



KSP JAVA 2025



OBJECT PERSISTENCE 2

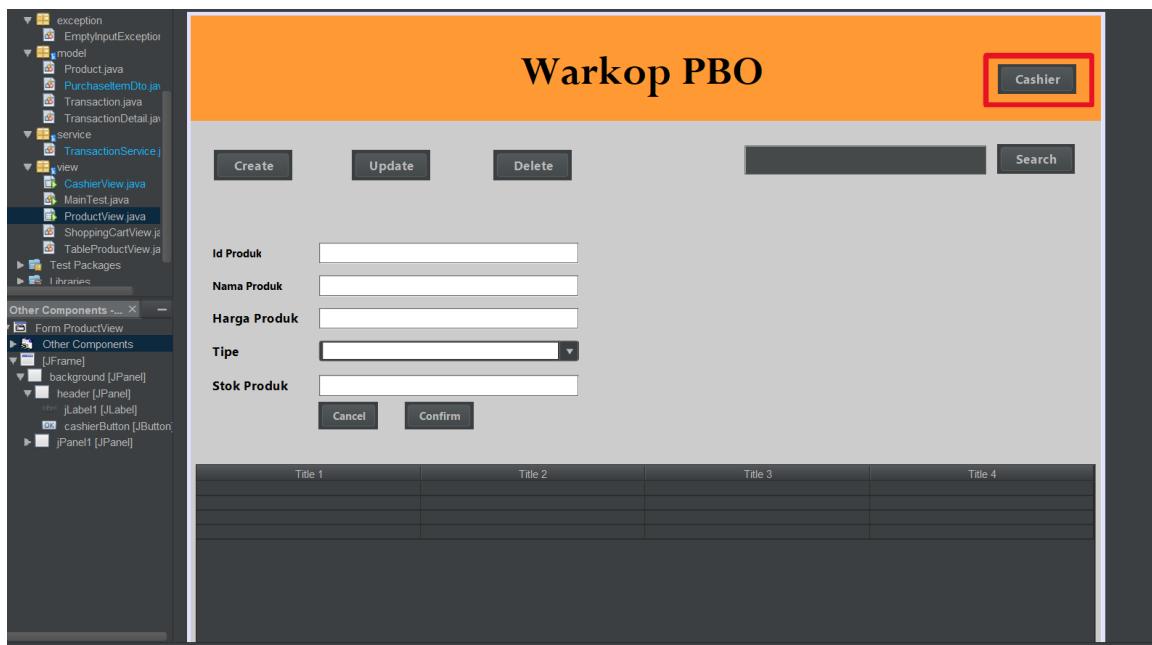
[VIEW](#)

Kevin Philips Tanamas
Tok Se Ka

Object Persistence 2 - View

Pada kali ini kita akan membuat tampilan Cashier dari Warkop PBO yang telah dibuat pada modul sebelumnya. Untuk tampilannya sendiri, memiliki banyak kesamaan dengan modul sebelumnya (sehingga dapat **dicopy** saja dari panel product). Hanya terdapat penambahan dan modifikasi dibagian body dari CashierView itu sendiri, penambahan terdiri dari 2 table baru untuk melihat stok data (**productTable**) dan untuk melihat keranjang dari pembelian yang akan dilakukan (**tableChart**). (Jika teman-teman mengerjakan modul view terlebih dahulu sebelum backend, maka akan terdapat beberapa **error**. Errornya bisa **diabaikan** saja terlebih dahulu).

1. Sebelum membuat **CashierView**, kita perlu membuat penghubung antara **ProductView** ke **CashierView**. Untuk mengakses pengkodean tombol Cashier, teman-teman dapat melakukan **double klik** pada button Cashier di **ProductView**

