Apotek Yang
Mulia

LAPORAN PROYEK AKHIR COMP6364

– Object Oriented Programming KELAS

KELAS LA-20

Oleh:

2602292421 - Anasthasya Ivani Trishardiono

2602126896 - Kevin Husodo

2602067792 - Marvel Stefano

2602207350 - Zelos Mikhael Mahanaim

Semester Ganjil 2023/2024

MALANG

LEMBAR PERSETUJUAN PROYEK AKHIR

Apotek Yang
Mulia

COMP6364 – Object Oriented Programming
KELAS BF20-LAB

Semester Genap 2018/2019

Laporan akhir proyek ini adalah benar karya kami :

Kevin Husodo	Zelos Mikhael Mahanaim	Marvel Stefano	Anasthasya Ivani Trishardiono
			
2602126896	2602207350	2602067792	2602092421

Malang, 24 Desember 2023

(Dio Saputra Kudori)

D6910

BAB I

LATAR BELAKANG

Hampir semua industri, termasuk bidang kesehatan, sekarang menggunakan teknologi informasi. Hal ini bisa dikaitkan juga dengan SIF (Sistem Informasi Farmasi). Meskipun teknologi membantu proses bisnis menjadi lebih efisien dan akurat, apotik tetap merupakan komponen penting dalam pelayanan kesehatan. Program kasir apotek berbasis Java dibuat untuk mempermudah pencatatan dan manajemen transaksi.

Keamanan dengan Login biasanya terdapat pada halaman utama pada setiap program. Pada program ini, tentu saja kami juga menyediakan fitur login dengan tujuan untuk menjaga data pengguna tetap rahasia dan mencegah akses yang tidak sah. Setiap pengguna, termasuk petugas apotik, harus mengotentikasi identitasnya dan akan dikonfirmasi dalama pengiriman OTP sebelum dapat mengakses fungsi program. Keamanan juga tidak serta merta membantu mencegah pencurian data namun dapat juga menjaga data pelanggan aman.

Input Pembelian juga merupakan item penting. Fungsi utama program ini adalah untuk menyimpan catatan atas semua transaksi yang dilakukan oleh pelanggan dengan apotek. Petugas dapat dengan mudah memasukkan detail pembelian seperti nama obat, jumlah, dan harga melalui antarmuka yang ramah pengguna. Untuk meningkatkan kesadaran pengguna, program juga dapat memberikan rekomendasi atau pemberitahuan tentang dosis, interaksi obat, dan informasi penting lainnya.

Manajemen Inventori Otomatis (MIO) merupakan salah satu sistem pendukung yang dapat memudahkan pengelolaan terhadap stok obat. Program ini mencatat transaksi pembelian dan mengelola inventori dengan otomatis sehingga dapat mempermudah untuk mengetahui stok obat. Selain itu, program ini membantu memastikan ketersediaan stok obat dan mencegah kehabisan barang yang diperlukan karena stok secara otomatis berkurang ketika obat dibeli. Sistem ini juga dapat membantu untuk mengurangi human eror atau kelalaian saat menghitung stok obat.

Menu Print Nota merupakan cara untuk membuat struk sebagai bukti apa saja yang dibeli dan harus selalu ada dalam suatu transaksi. Setelah transaksi dicatat, program dapat mencetak nota pembelian yang berisi detail transaksi, seperti nama obat, jumlah, harga, dan total biaya. Nota ini berfungsi sebagai bukti pembelian untuk pelanggan dan juga sebagai catatan untuk apotik itu sendiri.

Lalu terdapat riwayat transaksi dan laporan, dimana anda dapat menyimpan riwayat transaksi dan menghasilkan laporan setiap hari, bulanan, atau tahunan dengan program ini yang berisikan tanggal, produk yang dibeli, dan jumlah. Selain itu, dengan adanya riwayat transaksi dapat membantu mengelola dan juga dapat memantau stok obat yang akan berhubungan dengan MIO secara otomatis. Laporan ini membantu manajer apotik memahami tren penjualan, mengelola inventori, dan membuat keputusan dengan data.

Program kasir apotek berbasis Java ini diharapkan dapat meningkatkan efisiensi operasi, memberikan layanan yang lebih baik kepada pelanggan, dan membantu manajemen membuat keputusan dengan data terbaru.

BAB II

LANDASAN TEORI

3.1 CONSTRUCTOR

3.1.1 Constructor adalah method khusus yang akan dieksekusi pada saat pembuatan objek (instance). Biasanya method ini digunakan untuk inisialisasi atau mempersiapkan data untuk objek.

3.2 OVERLOADING OPERATOR

3.2.1 overloading operator adalah salah satu fitur pada java yang memungkinkan kita mendirikan dua atau lebih funcion dengan identitas yang sama selama memiliki keunikan pada funcion parameter

3.3 INHERITANCE

3.3.1 **Inheritance** atau *Pewarisan/Penurunan* adalah konsep pemrograman dimana sebuah *class* dapat '*menurunkan*' ***property*** dan ***method*** yang dimilikinya kepada ***class*** lain.

3.4 ASSOCIATION

3.4.1 **Association** merupakan objek yang dapat berupa grup atau kelompok, namun objek ini tidak secara komplit tergantung satu dengan lainnya.

3.5 AGGREGATION

3.5.1 **Aggregation** merupakan salah satu konsep dasar dalam pemrograman berorientasi objek yang digunakan untuk menggambarkan hubungan antara dua kelas atau objek. dapat diartikan juga sebagai objek satu kelas menggunakan objek dari kelas lain sebagai objek bagian dari dirinya sendiri.

3.6 COMPOSITION

3.6.1 **Composition** merupakan suatu objek dibuat sebagai bagian dari objek lain dan tidak dapat ada tanpa objek tersebut. Dengan kata lain, jika objek utama dihapus, maka semua objek yang dibuat sebagai bagian dari objek utama juga akan dihapus.

