Kelas Barang

❖ ***Fungsi Kelas***

- Kelas ini merepresentasikan entitas barang dengan atribut nama, kategori, harga satuan, dan stok.
- Menggunakan enkapsulasi dengan mendefinisikan atribut sebagai private agar data tidak langsung diakses atau dimodifikasi dari luar kelas.

❖ ***Atribut***

- nama, kategori, hargaSatuan, stok : Mewakili properti barang yang dapat diakses melalui getter dan dimodifikasi melalui setter.

❖ ***Metode***

- Constructor : Untuk menginisialisasi objek dengan nilai awal untuk setiap atribut.
- Getter dan Setter : Menyediakan akses ke atribut private sesuai prinsip enkapsulasi.
- Metode tambahan : Anda dapat menambahkan metode seperti tambahStok() dan kurangiStok() untuk memodifikasi stok barang sesuai kebutuhan.

Kelas Gudang

❖ ***Fungsi Kelas***

- Berfungsi sebagai pengelola data barang dalam bentuk daftar (ArrayList).
- Menerapkan enkapsulasi dengan atribut daftarBarang yang private agar hanya bisa diakses melalui metode yang disediakan.

❖ ***Atribut***

- daftarBarang : List untuk menyimpan koleksi barang yang ada di gudang.

❖ ***Metode***

- tambahBarang(Barang barang) : Untuk menambahkan objek barang baru ke dalam daftar.
- kurangiBarang(String nama, int jumlah) : Untuk mengurangi stok barang berdasarkan nama.
- updateBarang(String nama, String kategori, double harga) : Memperbarui informasi barang.
- getDaftarBarang() : Memberikan akses daftar barang (read-only).
- ArrayList<Barang> getDaftarBarang(): Mengembalikan daftar barang yang ada di gudang.

Kelas MainGudangGUI

❖ ***Atribut***

Kelas ini memiliki atribut berikut:

- gudang : Objek dari kelas Gudang yang menyimpan daftar barang.
- tableModel : Model data tabel menggunakan DefaultTableModel untuk mengatur data yang akan ditampilkan di tabel.
- barangTable : Komponen tabel (JTable) yang menampilkan data barang dalam gudang.

Potongan kode :

```
private Gudang gudang;
private JTable barangTable;
private DefaultTableModel tableModel;
```

❖ ***Komponen GUI Utama***

Kelas ini menggunakan panel-panel berbeda untuk fungsionalitas tertentu.

a. Panel Input Barang :

Berfungsi untuk menambahkan barang baru ke gudang.
Input berupa nama barang, kategori, harga, dan stok.
Tombol "Tambah Barang" untuk menambahkan data ke tabel.
Potongan Kode :

```
private JPanel createInputPanel() {
    JPanel inputPanel = new JPanel(new GridLayout(rows:5, cols:2));
    inputPanel.setBorder(BorderFactory.createTitledBorder(title:"In

    inputPanel.add(new JLabel(text:"Nama Barang:"));
    namaField = new JTextField();
    inputPanel.add(namaField);

    inputPanel.add(new JLabel(text:"Kategori:"));
    kategoriComboBox = new JComboBox<>(new String[]{
        "Makanan/Minuman", "Obat", "Alat Rumah Tangga", "Alat Elekt
    });
    inputPanel.add(kategoriComboBox);
```

b. Panel Tabel Barang

Menampilkan daftar barang dalam bentuk tabel dengan kolom aksi (Tambah Stok, Kurangi Stok, Update).
Potongan Kode :

```
private JPanel createTablePanel() {
    JPanel tablePanel = new JPanel(new BorderLayout());

    // Create table model
    String[] columnNames = {"Nama", "Kategori", "Harga Satuan", "Stok", "Tambah Stok", "Kurangi Stok",
    "Update"};
    tableModel = new DefaultTableModel(columnNames, 0) {
        @Override
        public boolean isCellEditable(int row, int column) {
            // Only make button columns editable
            return column >= 4;
        }
    };
    barangTable = new JTable(tableModel);
```