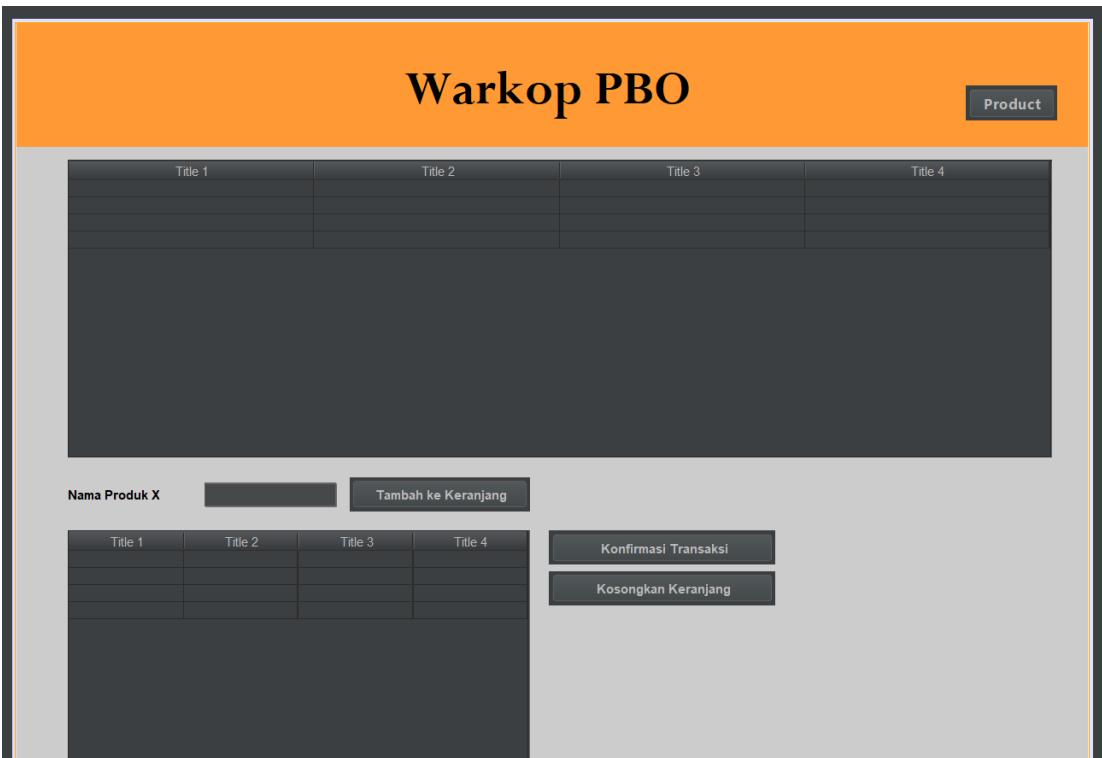


Pada tombol navigasi ke Cashier, tambahkan code berikut :

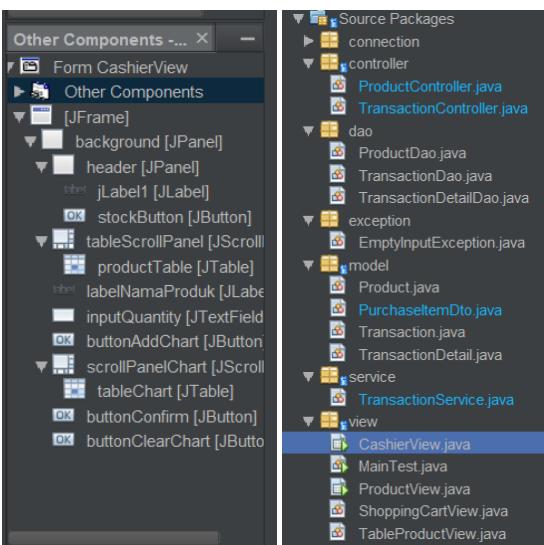
```
private void cashierButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    CashierView cv = new CashierView();  
    this.dispose();  
    cv.setVisible(true);  
}
```

- Kode ini menyelesaikan/menutup view yang sedang ditampilkan, dan menginisialisasi view baru yaitu **CashierView**.

