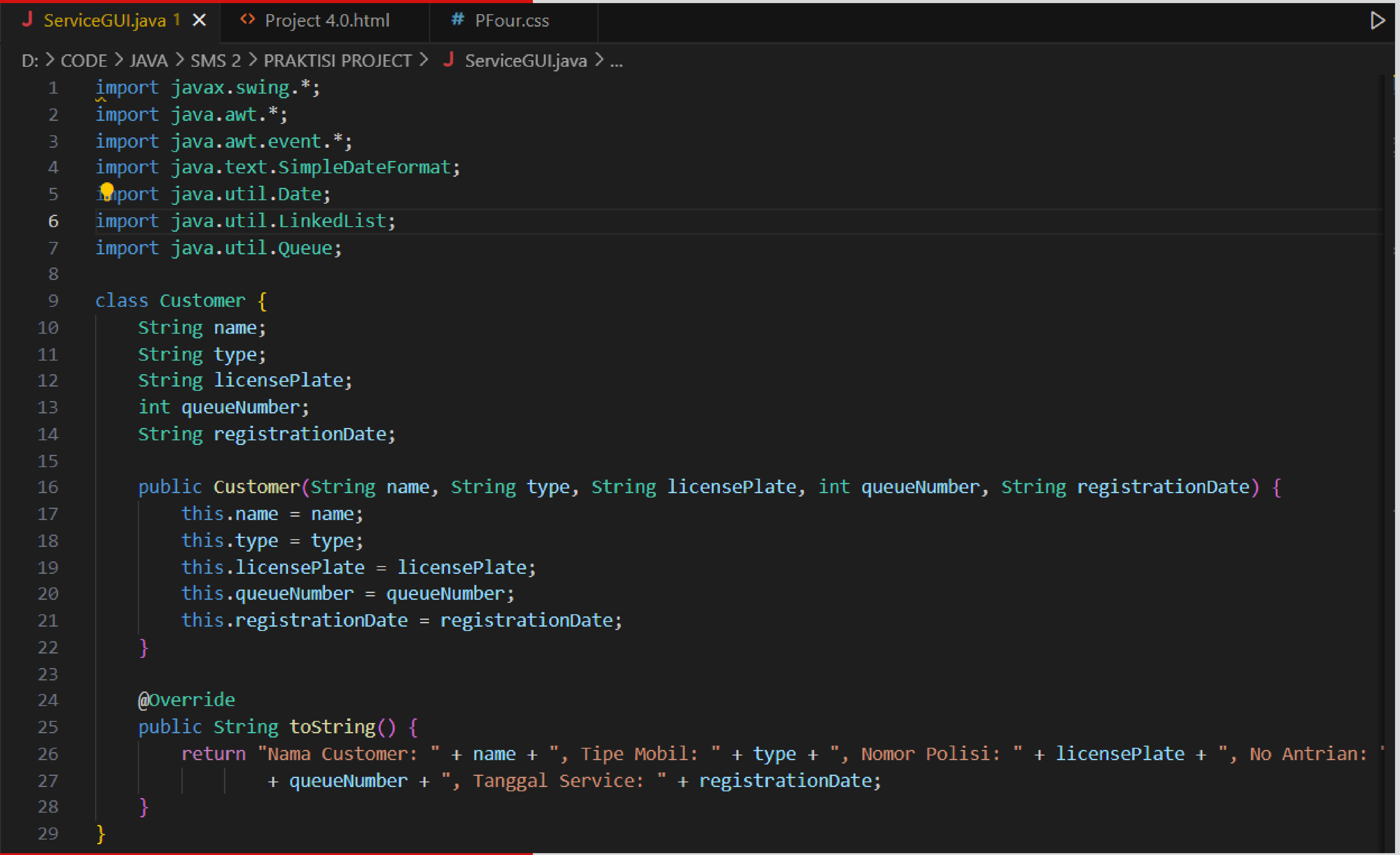
01 • Menambahkan detail pelanggan ke dalam antrian.

02 • Memproses pelanggan secara urutan kedatangan.

03 • Menampilkan status antrian saat ini.

04 • Mereset antrian bila diperlukan.




ServiceGUI.java 1
Project 4.0.html
PFour.css
D: > CODE > JAVA > SMS 2 > PRAKTISI PROJECT > ServiceGUI.java > ...