3.7 POLYMORPHISM

3.7.1 **Polymorphism** merupakan salah satu konsep utama dalam pemrograman yang memungkinkan suatu objek dapat memiliki banyak bentuk. Polymorphism memungkinkan objek dari kelas yang sama dapat berperilaku berbeda tergantung pada konteks penggunaannya.

3.8 ABSTRACT CLASS

3.8.1 **Abstract Class** merupakan fitur yang digunakan untuk menyediakan blueprint atau kerangka dasar bagi kelas-kelas turunannya. abstraksi kelas tersebut memungkinkann untuk mendefinisikan metode tanpa memberikan implementasi lengkapnya. Kelas abstrak ini tidak dapat diinstansiasi secara langsung, tetapi dapat diwarisi oleh sub-class yang menyediakan implementasi lengkap untuk metode-metodenya.

BAB 3

DEFINISI PROGRAM

1. ItemDatabase Class:

Fungsi Utama: Membuat koneksi database dan membuatnya tersedia untuk class lain yang membutuhkan interaksi dengan database.

Metode:·

- getConnection(): Membuat koneksi database menggunakan JDBC dan mengembalikan objek Connection. Jika koneksi berhasil, objek Connection dikembalikan; jika tidak, null dikembalikan.

2. LoginForm Class:

Fungsi Utama: Menampilkan formulir login dengan dua kolom untuk memasukkan nama pengguna dan kata sandi. Memeriksa validitas kredensial sebelum memberikan akses ke halaman terbatas.

Metode:

- userInterface(): Mengatur tata letak dan komponen formulir login.
- actionPerformed(): Menangani tindakan tombol dan eksekusi program.
- main(): Fungsi utama untuk menjalankan aplikasi.

3. ItemForm Class:

Fungsi Utama: Menangani antarmuka pengguna terkait manajemen stok atau inventaris dan berkomunikasi dengan database melalui ItemService.

Metode:

- btnSaveActionPerformed(), btnPrintActionPerformed(), generatePDF(), checkPriceActionPerformed(), checkMoneyActionPerformed(), initTable(), loadData(), reset(): Menangani berbagai tindakan dan fungsi formulir.
- ItemForm(ItemService itemService): Constructor dengan parameter untuk injeksi dependensi.
- ItemForm(): Constructor tanpa parameter.

4. ItemModel Class:

Fungsi Utama: Merangkum atribut item dan menyediakan metode getter dan setter untuk mengakses dan memodifikasi variabel tersebut.

Metode:

- Getter dan setter untuk id, nama, kuantitas, dan harga item.

5. TransactionModel Class:

   Fungsi Utama: Mewakili model transaksi dalam sistem manajemen stok dengan fields dan metode terkait.

Metode:

BAB 4

- Getter dan setter untuk id transaksi, id item, nama pelanggan, tanggal transaksi, total harga, jumlah item, dan nama admin.

6. ItemService Interface:

Fungsi Utama: Mendefinisikan kontrak untuk class yang menyediakan layanan terkait barang.

Metode:

- insert(TransactionModel request): Menambahkan transaksi baru ke database.
- getPrice(int id): Mengambil harga suatu barang dari database berdasarkan ID.
- report(String name): Menghasilkan laporan transaksi untuk nama pelanggan tertentu.

7. ItemServiceImpl Class:

Fungsi Utama: Mengimplementasikan interface ItemService, mengelola dan melacak transaksi dan item di database.

Metode:

- Implementasi dari metode-metode yang didefinisikan dalam ItemService interface.
- Melibatkan operasi database seperti penyisipan, pengambilan harga, dan pembuatan laporan.

Program ini menggambarkan pola desain MVC (Model-View-Controller) di mana ItemForm dan LoginForm bertindak sebagai View, ItemModel dan TransactionModel sebagai Model, dan ItemService dan ItemServiceImpl sebagai Controller yang mengelola logika bisnis dan interaksi dengan database.

HASIL

ItemDatabase.java

```
package stock;

import com.mysql.cj.jdbc.MysqlDataSource;
import java.sql.Connection;
import java.sql.SQLException;

public class ItemDatabase {
   private static Connection connection;

   public static Connection getConnection(){
      //pengecekan koneksi database
      // Load the JDBC driver
      if (connection==null){
         try{
           String username = "root";
           String password = "";
           String url = "jdbc:mysql://localhost:3306/stock";

            MysqlDataSource source = new MysqlDataSource();
           source.setUser(username);
           source.setPassword(password);
           source.setURL(url);

           connection = source.getConnection();
         } catch (SQLException ex){
           System.out.println("Error koneksi database");
         }
      }

      return connection;
   }
}
```

Code diatas adalah class Java yang menggunakan JDBC (Java Database Connectivity) untuk menangani koneksi ke database MySQL. Fungsi getConnection() class return objek Connection, yang digunakan untuk menjalankan perintah SQL dan berkomunikasi dengan database.

Method: Function getConnection() bertugas membuat koneksi database. Ini menciptakan objek MysqlDataSource dan menetapkan pengguna koneksi, kata sandi, dan URL. Jika koneksi berhasil, objek Connection return. Jika terjadi kesalahan, ia akan mengeluarkan pesan kesalahan dan return null.

Class: ItemDatabase adalah class Java yang menyertakan function getConnection(). Ini bertugas mengelola koneksi database dan membuatnya tersedia untuk class lain yang memerlukan interaksi database.

Overloading Operator: Method getConnection() di kelas java.sql.Connection membebani method getConnection() di kelas java.sql.Connection karena menyediakan implementasi khusus untuk memperoleh koneksi ke database MySQL menggunakan JDBC.

Association: Class ItemDatabase terkait dengan class Connection karena menggunakan instance Connection untuk menjalankan perintah SQL dan berkomunikasi dengan database.

Aggregation: Class ItemDatabase mengagregasi class MysqlDataSource karena class tersebut memiliki instance MysqlDataSource sebagai bidang dan menggunakannya untuk menyambung ke database.