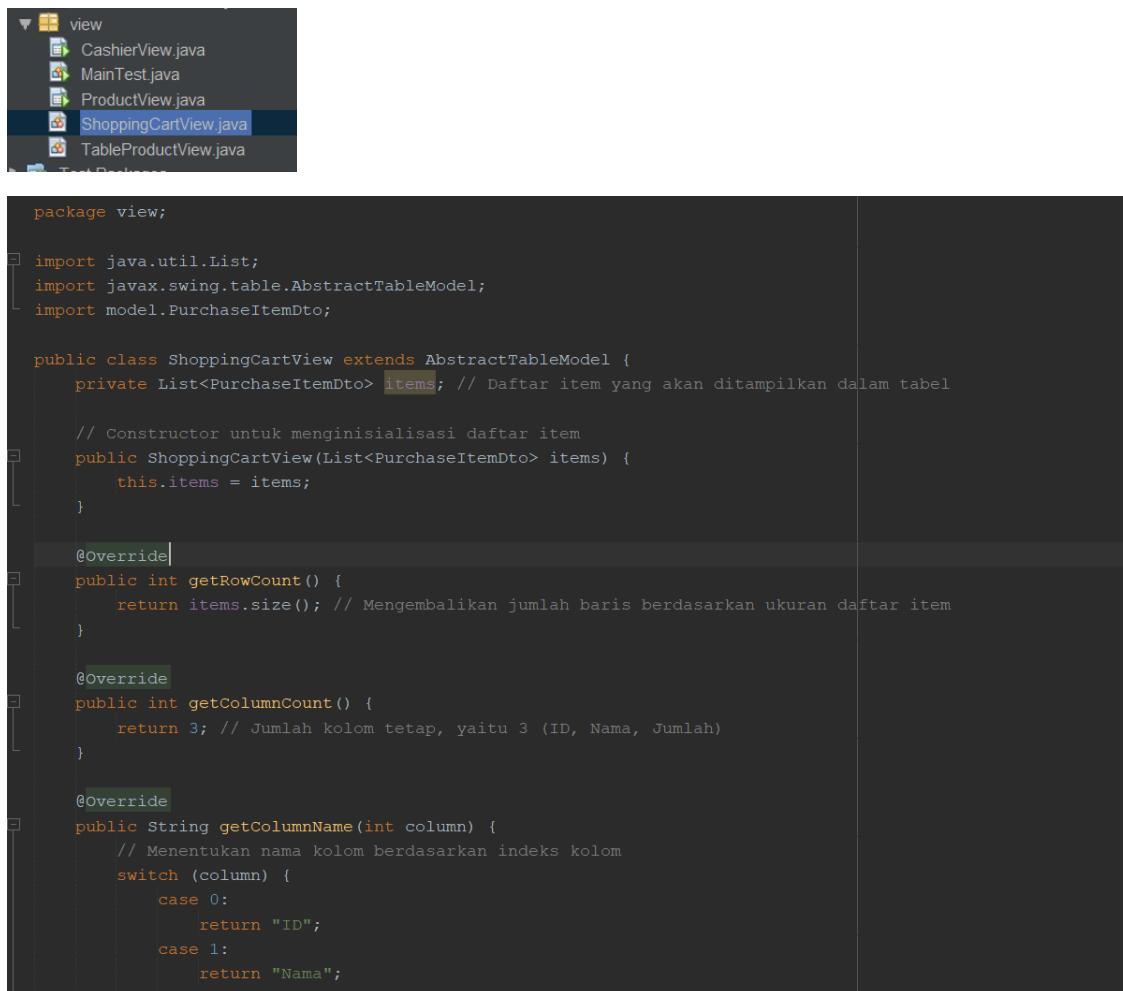
2. **CashierView** memiliki header yang mirip dengan **ProductView** (bedanya tombol navigasinya untuk kembali ke halaman **ProductView**), buatlah sesuai tampilan ini, dan sesuaikan nama variabel dari setiap komponennya. (Untuk ukuran dari panel, dan header **sama** seperti **ProductView**).