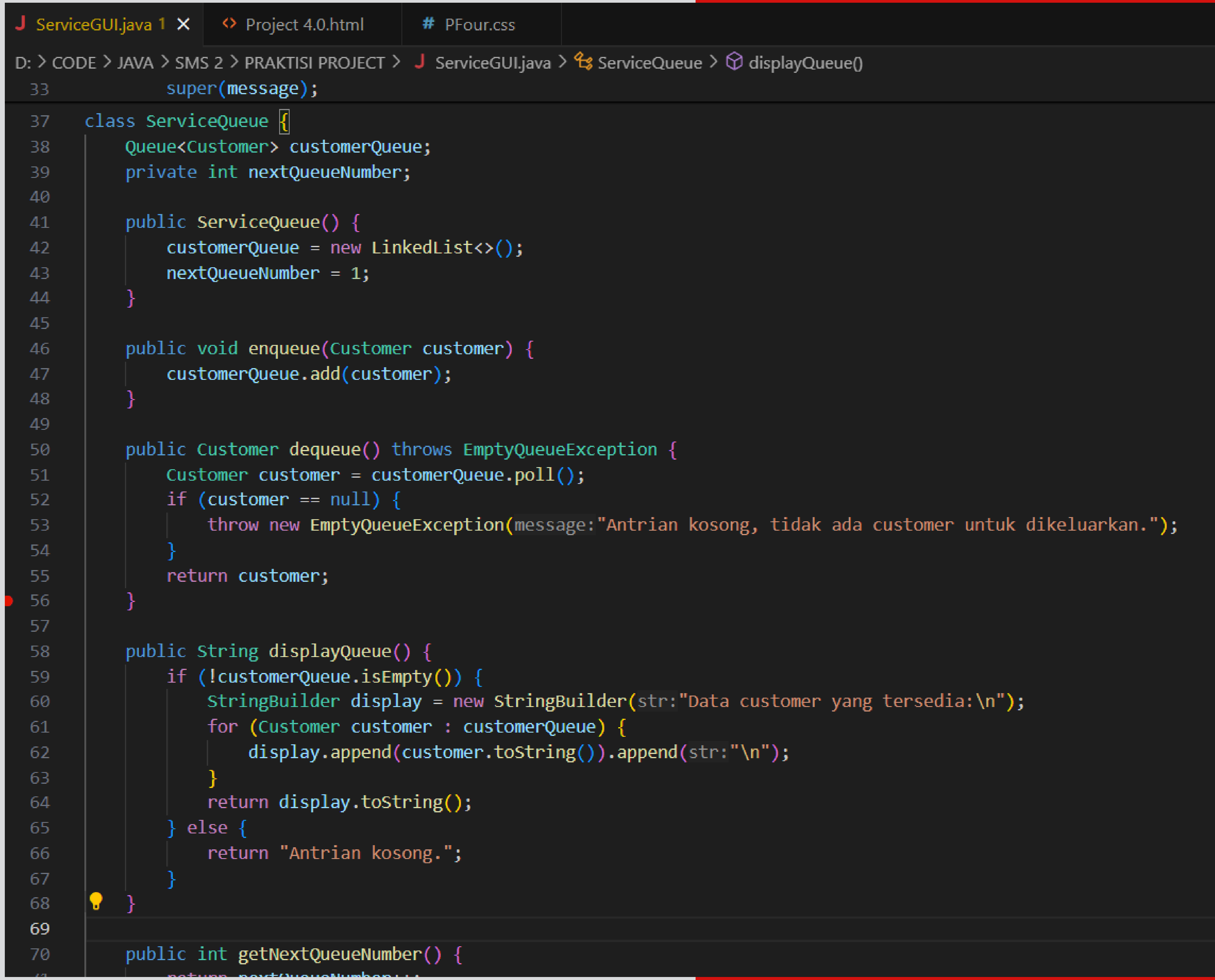
```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.LinkedList;
import java.util.Queue;

class Customer {
    String name;
    String type;
    String licensePlate;
    int queueNumber;
    String registrationDate;

    public Customer(String name, String type, String licensePlate, int queueNumber, String registrationDate) {
        this.name = name;
        this.type = type;
        this.licensePlate = licensePlate;
        this.queueNumber = queueNumber;
        this.registrationDate = registrationDate;
    }

    @Override
    public String toString() {
        return "Nama Customer: " + name + ", Tipe Mobil: " + type + ", Nomor Polisi: " + licensePlate + ", No Antrian: "
                + queueNumber + ", Tanggal Service: " + registrationDate;
    }
}
```

- **Atribut:**
    - **name**: Nama pelanggan.
    - **type**: Tipe mobil.
    - **licensePlate**: Nomor polisi mobil.
    - **queueNumber**: Nomor antrian unik.
    - **registrationDate**: Tanggal dan waktu pendaftaran.
- **Metode:**
    - **toString()**: Mengembalikan representasi string dari detail pelanggan.