Composition: Class ItemDatabase memperluas class MysqlDataSource dengan menyertakan instance MysqlDataSource sebagai bidang dan menggunakannya untuk menyambung ke database.

run.java

```
package stock;

public class run {
   public static void main(String[] args) {
      new LoginForm().setVisible(true);
   }
}
```

Code diatas adalah class Java dengan function main. Function main membuat instance class LoginForm dan membuatnya visible. class LoginForm bertugas menampilkan form login dengan dua kolom untuk memasukkan nama pengguna dan kata sandi. Setelah mengirimkan form login, kode dasar formulir memeriksa apakah kredensialnya valid sebelum memberikan akses kepada pengguna ke halaman yang restricted.

Method: Fungsi utama bertugas menghasilkan sebuah instance dari class LoginForm dan membuatnya visible.

class: class run adalah v Java yang menyertakan function main. Ini bertugas me-run aplikasi dan menyajikan form login.

Association: Class run diasosiasikan dengan class LoginForm karena class ini menghasilkan sebuah instance dari LoginForm dan membuatnya visible.

ItemForm.java

```
package stock;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.Date;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
```

```
import javax.swing.table.DefaultTableModel;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDPage;
import org.apache.pdfbox.pdmodel.PDPageContentStream;
import org.apache.pdfbox.pdmodel.font.PDType0Font;
import stock.model.TransactionModel;
import stock.service.ItemService;
import stock.service.ItemServiceImpl;
import org.apache.pdfbox.pdmodel.font.PDType1Font;

public class ItemForm extends javax.swing.JFrame {
    private DefaultTableModel tableModel;
    private Connection connection;

    private ItemService itemService;

    public ItemForm(ItemService itemService) {
        this.itemService = itemService;
    }
    /**
     * Creates new form FormUtama
     */
    public ItemForm() {
        initComponents();
        initTable();
        loadData();
    }

    @SuppressWarnings("unchecked")
    private void initComponents() {

        jScrollPane2 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        custName = new javax.swing.JTextField();
        idObat = new javax.swing.JTextField();
        jumlah = new javax.swing.JTextField();
        money = new javax.swing.JTextField();
        _returns = new javax.swing.JTextField();
        jScrollPane3 = new javax.swing.JScrollPane();
        itemTable = new javax.swing.JTable();
        btnPrint = new javax.swing.JButton();
        btnSave = new javax.swing.JButton();
        btnBatal = new javax.swing.JButton();
        totalHarga = new javax.swing.JTextField();
        checkPrice = new javax.swing.JButton();
        checkMoney = new javax.swing.JButton();

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
```

```
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null}
        },
        new String [] {
            "Title 1", "Title 2", "Title 3", "Title 4"
        }
    ));
    jScrollPane2.setViewportView(jTable1);

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Aplikasi Stock");

    jLabel1.setText("Nama Customer");

    jLabel2.setText("ID Obat");

    jLabel3.setText("Jumlah");

    jLabel4.setText("Total Harga");

    jLabel5.setText("Uang");

    jLabel6.setText("Kembalian");

    itemTable.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null}
        },
        new String [] {
            "Title 1", "Title 2", "Title 3", "Title 4"
        }
    ));
    itemTable.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            itemTableMouseClicked(evt);
        }
    });
    jScrollPane3.setViewportView(itemTable);

    btnPrint.setText("Print");
    btnPrint.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnPrintActionPerformed(evt);
        }
    });

    btnSave.setText("Simpan");
    btnSave.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnSaveActionPerformed(evt);
        }
    });
```

```
        btnBatal.setText("Batal");
        btnBatal.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnBatalActionPerformed(evt);
            }
        });

        totalHarga.setEditable(false);
        totalHarga.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                totalHargaActionPerformed(evt);
            }
        });

        checkPrice.setText("Harga");
        checkPrice.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                checkPriceActionPerformed(evt);
            }
        });

        checkMoney.setText("Uang");
        checkMoney.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                checkMoneyActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
, false)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(btnSave)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(btnPrint)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 153,
Short.MAX_VALUE)
                                .addComponent(btnBatal))
                            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
```

```
                        .addComponent(jLabel1)
                        .addComponent(jLabel2)
                        .addComponent(jLabel3)
                        .addComponent(jLabel5)
                        .addComponent(jLabel4)
                        .addComponent(jLabel6))
                    .addGap(72, 72, 72)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
, false)
                        .addComponent(idObat,
javax.swing.GroupLayout.DEFAULT_SIZE, 215, Short.MAX_VALUE)
                        .addComponent(jumlah)
                        .addComponent(money)
                        .addComponent(_returns)
                        .addComponent(totalHarga)
                        .addComponent(custName))))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
, false)
                    .addComponent(checkPrice,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(checkMoney,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT_SIZE,
1044, Short.MAX_VALUE)))
    );

    layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {btnBatal, btnPrint, btnSave});

    layout.setVerticalGroup(
       layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
       .addGroup(layout.createSequentialGroup()
          .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
            .addComponent(jLabel1)
            .addComponent(custName, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
          .addGap(22, 22, 22)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
            .addComponent(jLabel2)
            .addComponent(idObat, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
            .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                .addComponent(jLabel3)
                .addComponent(jumlah, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                .addComponent(jLabel4)
                .addComponent(totalHarga, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(checkPrice))
            .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                .addComponent(jLabel5)
                .addComponent(money, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(checkMoney))
            .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                .addComponent(jLabel6)
                .addComponent(_returns, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
29, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                .addComponent(btnPrint)
                .addComponent(btnSave)
                .addComponent(btnBatal))
            .addGap(18, 18, 18)
            .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE,
191, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {
        TransactionModel data = new TransactionModel();
        data.setName(custName.getText());
        data.setAdmin("Admin");
        data.setDate_buy(LocalDate.now());
```

```
        data.setItem_id((int) Integer.parseInt(idObat.getText()));