Untuk nama variabel dari setiap komponennya, dan susunan package filenya bisa disesuaikan seperti gambar dibawah ini:



- CashierView memiliki 2 tabel, tabel yang berada di sisi atas untuk menampilkan daftar produk (nama variabelnya : **productTable**) dan tabel disisi kiri bawah untuk menampilkan daftar produk yang diinputkan kedalam keranjang (nama variabelnya : **tableChart**)
 - Sebuah input field (**inputQuantity**), untuk menginputkan jumlah item yang ingin dibeli per produknya
 - Sebuah label yang akan akan diset dengan nama produk yang sesuai dengan item produk yang sedang dipilih (**labelNamaProduk**).
 - Tombol untuk konfirmasi transaksi (nama variabel : **buttonConfirm**) dan pembatalan transaksi (nama variabel : **buttonClearChart**)
3. Sebelum masuk ke pengkodean **CashierView**, kita perlu membuat **ShoppingCartView** untuk mapping data collection **PurchasedItemDto** agar dapat ditampilkan ke dalam tabel. Pengkodean **ShoppingCartView** bisa dilakukan dengan kode sebagai berikut:



The screenshot shows a Java file named `ShoppingCartView.java` in a package named `view`. The code implements `AbstractTableModel` and overrides methods for row and column counts, and column names. It also contains logic to return the size of the `items` list.

```

package view;

import java.util.List;
import javax.swing.table.AbstractTableModel;
import model.PurchaseItemDto;

public class ShoppingCartView extends AbstractTableModel {
    private List<PurchaseItemDto> items; // Daftar item yang akan ditampilkan dalam tabel

    // Constructor untuk menginisialisasi daftar item
    public ShoppingCartView(List<PurchaseItemDto> items) {
        this.items = items;
    }

    @Override
    public int getRowCount() {
        return items.size(); // Mengembalikan jumlah baris berdasarkan ukuran daftar item
    }

    @Override
    public int getColumnCount() {
        return 3; // Jumlah kolom tetap, yaitu 3 (ID, Nama, Jumlah)
    }

    @Override
    public String getColumnName(int column) {
        // Menentukan nama kolom berdasarkan indeks kolom
        switch (column) {
            case 0:
                return "ID";
            case 1:
                return "Nama";
            case 2:
                return "Jumlah";
        }
    }
}

```

```

@Override
public String getColumnName(int column) {
    // Menentukan nama kolom berdasarkan indeks kolom
    switch (column) {
        case 0:
            return "ID";
        case 1:
            return "Nama";
        case 2:
            return "Jumlah";
        default:
            return null; // Jika indeks kolom tidak valid
    }
}

@Override
public Object getValueAt(int rowIndex, int columnIndex) {
    // Mengambil data dari item berdasarkan baris dan kolom
    switch (columnIndex) {
        case 0:
            return items.get(rowIndex).getProduct().getId(); // ID produk
        case 1:
            return items.get(rowIndex).getProduct().getName(); // Nama produk
        case 2:
            return items.get(rowIndex).getQuantityPurchased(); // Jumlah yang dibeli
        default:
            return null; // Jika indeks kolom tidak valid
    }
}

```

4. Selanjutnya merupakan bagian pengkodean **CashierView**, Untuk mengakses pengkodean **CashierView** bisa melakukan klik pada file **CashierView** dibagian source. Fungsi utama dari **CashierView** adalah untuk menampilkan daftar produk, memilih dan menambahkan keranjang belanja, serta melakukan checkout (menyelesaikan transaksi).



```
5 package view;
6
7 import controller.ProductController;
8 import controller.TransactionController;
9 import java.util.ArrayList;
10 import java.util.HashSet;
11 import java.util.List;
12 import java.util.stream.Collectors;
13 import javax.swing.JOptionPane;
14 import javax.swing.table.TableModel;
15 import model.Product;
16 import model.PurchaseItemDto;
17
18 /**
19 *
20 * @author Kevin Philips Tanamas
21 */
22 public class CashierView extends javax.swing.JFrame {
23     private final ProductController productController = new ProductController();
24     private final TransactionController transactionController = new TransactionController();
25
26     Product selectedProduct = null;
27     List<Product> products;
28     List<PurchaseItemDto> purchaseItemDtos = new ArrayList<>();
29
30     public CashierView() {
31         initComponents();
32         setComponent(false);
33         reloadTables();
34     }