KELAS CUSTOMER

KELAS SERVICEQUEUE

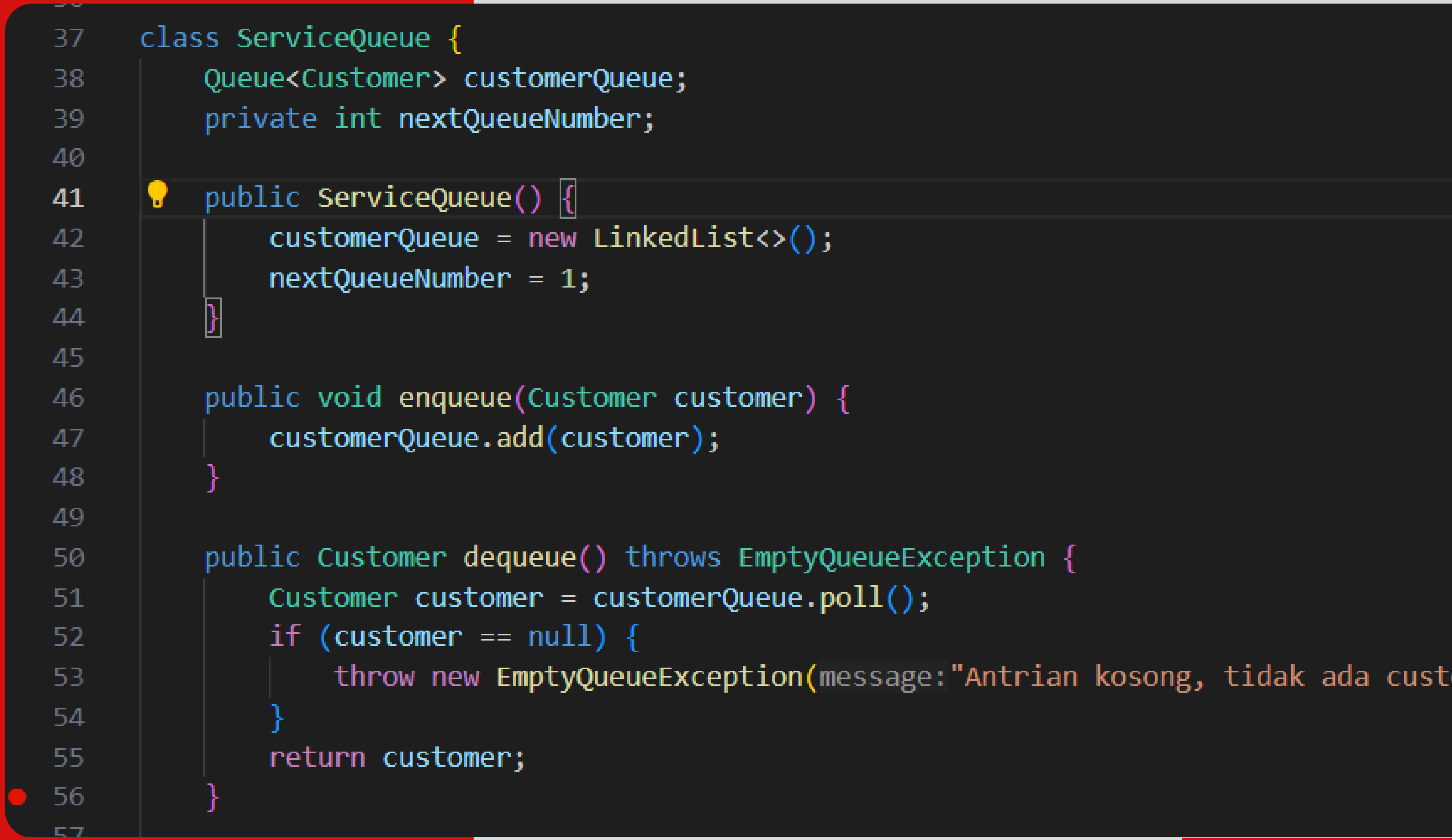
- **Atribut:**
  - **customerQueue**: Antrian untuk menampung pelanggan.
  - **nextQueueNumber**: Melacak nomor antrian berikutnya yang tersedia.
- **Metode:**
  - **enqueue(Customer customer)**: Menambahkan pelanggan ke dalam antrian.
  - **dequeue()**: Menghapus dan mengembalikan pelanggan berikutnya dalam antrian.
  - **displayQueue()**: Menampilkan status antrian saat ini.
  - **getNextQueueNumber()**: Menghasilkan nomor antrian berikutnya.
  - **resetQueue()**: Mereset antrian dan nomor antrian.