        data.setQty((int) Integer.parseInt(jumlah.getText()));
        data.setTotal_price(Double.valueOf(totalHarga.getText()));

        ItemService itemService = new ItemServiceImpl();
        itemService.insert(data);

        loadData();
        reset();
    }

    private void btnPrintActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        ItemService itemService = new ItemServiceImpl();
        List<Object[]> resultList =  itemService.report(custName.getText());

        String customString = String.format("%-15s | %-25s | %-15s%n", "No Transaksi",
"Nama Obat", "Total Harga");

        for (int i = 0; i < resultList.size(); i++) {
            Object[] resultRow = resultList.get(i);

            int nomorTransaksi = (int) resultRow[3];
            String namaObat = (String) resultRow[2];
            double totalHarga = (double) resultRow[1];

            customString += String.format("%-15d | %-25s | %-15.2f%n", nomorTransaksi,
namaObat, totalHarga);
        }

        System.out.println(customString);
        String report = "Nota Pembelian \n" +
                    "\n Tanggal              : " + LocalDate.now() +
                    "\n Nama Kasir         : " + "ADMIN" +
                    "\n Nama Customer    : " + custName.getText() + "\n" +
                    customString;
        generatePDF("nota.pdf", report);

    }

    private void generatePDF(String filePath, String content) {
        try (PDDocument document = new PDDocument()) {
            PDPage page = new PDPage();
            document.addPage(page);
            try (InputStream fontStream =
getClass().getResourceAsStream("/stock/resources/Montserrat-Regular.ttf")) {
                PDType0Font font = PDType0Font.load(document, fontStream);

                try (PDPageContentStream contentStream = new
PDPageContentStream(document, page)) {
                    contentStream.setFont(font, 12);

                    float margin = 50;
                    float yStart = page.getMediaBox().getHeight() - margin;
                    float yPosition = yStart;
```

```
                // Set the line height
                float lineHeight =
font.getFontDescriptor().getFontBoundingBox().getHeight() / 1000 * 12;

                // Split the content into lines and add each line with newline
                String[] lines = content.split("\\n");
                for (String line : lines) {
                    if (yPosition - lineHeight < margin) {
                        // Create a new page if the content exceeds the current page
                        contentStream.endText();
                        contentStream.close();
                        page = new PDPage();
                        document.addPage(page);
                        contentStream.moveTo(margin, page.getMediaBox().getHeight() -
margin);
                        yPosition = page.getMediaBox().getHeight() - margin;
                    }

                    contentStream.beginText();
                    contentStream.newLineAtOffset(margin, yPosition -= lineHeight);
                    contentStream.showText(line);
                    contentStream.endText();
                }
            }
        }
        document.save(filePath);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void btnBatalActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    reset();
}

private void itemTableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
}

private void totalHargaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void checkPriceActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    ItemService itemService = new ItemServiceImpl();
    Double basePrice = itemService.getPrice((int) Integer.parseInt(idObat.getText()));
    int qty = (int) Integer.parseInt(jumlah.getText());
    Double totalPrice = basePrice * qty;
    totalHarga.setText(totalPrice.toString());
}

private void checkMoneyActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Double totalPrice = Double.valueOf(totalHarga.getText());
```

```
        Double _money = Double.valueOf(money.getText());
        Double ret = _money - totalPrice;
        _returns.setText(ret.toString());
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /*
         * Set the Nimbus look and feel
         */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
        /*
         * If Nimbus (introduced in Java SE 6) is not available, stay with the
         * default look and feel. For details see
         * http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ItemForm.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ItemForm.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ItemForm.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ItemForm.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /*
         * Create and display the form
         */
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new ItemForm().setVisible(true);
            }
        });
```

```
    }
    // Variables declaration - do not modify
    private javax.swing.JTextField _returns;
    private javax.swing.JButton btnBatal;
    private javax.swing.JButton btnPrint;
    private javax.swing.JButton btnSave;
    private javax.swing.JButton checkMoney;
    private javax.swing.JButton checkPrice;
    private javax.swing.JTextField custName;
    private javax.swing.JTextField idObat;
    private javax.swing.JTable itemTable;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JTable jTable1;
    private javax.swing.JTextField jumlah;
    private javax.swing.JTextField money;
    private javax.swing.JTextField totalHarga;
    // End of variables declaration

    private void initTable(){
        tableModel = new DefaultTableModel();
        itemTable.setModel(tableModel);
        tableModel.addColumn("ID");
        tableModel.addColumn("Item Name");
        tableModel.addColumn("Price");
        tableModel.addColumn("Stock");
    }

    private void loadData(){
        //menghapus seluruh data yg ada di tabel
        tableModel.getDataVector().removeAllElements();
        //pemberitahuan tabel telah kosong
        tableModel.fireTableDataChanged();

        try{
            connection = ItemDatabase.getConnection();
            String query = "SELECT * FROM items";

            Statement statement = (Statement) connection.createStatement();
            ResultSet resultSet = statement.executeQuery(query);

            //mengisi tabel dgn data hasil query
            while (resultSet.next()){
                Object[] item = new Object[4];
                item[0] = resultSet.getString("id");
                item[1] = resultSet.getString("name");
                item[2] = resultSet.getDouble("price");
                item[3] = resultSet.getInt("qty");
                tableModel.addRow(item);
            }
```

```
            resultSet.close();
            statement.close();
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }

    private void reset(){
        custName.setText("");
        idObat.setText("");
        money.setText("");
        jumlah.setText("");
        _returns.setText("");
        totalHarga.setText("");
    }
}
```

Class ItemForm adalah komponen program graphical user interface (GUI), terkait dengan manajemen stok atau inventaris, dan berkomunikasi dengan database melalui ItemService untuk menjalankan tindakan seperti entri data, pelaporan, dan pengambilan harga.

Methods: Class ini memiliki beberapa Methods, termasuk btnSaveActionPerformed, btnPrintActionPerformed, generatePDF, checkPriceActionPerformed, checkMoneyActionPerformed, initTable, loadData, dan reset. Method-Method ini menangani berbagai tindakan dan fungsi formulir, termasuk menyimpan data, mencetak, membuat PDF, dan memanipulasi data.

Constructor: Class ini memiliki dua Constructor satu dengan parameter (public ItemForm(ItemService itemService)) dan satu tanpa parameter. Constructor dengan parameter digunakan untuk injeksi dependensi, yang memungkinkan ItemService disediakan dari sumber eksternal.