c. Panel Pencarian Barang

Memungkinkan pencarian barang berdasarkan nama.
Tombol "Cari" untuk memfilter tabel berdasarkan input.
Potongan Kode :

```
private JPanel createSearchPanel() {
    JPanel searchPanel = new JPanel(new GridLayout(rows:2, cols:1));
    searchPanel.setBorder(BorderFactory.createTitledBorder(title:"Search Bar"));

    searchField = new JTextField();
    searchField.addActionListener(e -> searchBarang());
    searchPanel.add(searchField);

    JButton cancelButton = new JButton(text:"Cancel");
    cancelButton.addActionListener(e -> cancelSearch());
    searchPanel.add(cancelButton);

    return searchPanel;
}
```

```
private void searchBarang() {
    String searchQuery = searchField.getText().toLowerCase();
    tableModel.setRowCount(rowCount:0); // Clear the table

    for (Barang barang : gudang.getDaftarBarang()) {
        if (barang.getNama().toLowerCase().contains(searchQuery)) {
            tableModel.addRow(new Object[]{
                    barang.getNama(),
                    barang.getKategori(),
                    barang.getHargaSatuan(),
                    barang.getStok(),
                    "Tambah Stok",
                    "Kurangi Stok",
                    "Update"
            });
        }
    }
}
```

❖ ***Fitur Utama***

a. Tambah Barang

Data barang baru ditambahkan ke daftar gudang dan tabel diperbarui.

Potongan Kode :

```
private void tambahBarang() {
    try {
        String nama = namaField.getText();
        String kategori = (String) kategoriComboBox.getSelectedItem();
        double harga = Double.parseDouble(hargaField.getText());
        int stok = Integer.parseInt(stokField.getText());

        Barang barang = new Barang(nama, kategori, harga, stok);
        gudang.tambahBarang(barang);

        // Update table
        tableModel.addRow(new Object[]{nama, kategori, harga, stok, "Tambah Stok", "Kura

        // Clear input fields
        clearInputFields();
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(this,
```

b. Update Barang

Menampilkan dialog untuk mengubah kategori atau harga barang dari tabel.

Potongan Kode :

```
private void updateBarang(int row) {
    String nama = tableModel.getValueAt(row, column:0).toString();

    JPanel updatePanel = new JPanel(new GridLayout(rows:3, cols:2));
    JTextField hargaField = new JTextField(tableModel.getValueAt(row, column:2).toString());
    JComboBox<String> kategoriComboBox = new JComboBox<>(new String[]{
            "Makanan/Minuman", "Obat", "Alat Rumah Tangga", "Alat Elektronik", "Pakaian"
    });
    kategoriComboBox.setSelectedItem(tableModel.getValueAt(row, column:1).toString());

    updatePanel.add(new JLabel(text:"Kategori:"));
    updatePanel.add(kategoriComboBox);
    updatePanel.add(new JLabel(text:"Harga Satuan:"));
    updatePanel.add(hargaField);

    int result = JOptionPane.showConfirmDialog(parentComponent:null, updatePanel, title:"Upd
    JOptionPane.OK_CANCEL_OPTION);
```

c. Tambah/Kurangi Stok

Tombol "Tambah Stok" dan "Kurangi Stok" langsung mengubah stok barang dalam tabel.

```
private void tambahStok(int row) {
    String jumlahStr = JOptionPane.showInputDialog(message:"Masukkan jumlah stok yang akan ditambahkan:");
    try {
        int jumlah = Integer.parseInt(jumlahStr);
        String nama = tableModel.getValueAt(row, column:0).toString();
        gudang.kurangiBarang(nama, -jumlah); // Kurangi negatif = tambah stok
        refreshTable();
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(button, message:"Masukkan angka yang valid!");
    }
}

private void kurangiStok(int row) {
    String jumlahStr = JOptionPane.showInputDialog(message:"Masukkan jumlah stok yang akan dikurangi:");
    try {
        int jumlah = Integer.parseInt(jumlahStr);
        String nama = tableModel.getValueAt(row, column:0).toString();
        gudang.kurangiBarang(nama, jumlah);
        refreshTable();
```

d. Refresh Tabel

Metode ini bertanggung jawab untuk memperbarui atau merefresh tampilan tabel di GUI setelah ada perubahan data, seperti menambah barang, mengubah stok, atau melakukan update barang.