ServiceGUI.java 1
Project 4.0.html
PFour.css
D: > CODE > JAVA > SMS 2 > PRAKTISI PROJECT > ServiceGUI.java > ServiceQueue > displayQueue()

```java
        super(message);

class ServiceQueue {
    Queue<Customer> customerQueue;
    private int nextQueueNumber;

    public ServiceQueue() {
        customerQueue = new LinkedList<>();
        nextQueueNumber = 1;
    }

    public void enqueue(Customer customer) {
        customerQueue.add(customer);
    }

    public Customer dequeue() throws EmptyQueueException {
        Customer customer = customerQueue.poll();
        if (customer == null) {
            throw new EmptyQueueException(message:"Antrian kosong, tidak ada customer untuk dikeluarkan.");
        }
        return customer;
    }

    public String displayQueue() {
        if (!customerQueue.isEmpty()) {
            StringBuilder display = new StringBuilder(str:"Data customer yang tersedia:\n");
            for (Customer customer : customerQueue) {
                display.append(customer.toString()).append(str:"\n");
            }
            return display.toString();
        } else {
            return "Antrian kosong.";
        }
    }

    public int getNextQueueNumber() {
```

JAVA GENERIC

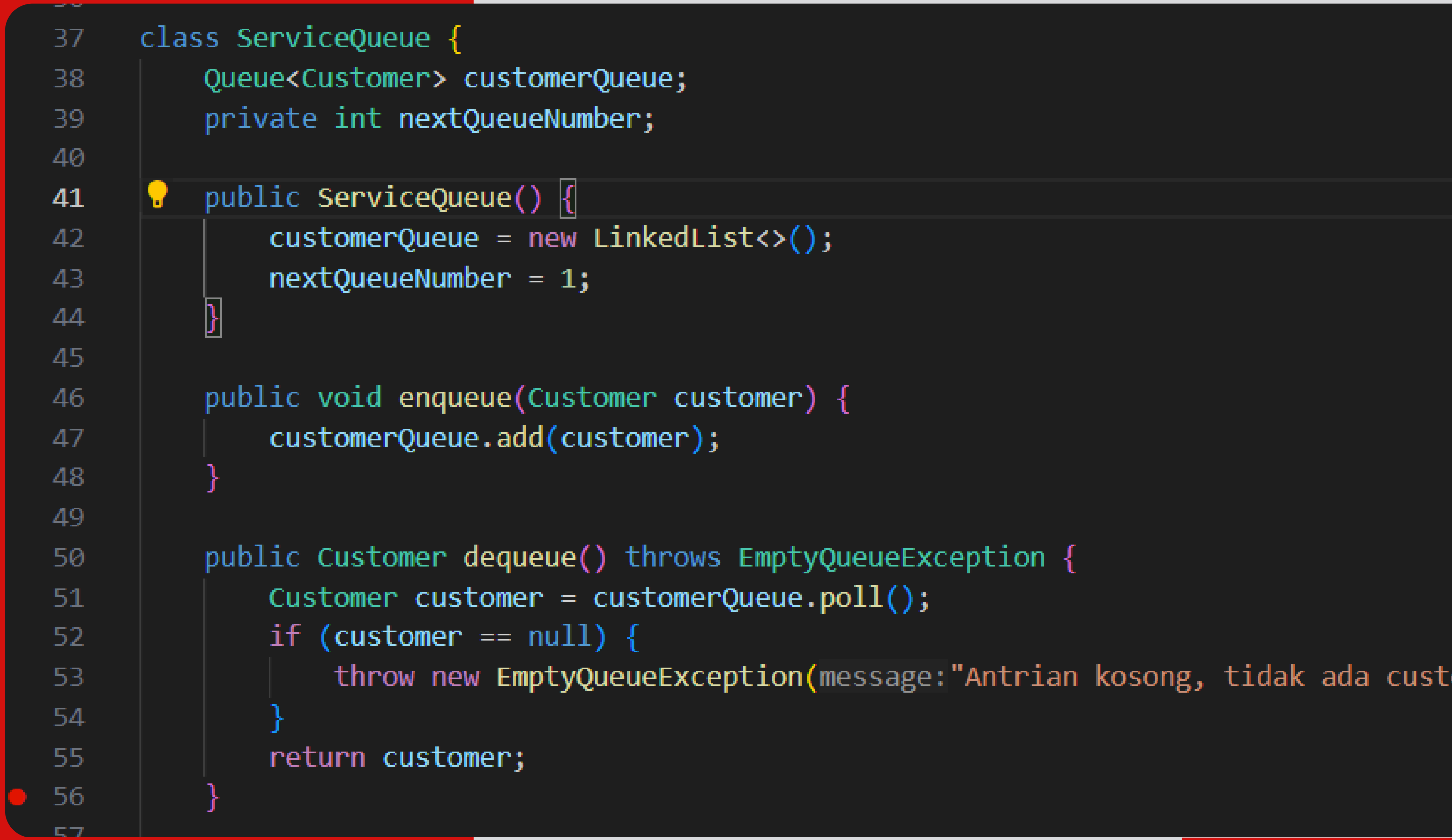
- **Penggunaan Generics:**
  - Meningkatkan keamanan tipe dan fleksibilitas kode.
  - **Queue<Customer>** memastikan hanya objek **Customer** yang bisa masuk antrian.



```
class ServiceQueue {
    Queue<Customer> customerQueue;
    private int nextQueueNumber;

    public ServiceQueue() {
        customerQueue = new LinkedList<>();
        nextQueueNumber = 1;
    }

    public void enqueue(Customer customer) {
        customerQueue.add(customer);
    }

    public Customer dequeue() throws EmptyQueueException {
        Customer customer = customerQueue.poll();
        if (customer == null) {
            throw new EmptyQueueException(message:"Antrian kosong, tidak ada cust
        }
        return customer;
    }
```