Association: Class terhubung dengan antarmuka ItemService dan implementasinya, ItemServiceImpl, karena menggunakan keduanya untuk melakukan hal-hal seperti penyisipan data, pelaporan, dan pengambilan harga.

Aggregation: Karena kelas menggunakan DefaultTableModel dan objek Connection sebagai variabel anggota, kelas menggabungkan keduanya.

Polymorphism: Class menunjukkan Polymorphism dengan cara overloading Method seperti btnSaveActionPerformed dan generatePDF untuk menyediakan berbagai fungsi berdasarkan signature Method.

LoginForm.java

```
package stock;

import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
```

```
import javax.swing.JTextField;

public class LoginForm extends javax.swing.JFrame implements ActionListener {

//    public LoginForm() {
//        initComponents();
//    }

    private JTextField f_User;
    private JPasswordField f_Pass;
    private JButton login, batal;

    public LoginForm(){
        setSize(450, 200);
        setResizable(false);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        userInterface();
        setVisible(true);
    }

    private void userInterface(){
        JLabel judul = new JLabel("Form Login");
        judul.setHorizontalAlignment(JLabel.CENTER);
        getContentPane().add(judul, "North");

        JPanel panComp = new JPanel();
        panComp.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

        JLabel user = new JLabel("Username");
        user.setPreferredSize(new Dimension(110, 25));
        panComp.add(user);

        f_User = new JTextField();
        f_User.setPreferredSize(new Dimension(210, 25));
        panComp.add(f_User);

        JLabel pass = new JLabel("Password");
        pass.setPreferredSize(new Dimension(110, 25));
        panComp.add(pass);

        f_Pass = new JPasswordField();
        f_Pass.setPreferredSize(new Dimension(210, 25));
        panComp.add(f_Pass);

        getContentPane().add(panComp);

        JPanel panButton = new JPanel();
        panButton.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

        login = new JButton("Login");
        panButton.add(login);
        login.addActionListener(this);

        batal = new JButton("Cancel");
        panButton.add(batal);
```

```
        batal.addActionListener(this);

        getContentPane().add(panButton, "South");
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look
and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(LoginForm.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(LoginForm.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(LoginForm.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(LoginForm.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new LoginForm().setVisible(true);
            }
        });
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==login){
            String pass = String.valueOf(f_Pass.getPassword());
            if(f_User.getText().equals("admin") && pass.equals("admin")){
                JOptionPane.showMessageDialog(null, "Berhasil Login");
                this.setVisible(false);
                new ItemForm().setVisible(true);
            }else{
                JOptionPane.showMessageDialog(null, "Gagal Login");
            }
        }else if(e.getSource()==batal){
            System.exit(0);
        }
    }

    // Variables declaration - do not modify
    // End of variables declaration
}
```

Class LoginForm adalah komponen program graphical user interface (GUI), kemungkinan besar terhubung ke manajemen stok atau inventaris, dan berkomunikasi dengan database melalui ItemService untuk menyelesaikan tugas seperti entri data, pelaporan, dan pengambilan harga. Class ini juga menangani kejadian dan interaksi GUI untuk menyediakan program dengan user interface yang berfungsi.

Method: Ada berbagai Method di class, termasuk userInterface, actionPerformed, dan main. Method ini mengelola banyak aktivitas dan fitur formulir login, seperti konstruksi user interface, tindakan tombol, dan eksekusi program.

Constructor: Class ini memiliki Constructor public LoginForm() yang mengatur ukuran, tata letak, dan komponen formulir login.

Association: Class terhubung dengan berbagai komponen Swing seperti JLabel, JTextField, JPasswordField, dan JButton karena menggunakan komponen ini untuk menghasilkan user interface untuk formulir login.

Aggregation: Karena memiliki komponen-komponen ini sebagai variabel anggota, class menggabungkan beberapa komponen Swing seperti JLabel, JTextField, JPasswordField, dan JButton.

Polymorphism: Polymorphism ditunjukkan oleh class yang mengimplementasikan interface ActionListener dan memodifikasi metode actionPerformed untuk memberikan berbagai perilaku tergantung pada tombol yang diklik.

ItemModel.java

```
package stock.model;

public class ItemModel {
   private String id;

   private String name;

   private Integer qty;

   private Double price;

   public String getId() {
      return id;
   }

   public void setId(String id) {
      this.id = id;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public Integer getQty() {
      return qty;
   }

   public void setQty(Integer qty) {
      this.qty = qty;
   }

   public Double getPrice() {
      return price;
   }

   public void setPrice(Double price) {
      this.price = price;
   }

}
```

Class ItemModel adalah class Java sederhana yang merangkum atribut item dan menawarkan method getter dan setter untuk mengakses dan memodifikasi variabel-variabel ini. Dalam code tersebut, tidak terdapat inheritance, overloading operators, multithreading, association, aggregation, or polymorphism.

Class: class ItemModel adalah class Java yang mewakili item stok.
Method: class memiliki Method getter dan setter untuk id properti, nama, kuantitas, dan harga. Method ini digunakan untuk mendapatkan dan memperbarui nilai atribut class.