Potongan Kode :

```
private void refreshTable() {
    // Clear existing rows
    tableModel.setRowCount(rowCount:0);

    // Refill tabel dengan data terkini
    for (Barang barang : gudang.getDaftarBarang()) {
        tableModel.addRow(new Object[]{
                barang.getNama(),
                barang.getKategori(),
                barang.getHargaSatuan(),
                barang.getStok(),
                "Tambah Stok",
                "Kurangi Stok",
                "Update"
        });
    }
}
```

❖ ***Subclass***

a. ButtonRenderer

Subclass dari JButton dan TableCellRenderer untuk menampilkan tombol di kolom tabel.

Potongan Kode :

```
private class ButtonRenderer extends JButton implements TableCellRenderer {
    public ButtonRenderer() {
        setOpaque(isOpaque:true);
    }

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value,
                                                   boolean isSelected, boolean hasFocus,
                                                   int row, int column) {
        setText(value.toString());
        return this;
    }
}
```

b. ButtonEditor

Subclass dari DefaultCellEditor untuk menangani aksi tombol dalam tabel.

```
private class ButtonEditor extends DefaultCellEditor {
    private JButton button;
    private String label;
    private int selectedRow;

    public ButtonEditor(JButton checkBox) {
        super(new JCheckBox());
        button = new JButton();
        button.setOpaque(isOpaque:true);
        button.addActionListener(e -> fireEditingStopped());
    }
```

Elemen OOP dalam Program

1. ***Enkapsulasi***

Enkapsulasi adalah prinsip OOP di mana data disembunyikan (private) dan hanya bisa diakses atau dimodifikasi melalui metode publik (getter dan setter). Dalam kode Anda, enkapsulasi digunakan di kelas Barang dan Gudang.

Contoh Enkapsulasi di Kelas Barang :

```
public class Barang {
    private String nama;
    private String kategori;
    private double hargaSatuan;
    private int stok;
```

Penjelasan :

- Atribut nama, kategori, hargaSatuan, dan stok dideklarasikan dengan akses private, sehingga tidak bisa diakses langsung dari luar kelas.
- Metode getNama(), setNama(), getKategori(), setKategori(), getHargaSatuan(), setHargaSatuan(), getStok(), dan setStok() adalah metode publik yang memungkinkan akses dan modifikasi atribut tersebut secara terkendali.

2. ***Inheritance (Pewarisan) dan Subclass***

Inheritance adalah konsep OOP di mana sebuah kelas dapat mewarisi sifat dan metode dari kelas induknya. Pada kode Anda, terdapat pewarisan yang digunakan untuk membuat subclass yang menangani tombol dalam tabel.

Contoh Inheritance di Kelas ButtonRenderer dan ButtonEditor :

```
private class ButtonRenderer extends JButton implements TableCellRenderer {
    public ButtonRenderer() {
        setOpaque(isOpaque:true);
    }

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value,
                                                   boolean isSelected, boolean hasFocus,
                                                   int row, int column) {
        setText(value.toString());
        return this;
    }
}

private class ButtonEditor extends DefaultCellEditor {
    private JButton button;
    private String label;
    private int selectedRow;
```

3. ***Polimorfisme***

Polimorfisme adalah kemampuan objek untuk mengambil banyak bentuk. Polimorfisme biasanya tercapai melalui overriding atau overloading metode. Dalam kasus ini, ButtonRenderer dan ButtonEditor menimpa (override) metode getTableCellRendererComponent() dan getTableCellEditorComponent().