JAVA COLLECTION

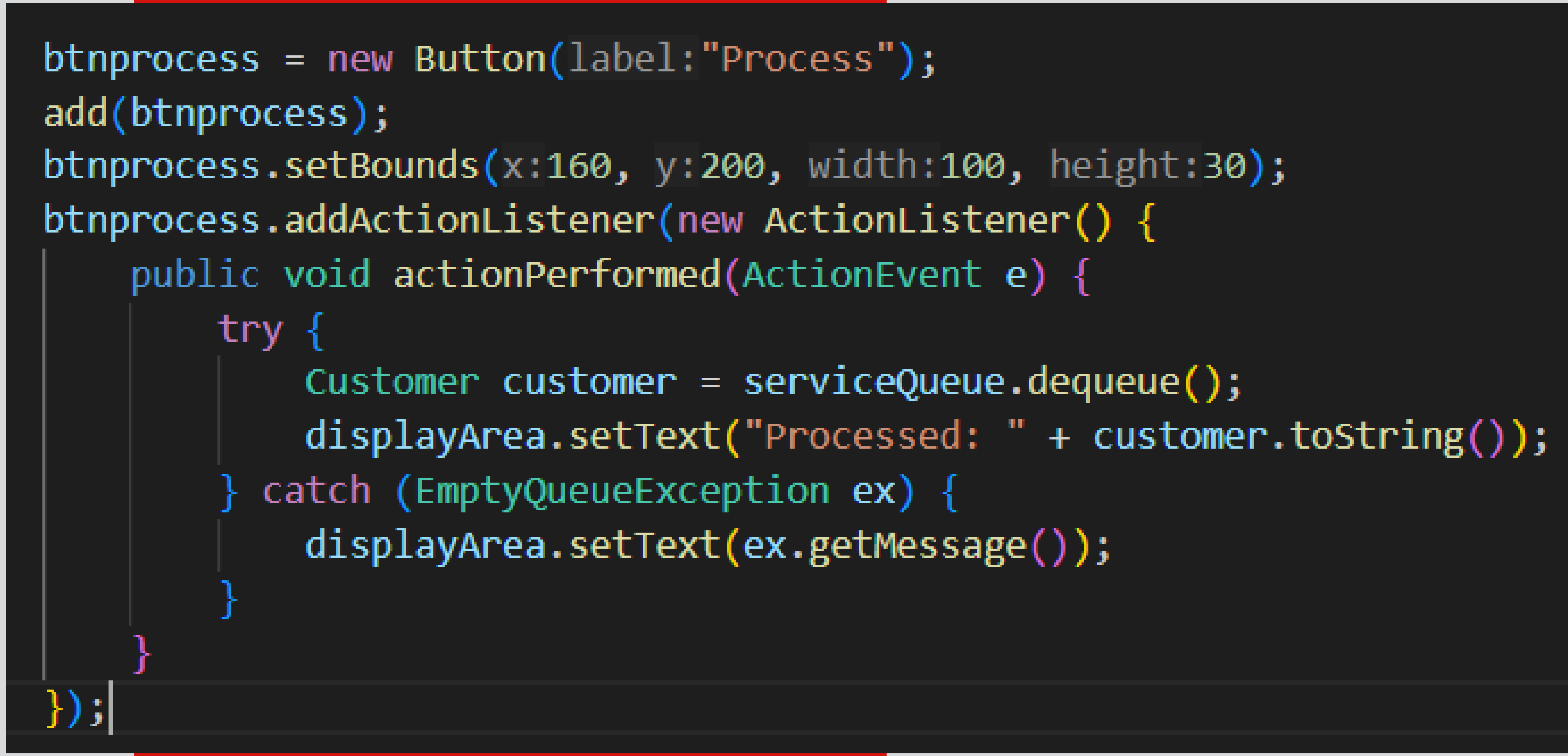
- **Penggunaan Queue:**
  - Implementasi menggunakan **LinkedList** untuk mendukung operasi antrian.
  - **Queue<Customer> customerQueue** menyimpan objek pelanggan dalam urutan kedatangan.



```java
class ServiceQueue {
    Queue<Customer> customerQueue;
    private int nextQueueNumber;

    public ServiceQueue() {
        customerQueue = new LinkedList<>();
        nextQueueNumber = 1;
    }

    public void enqueue(Customer customer) {
        customerQueue.add(customer);
    }

    public Customer dequeue() throws EmptyQueueException {
        Customer customer = customerQueue.poll();
        if (customer == null) {
            throw new EmptyQueueException(message:"Antrian kosong, tidak ada cust
        }
        return customer;
    }
```