TransactionModel.java

```
package stock.model;

import java.time.LocalDate;

public class TransactionModel {
    private int id;

    private int item_id;

    private String name;

    private LocalDate date_buy;

    private Double total_price;

    private int qty;

    private String admin;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getItem_id() {
        return item_id;
    }

    public void setItem_id(int item_id) {
        this.item_id = item_id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public LocalDate getDate_buy() {
        return date_buy;
    }

    public void setDate_buy(LocalDate date_buy) {
        this.date_buy = date_buy;
    }
```

```
    public Double getTotal_price() {
        return total_price;
    }

    public void setTotal_price(Double total_price) {
        this.total_price = total_price;
    }

    public int getQty() {
        return qty;
    }

    public void setQty(int qty) {
        this.qty = qty;
    }

    public String getAdmin() {
        return admin;
    }

    public void setAdmin(String admin) {
        this.admin = admin;
    }
}
```

code diatas adalah class Java yang disebut TransactionModel. Class ini mewakili model transaksi dalam sistem manajemen stok. Ini berisi beberapa method, constructor, dan fields.

getId(), setId(int id): Method dan fields ini digunakan untuk getter dan setter ID transaksi.

getItem_id(), setItem_id(int item_id): Method dan fields ini digunakan untuk getter dan setter ID item yang terkait dengan transaksi.

getName(), setName(String name): Method dan fields ini digunakan untuk getter dan setter nama pelanggan yang terlibat dalam transaksi.

getDate_buy(), setDate_buy(LocalDate date_buy): Method dan fields ini digunakan untuk getter dan setter tanggal transaksi dilakukan.

getTotal_price(), setTotal_price(Double total_price): Method dan fields ini digunakan untuk getter dan setter total harga transaksi.

getQty(), setQty(int qty): Method dan fields ini digunakan untuk getter dan setter jumlah item dalam transaksi.

getAdmin(), setAdmin(String admin): Method dan fields ini digunakan untuk getter dan setter nama admin yang terkait dengan transaksi.

ItemService.Java

```
package stock.service;

import java.util.List;
import stock.model.TransactionModel;

public interface ItemService {

   public void insert(TransactionModel request);

   public Double getPrice(int id);

   public List<Object[]> report(String name);
}
```

Code diatas adalah sebuah interface yang disebut ItemService dalam paket stock.service. Interface ini berisi tiga signatures method terkait pengelolaan barang dan transaksi.

Interface: ItemService adalah interface Java yang mendefinisikan kontrak untuk class yang menyediakan layanan terkait barang.

Method: interface ini berisi tiga signatures method: insert, getPrice, dan report. Method ini dimaksudkan untuk diimplementasikan oleh class yang menyediakan fungsionalitas sebenarnya untuk menyisipkan transaksi, mengambil harga, dan menghasilkan laporan.

ItemServiceImpl.java

```
package stock.service;

   import java.sql.Connection;
   import java.sql.PreparedStatement;
   import java.sql.ResultSet;
   import java.sql.SQLException;
   import java.util.ArrayList;
   import java.util.List;
   import stock.ItemDatabase;
   import stock.model.TransactionModel;

   public class ItemServiceImpl implements ItemService{

      private Connection connection;

      @Override
      public void insert(TransactionModel request){

         //masukkan data ke database
         connection = ItemDatabase.getConnection();
         String query = "INSERT INTO transaction"
               + "(name, item_id, total_price, admin, qty, date_buy)"
               + "VALUES (?,?,?,?,?,?)";
         try {
            PreparedStatement statement= (PreparedStatement)
connection.prepareStatement(query);

            statement.setString(1, request.getName());
```

```
            statement.setInt(2, request.getItem_id());
            statement.setDouble(3, request.getTotal_price());
            statement.setString(4, request.getAdmin());
            statement.setInt(5, request.getQty());
            statement.setString(6, request.getDate_buy().toString());

            statement.executeUpdate();
        } catch (SQLException ex){
            System.out.println("Error: " + ex.getMessage());
        }
    }

    @Override
    public Double getPrice(int id) {
        connection = ItemDatabase.getConnection();
        String query = "select price from items where id = ? limit 1";
        Double result = 0.00;
        try {
            PreparedStatement statement = connection.prepareStatement(query);
            statement.setInt(1, id);

            ResultSet resultSet = statement.executeQuery();

            if (resultSet.next()) {
                result = resultSet.getDouble("price");
            } else {
                return 0.00;
            }
        } catch (SQLException ex) {
            System.out.println("Error: " + ex.getMessage());
        }

        return result;
    }

    @Override
    public List<Object[]> report(String request) {
        connection = ItemDatabase.getConnection();
        String query = "select a.name as nama,\n" +
                       "       a.total_price as total_harga,\n" +
                       "       b.name as nama_obat,\n" +
                       "       a.id as nomor_transaksi\n" +
                       "from transaction a\n" +
                       "         join items b on a.item_id = b.id\n" +
                       "where a.name = ?";

        List<Object[]> resultList = new ArrayList<Object[]>();

        try (PreparedStatement statement = connection.prepareStatement(query)) {
            statement.setString(1, request);

            try (ResultSet resultSet = statement.executeQuery()) {
                while (resultSet.next()) {
                    String nama = resultSet.getString("nama");
                    double totalHarga = resultSet.getDouble("total_harga");
                    String namaObat = resultSet.getString("nama_obat");
```

```
                int nomorTransaksi = resultSet.getInt("nomor_transaksi");

                Object[] resultRow = {nama, totalHarga, namaObat, nomorTransaksi};
                resultList.add(resultRow);
            }
          }
        } catch (SQLException ex) {
          System.out.println("Error: " + ex.getMessage());
        }

        return resultList;
    }

}
```

Code di atas menunjukkan cara menggunakan JDBC (Java Database Connectivity) untuk berkomunikasi dengan database. Class ItemServiceImpl bertugas memasukkan transaksi baru ke dalam database, mendapatkan harga item, dan memberikan laporan transaksi untuk klien tertentu. Interface ItemService menentukan metode mana yang harus diimplementasikan oleh Class ItemServiceImpl.

Inheritance: Class ItemServiceImpl adalah implementasi interface ItemService. Dibutuhkan method interface dan menambahkan implementasinya sendiri.