Contoh Polimorfisme di Kelas ButtonRenderer dan ButtonEditor :

```
    public Component getTableCellRendererComponent(JTable table, Object value,
                                                   boolean isSelected, boolean hasFocus,
                                                   int row, int column) {
        setText(value.toString());
        return this;
    }
}

private class ButtonEditor extends DefaultCellEditor {
    private JButton button;
    private String label;
    private int selectedRow;

    public ButtonEditor(JButton checkBox) {
        super(new JCheckBox());
        button = new JButton();
        button.setOpaque(isOpaque:true);
        button.addActionListener(e -> fireEditingStopped());
    }

    @Override
    public Component getTableCellEditorComponent(JTable table, Object value,
                                                 boolean isSelected, int row, int column) {
        label = value.toString();
        selectedRow = row;
        button.setText(label);
        return button;
    }
```

4. ***Subclass dan Superclass (Extends)***
   Kelas-kelas seperti ButtonRenderer dan ButtonEditor mewarisi kelas JButton dan DefaultCellEditor dengan menggunakan keyword extends. Kelas ini adalah contoh dari subclass yang meng-extend superclassnya.
   Contoh penggunaan extends :

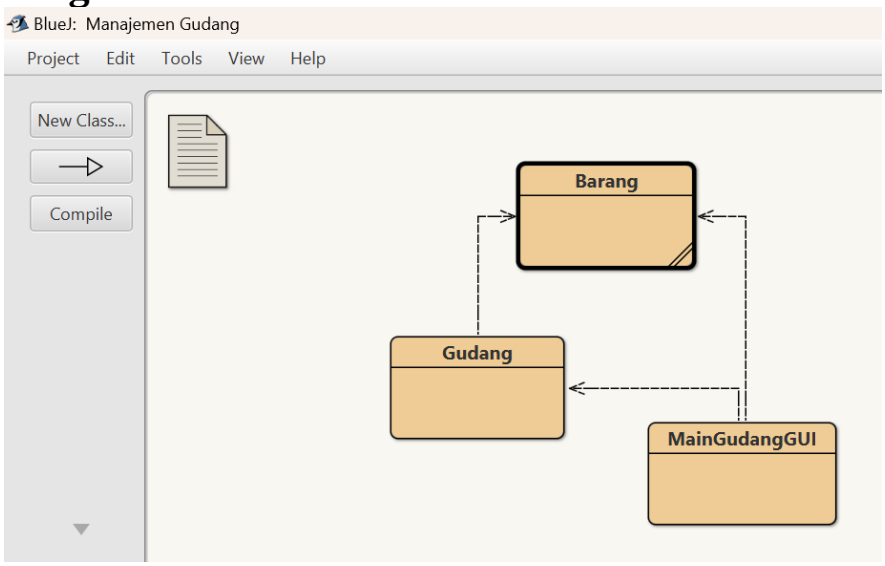
```
private class ButtonRenderer extends JButton implements TableCellRenderer {
    public ButtonRenderer() {
        setOpaque(isOpaque:true);
    }

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value,
                                                   boolean isSelected, boolean hasFocus,
                                                   int row, int column) {
        setText(value.toString());
        return this;
    }
}

private class ButtonEditor extends DefaultCellEditor {
    private JButton button;
    private String label;
    private int selectedRow;
```

5. ***Abstraksi (Abstract)***
   Meskipun dalam kode ini tidak ada kelas yang mendeklarasikan diri sebagai abstract, prinsip abstraksi tetap terlihat dalam kelas-kelas yang mengimplementasikan antarmuka seperti TableCellRenderer dan DefaultCellEditor. Kelas-kelas ini mengabstraksi logika render dan edit tombol dalam tabel, yang kemudian diimplementasikan secara spesifik dalam subclass.

Diagram Kelas



Tampilan Program