JAVA EXCEPTION HANDLING

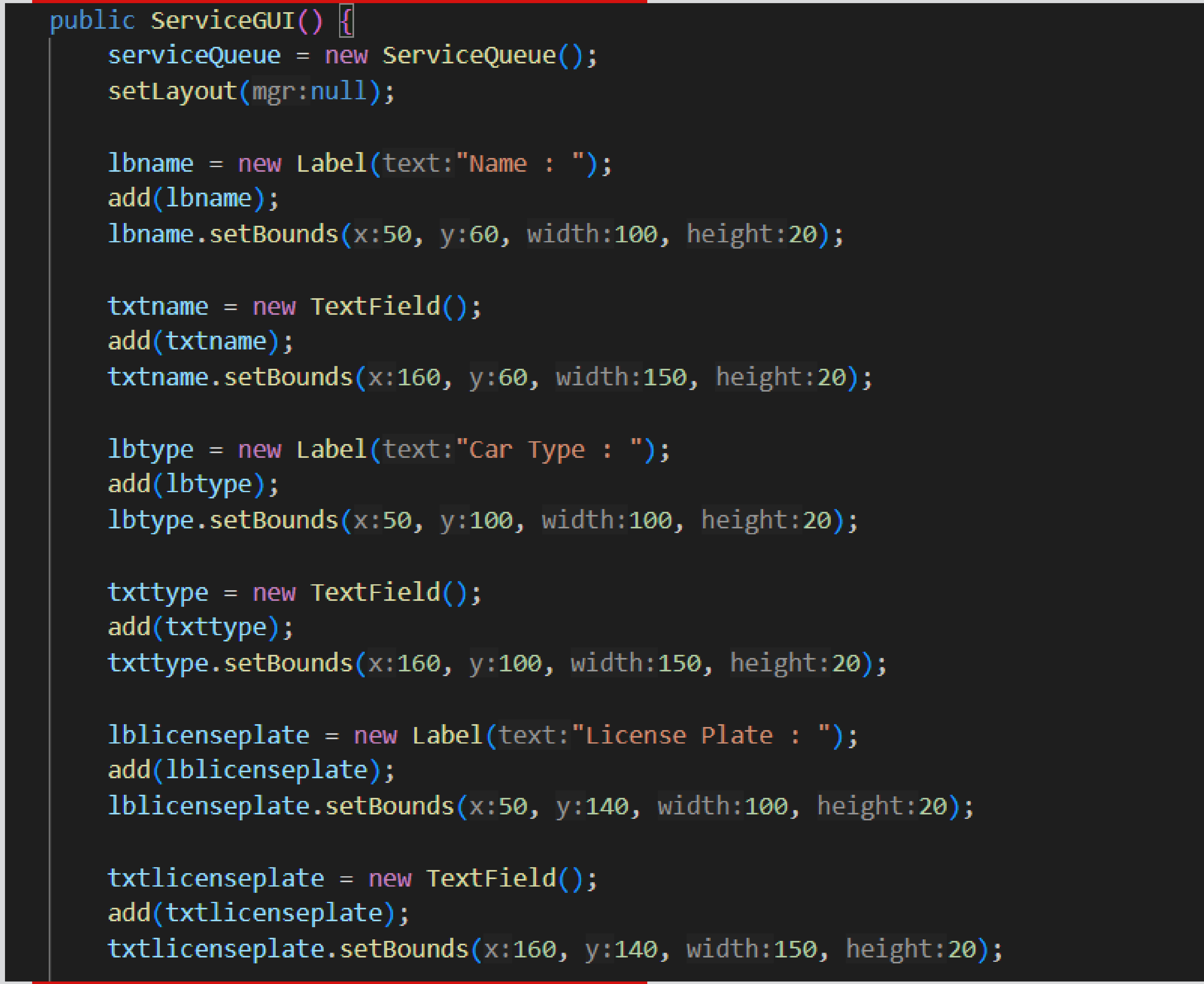
- **EmptyQueueException:**
  - Dilemparkan ketika mencoba mengeluarkan dari antrian kosong.
  - Memberikan pesan yang menunjukkan bahwa antrian kosong.
- **Java Exception Handling:**
  - Menggunakan **try-catch** untuk menangani eksepsi.
  - Memberikan umpan balik melalui pesan di GUI.



```
btnprocess = new Button(label:"Process");
add(btnprocess);
btnprocess.setBounds(x:160, y:200, width:100, height:30);
btnprocess.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            Customer customer = serviceQueue.dequeue();
            displayArea.setText("Processed: " + customer.toString());
        } catch (EmptyQueueException ex) {
            displayArea.setText(ex.getMessage());
        }
    }
});
```