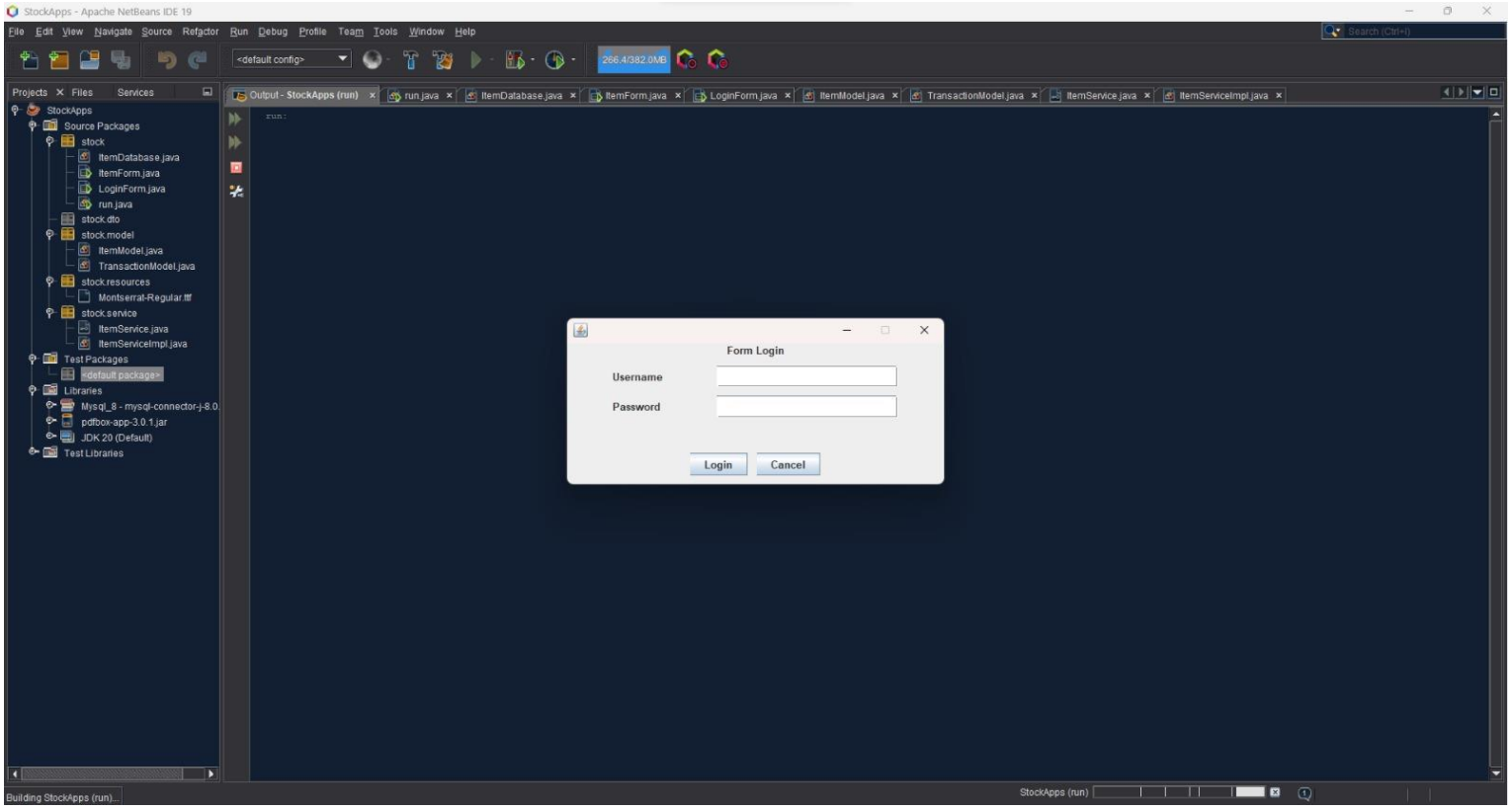
Method:

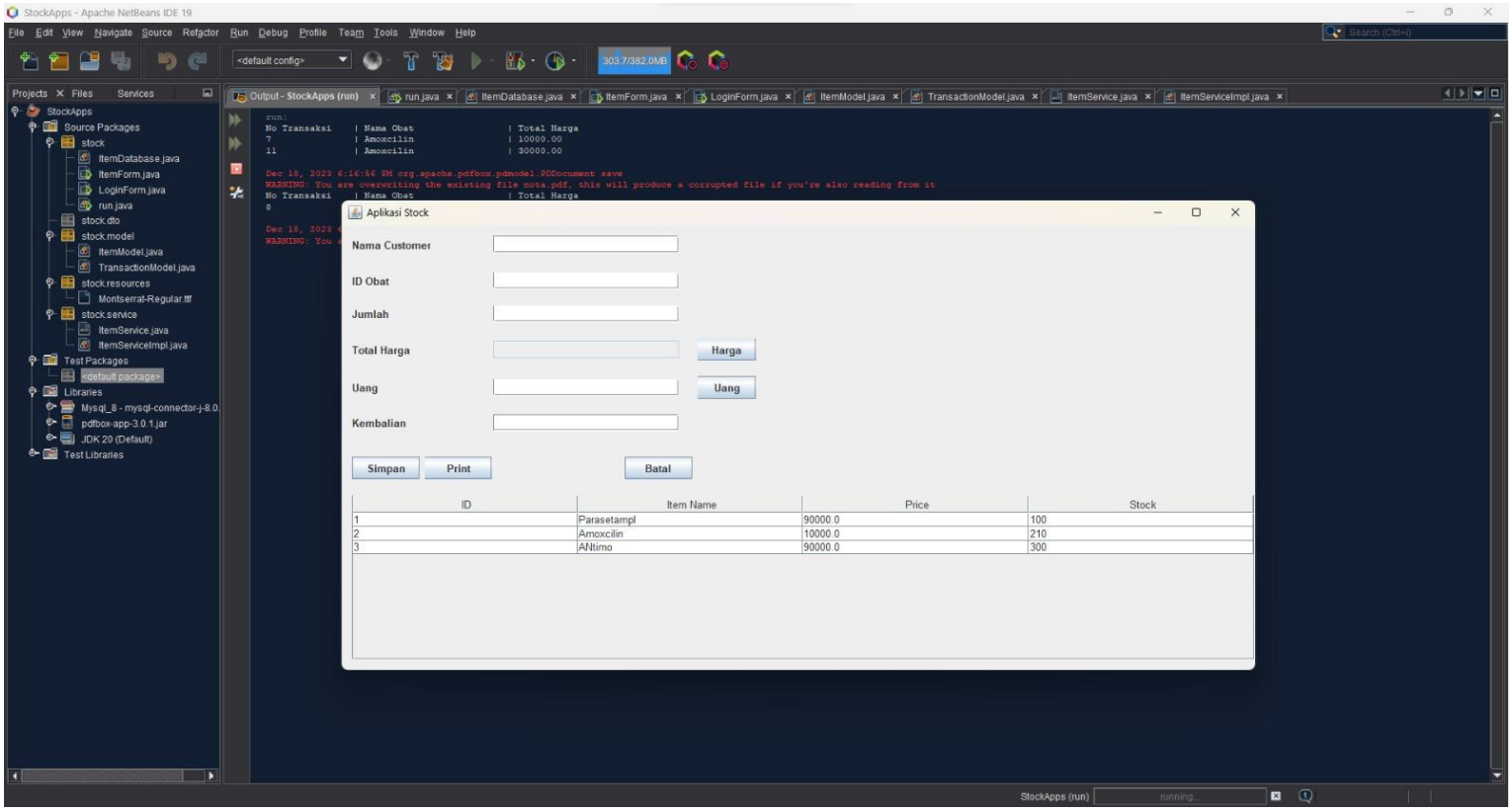
- insert(TransactionModel Request): Menambahkan transaksi baru ke database.
- getPrice(int id): Method ini mengambil dari database harga suatu barang dengan ID yang ditentukan.
- report (String name): function ini mengembalikan kumpulan objek yang menyediakan laporan transaksi untuk nama pelanggan yang ditentukan.

