Nama : Ni Kadek Juni Antari

Nim : 2201010172

Quis 1 :

1. Usulkan sebuah ide project aplikasi (aplikasi bisa menggunakan database dan bisa tidak menggunakan database) penjelasan tentang project yang di desain dan beri penjelasan kenapa menggunakan teori tersebut
2. Usulan project dan penjelasan boleh di tulis tangan atau menggunakan aplikasi pengolah kata seperti Ms. Word, Notepad, ataupun yang lainnya di buat dalam format PDF dilengkapi dengan NIM dan Nama serta di upload ke repositories github (petunjuk quis point c)
3. Konfirmasikan quis anda pada bagian konfirmasi quis 1 di pertemuan 16

Jawaban :

1.Pendahuluan

Aplikasi Etalase Manajemen Barang Toko dipilih sebagai studi kasus dalam pembahasan konsep Object-Oriented Programming (OOP) karena aplikasi ini merupakan contoh yang representatif untuk menunjukkan bagaimana prinsip-prinsip OOP dapat diterapkan dalam pengembangan perangkat lunak. Berikut adalah beberapa alasan utama mengapa aplikasi ini dipilih:

1. Kebutuhan Nyata di Dunia Bisnis: Manajemen barang merupakan salah satu aspek penting dalam operasional toko. Setiap toko perlu mengelola inventaris mereka dengan baik untuk memastikan ketersediaan barang, menghindari kelebihan stok atau kekurangan stok, serta untuk menganalisis data penjualan. Aplikasi Etalase Manajemen Barang Toko

memungkinkan pemilik toko untuk melakukan manajemen inventaris secara efisien.

2. Implementasi Prinsip OOP: Aplikasi ini memberikan kesempatan untuk menerapkan berbagai konsep OOP seperti encapsulation, inheritance, polymorphism, dan access modifiers. Dengan menggunakan Java sebagai bahasa pemrograman, kita dapat dengan mudah mendemonstrasikan bagaimana setiap konsep OOP dapat digunakan untuk membangun aplikasi yang modular, mudah dipelihara, dan scalable.
3. Struktur Data yang Kompleks: Manajemen barang toko melibatkan berbagai entitas seperti produk, pelanggan, dan transaksi. Setiap entitas ini memiliki atribut dan perilaku yang berbeda, yang dapat diorganisasikan dalam kelas-kelas yang saling berinteraksi. Ini memberikan contoh yang baik tentang bagaimana hubungan antara kelas dapat dikelola menggunakan prinsip OOP.
4. Kemudahan Ekspansi: Aplikasi Etalase Manajemen Barang Toko dapat dengan mudah dikembangkan lebih lanjut dengan menambahkan fitur-fitur baru seperti manajemen pelanggan, pelacakan transaksi, laporan penjualan, dan integrasi dengan sistem pembayaran. Hal ini menunjukkan bagaimana desain berbasis OOP dapat membuat aplikasi lebih fleksibel dan mudah diperluas.
5. Penggunaan Database: Aplikasi ini juga mencakup penggunaan database untuk menyimpan dan mengambil data, yang merupakan bagian penting dari banyak aplikasi bisnis. Dengan mengintegrasikan database dalam aplikasi, kita dapat menunjukkan bagaimana konsep OOP dapat diterapkan dalam konteks pengelolaan data yang persisten.

Dengan menggunakan Aplikasi Etalase Manajemen Barang Toko sebagai studi kasus, kita dapat mengeksplorasi berbagai aspek pengembangan perangkat lunak

berbasis OOP dan menunjukkan bagaimana prinsip-prinsip ini dapat diterapkan dalam dunia nyata untuk menciptakan solusi perangkat lunak yang efektif dan efisien.

2.Konsep OOP

2.1 Access Modifiers

Access Modifiers menentukan visibilitas dan aksesibilitas anggota kelas (atribut dan metode). Berikut adalah contoh penggunaan Access Modifiers dalam kode:

```
public class Product {
    private int id; // hanya bisa diakses dalam kelas ini
    protected String name; // bisa diakses dalam paket yang sama dan subclass
    public double price; // bisa diakses dari mana saja

    public int getId() {
        return id;
    }

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

2.2 Inheritance

Inheritance memungkinkan satu kelas untuk mewarisi atribut dan metode dari kelas lain. Contoh:

```
public class Product {
    private int id;
    private String name;
    private double price;

    // Constructor, getters, and setters
}

public class ElectronicProduct extends Product {
    private int warrantyPeriod;

    public ElectronicProduct(int id, String name, double price, int warrantyPeriod)
{
        super(id, name, price);
        this.warrantyPeriod = warrantyPeriod;
    }

    public int getWarrantyPeriod() {
        return warrantyPeriod;
    }
```

```
    public void setWarrantyPeriod(int warrantyPeriod) {
        this.warrantyPeriod = warrantyPeriod;
    }
}
```

2.3 Polymorphism

Polymorphism memungkinkan metode yang sama untuk digunakan pada objek kelas yang berbeda. Contoh:

```
public class Product {
    public void displayInfo() {
        System.out.println("This is a product.");
    }
}

public class ElectronicProduct extends Product {
    @Override
    public void displayInfo() {
        System.out.println("This is an electronic product.");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Product p1 = new Product();
        Product p2 = new ElectronicProduct();

        p1.displayInfo(); // Output: This is a product.
        p2.displayInfo(); // Output: This is an electronic product.
    }
}
```

2.4 Encapsulation

Encapsulation adalah konsep OOP yang menyembunyikan detail implementasi dari pengguna. Contoh:

```
public class Product {
    private int id;
    private String name;
    private double price;

    // Constructor
    public Product(int id, String name, double price) {
        this.id = id;
        this.name = name;
```

```
    this.price = price;
}

// Getters and setters
public int getId() {
    return id;
}

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

public String getName() {
    return name;
}

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

public double getPrice() {
    return price;
```

```
    }

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

3. Implementasi dalam Aplikasi Etalase Manajemen Barang Toko

Berikut adalah implementasi konsep-konsep OOP dalam kode program aplikasi yang sudah ada:

```
public class ProductFormJuni extends javax.swing.JFrame {

    // Connection details are encapsulated
    private Connection con;
    private PreparedStatement pst;
    private ResultSet rs;

    public ProductFormJuni() {
        initComponents();
        Connect();
        LoadProductNo();
        Fetch();
```

```
    }

    private void Connect() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost/tugas_netbeans_java_barang","root","");
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(ProductFormJuni.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void LoadProductNo() {
        try {
            pst = con.prepareStatement("SELECT id FROM product_tbl");
            rs = pst.executeQuery();
            txtpid.removeAllItems();
            while(rs.next()) {
                txtpid.addItem(rs.getString(1));
            }
```

```
        } catch (SQLException ex) {

Logger.getLogger(ProductFormJuni.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void Fetch() {
        try {
            pst = con.prepareStatement("SELECT * FROM product_tbl");
            rs = pst.executeQuery();
            ResultSetMetaData rss = rs.getMetaData();
            int q = rss.getColumnCount();

            DefaultTableModel df = (DefaultTableModel)jTable1.getModel();
            df.setRowCount(0);
            while(rs.next()) {
                Vector<String> v2 = new Vector<>();
                for(int a = 1; a <= q; a++) {
                    v2.add(rs.getString("id"));
                    v2.add(rs.getString("pname"));
                    v2.add(rs.getString("price"));
```

```
                v2.add(rs.getString("qty"));
            }
            df.addRow(v2);
        }
    } catch (SQLException ex) {
Logger.getLogger(ProductFormJuni.class.getName()).log(Level.SEVERE, null, ex);
    }
}

// Encapsulated methods for button actions (Add, Update, Delete, Search, Export)
private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String pname = txtPname.getText();
        String price = txtPrice.getText();
        String qty = txtQty.getText();

        pst = con.prepareStatement("INSERT INTO product_tbl (pname,price,qty) VALUES(?,?,?)");
        pst.setString(1, pname);
        pst.setString(2, price);
```

```
            pst.setString(3, qty);


            int k = pst.executeUpdate();
            if(k == 1) {
                JOptionPane.showMessageDialog(this, "Record Added Successfully");
                txtPname.setText("");
                txtPrice.setText("");
                txtQty.setText("");
                Fetch();
                LoadProductNo();
            } else {
                JOptionPane.showMessageDialog(this, "Record Failed to Save");
            }
        } catch (SQLException ex) {

Logger.getLogger(ProductFormJuni.class.getName()).log(Level.SEVERE, null, ex);
        }
    }


    // Similarly, implement other action methods (Update, Delete, Search, Export)
```

```
// ...

// Main method to run the application
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(() -> {
        new ProductFormJuni().setVisible(true);
    });
}

// Variable declaration - do not modify
private javax.swing.JButton btnAdd;
private javax.swing.JButton btnDelete;
private javax.swing.JButton btnExport;
private javax.swing.JButton btnSearch;
private javax.swing.JButton btnUpdate;
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.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
```

```
    private javax.swing.JTable jTable1;
    private javax.swing.JTextField txtPname;
    private javax.swing.JTextField txtPrice;
    private javax.swing.JTextField txtQty;
    private javax.swing.JComboBox<String> txtpid;
    // End of variables declaration
}
```

4. Kesimpulan

Dengan menggunakan konsep OOP, kita dapat membangun aplikasi yang lebih terstruktur, mudah dikelola, dan skalabel. Implementasi Access Modifiers, Inheritance, Polymorphism, dan Encapsulation dalam proyek Aplikasi Etalase Manajemen Barang Toko memberikan contoh nyata bagaimana prinsip-prinsip OOP diterapkan dalam pengembangan perangkat lunak.

Quis 2

Quis 2
Implementasi Desain aplikasi dari Quis 1
1. Implementasikan desain aplikasi anda dengan menggunakan pemrograman JAVA dan simpan project di github yang telah di buat pada quis 1 dan konfirmasikan URL Repositores di bagian konfirmasi project UAS pada pertemuan 16

: https://github.com/JuniAntari14/OOP-UAS-2201010172/