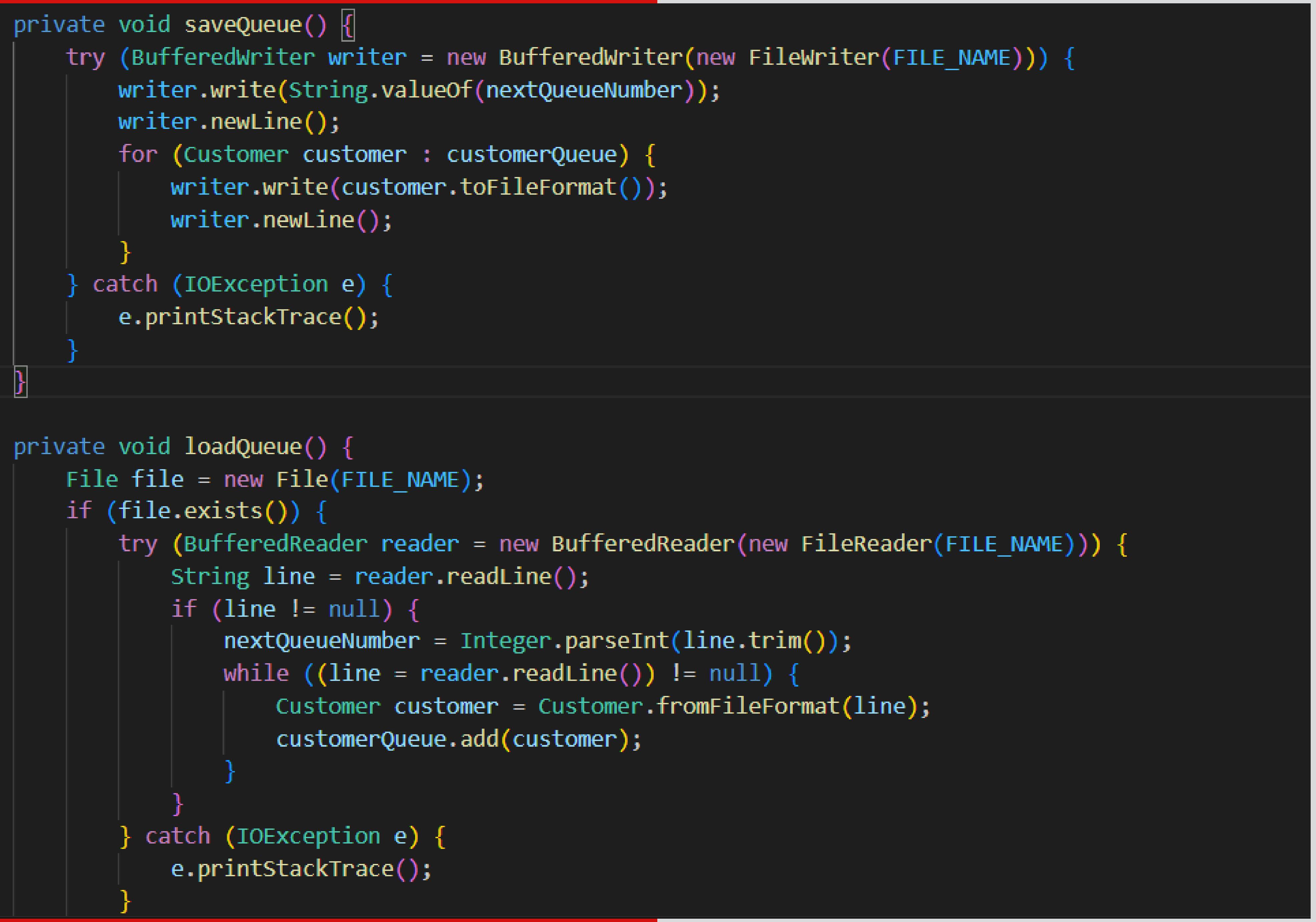
JAVA GUI

- **Label:**
  - **lbname**, **lbtype**, **lblicenseplate**: Menampilkan petunjuk untuk input pengguna.
- **TextFields:**
  - **txtname**, **txttype**, **txtlicenseplate**: Menangkap input pengguna.
- **Button:**
  - **btninsert**: Menambahkan pelanggan ke antrian.
  - **btnprocess**: Memproses pelanggan berikutnya.
  - **btndisplay**: Menampilkan antrian.
  - **btnreset**: Mereset antrian.
- **TextArea:**
  - **displayArea**: Menunjukkan pesan dan status antrian.



```
public ServiceGUI() {
    serviceQueue = new ServiceQueue();
    setLayout(mgr:null);

    lbname = new Label(text:"Name : ");
    add(lbname);
    lbname.setBounds(x:50, y:60, width:100, height:20);

    txtname = new TextField();
    add(txtname);
    txtname.setBounds(x:160, y:60, width:150, height:20);

    lbtype = new Label(text:"Car Type : ");
    add(lbtype);
    lbtype.setBounds(x:50, y:100, width:100, height:20);

    txttype = new TextField();
    add(txttype);
    txttype.setBounds(x:160, y:100, width:150, height:20);

    lblicenseplate = new Label(text:"License Plate : ");
    add(lblicenseplate);
    lblicenseplate.setBounds(x:50, y:140, width:100, height:20);

    txtlicenseplate = new TextField();
    add(txtlicenseplate);
    txtlicenseplate.setBounds(x:160, y:140, width:150, height:20);
```

JAVA OPERATION FILE



```java
private void saveQueue() {
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE_NAME))) {
        writer.write(String.valueOf(nextQueueNumber));
        writer.newLine();
        for (Customer customer : customerQueue) {
            writer.write(customer.toFileFormat());
            writer.newLine();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void loadQueue() {
    File file = new File(FILE_NAME);
    if (file.exists()) {
        try (BufferedReader reader = new BufferedReader(new FileReader(FILE_NAME))) {
            String line = reader.readLine();
            if (line != null) {
                nextQueueNumber = Integer.parseInt(line.trim());
                while ((line = reader.readLine()) != null) {
                    Customer customer = Customer.fromFileFormat(line);
                    customerQueue.add(customer);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Setiap data yang di input akan disimpan didalam file database.txt

THANK YOU