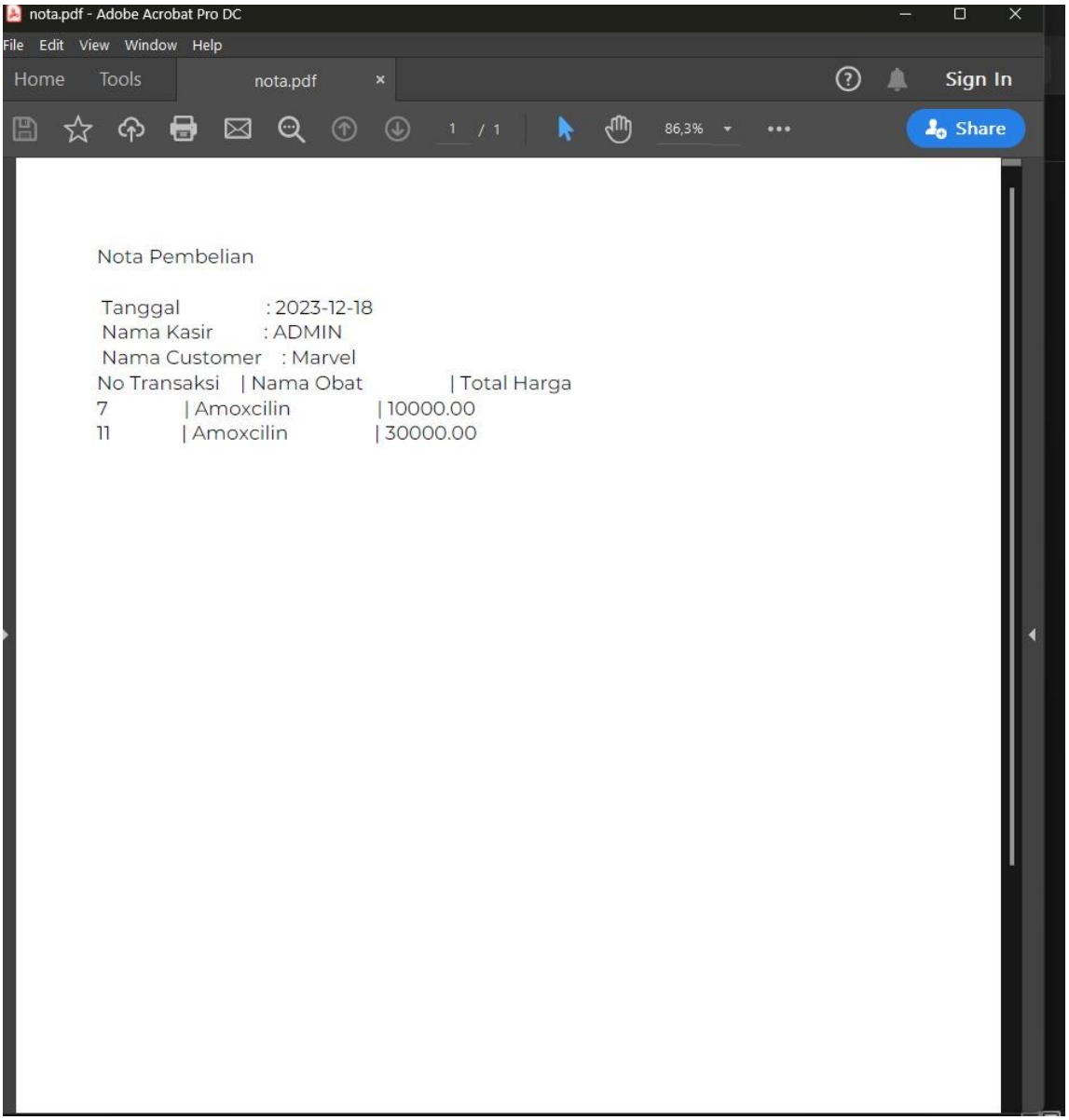
Class:

- ItemServiceImpl: Class ini merupakan implementasi interface ItemService. Class Ini mengelola dan melacak transaksi dan item di database.

SCREENSHOT APLIKASI






nota.pdf - Adobe Acrobat Pro DC
File Edit View Window Help
Home Tools
nota.pdf
Sign In
1 / 1
86,3%
Share
Nota Pembelian
Tanggal : 2023-12-18
Nama Kasir : ADMIN
Nama Customer : Marvel
No Transaksi | Nama Obat | Total Harga
7 | Amoxcilin | 10000.00
11 | Amoxcilin | 30000.00