```

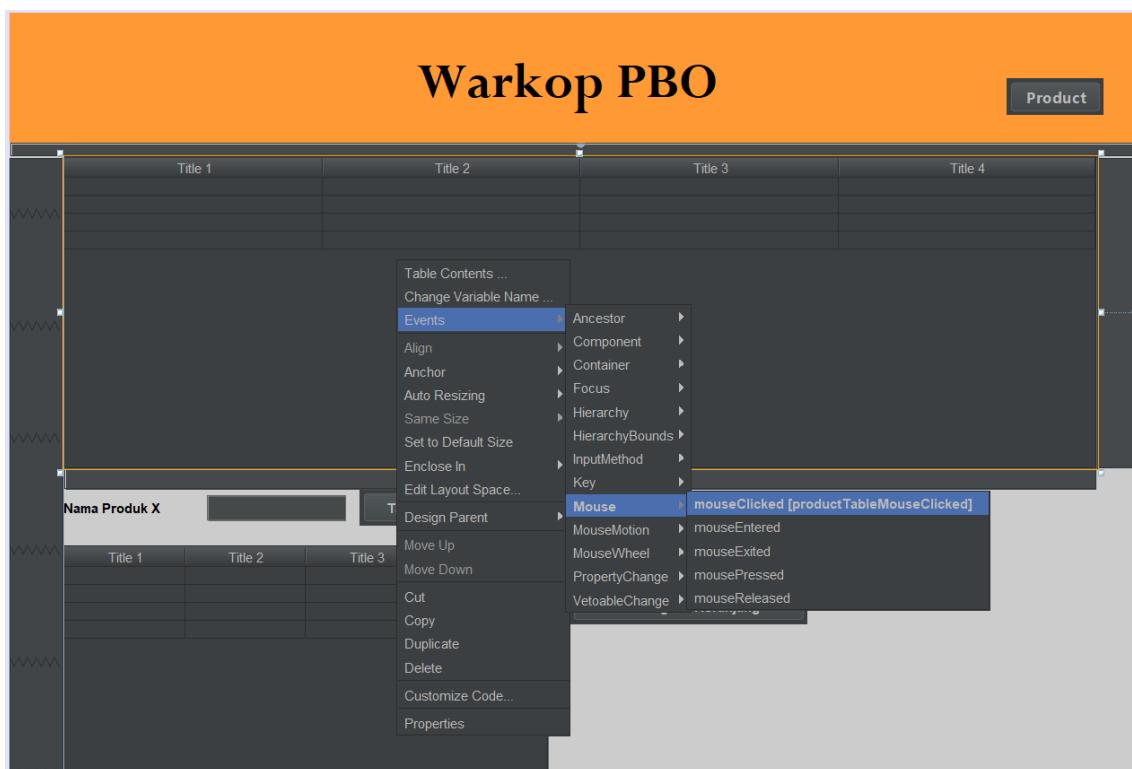
```

36     public void setComponent(boolean value){
37         inputQuantity.setEnabled(value);
38         buttonAddChart.setEnabled(value);
39         buttonClearChart.setEnabled(false);
40         buttonConfirm.setEnabled(false);
41     }
42
43     public TableProductView mapToTableProduct(String query) {
44         // Ambil semua produk dari controller berdasarkan query
45         List<Product> allProducts = productController.read(query);
46         List<Product> filteredProducts = new ArrayList<>();
47
48         for (Product p : allProducts) {
49             if (p.getStock() > 0) {
50                 filteredProducts.add(p);
51             }
52         }
53         this.products = filteredProducts;
54         if (this.products.isEmpty()) {
55             System.out.println("Product Kosong");
56         }
57         return new TableProductView(this.products);
58     }
59
60     public ShoppingCartView mapToShoppingChart() {
61         return new ShoppingCartView(this.purchaseItemDtos);
62     }
63
64     public void reloadTables() {
65         productTable.setModel(mapToTableProduct(""));
66         tableChart.setModel(mapToShoppingChart());
67     }

```

- **selectedProduct**: merepresentasikan produk yang sedang dipilih
- **products**: representasi daftar produk
- **PurchasedItemDtos**: representasi daftar pembelian produk.
- **setComponent**: method untuk mendefinisikan kondisi komponen
- **reloadTables**: adalah fungsi untuk menampilkan data ditabel.
- **setComponent & reloadTables** ini akan dipanggil setelah saat inisialisasi dan setelah aksi-aksi tertentu.
- **mapToTableProduct**: Mengubah daftar produk menjadi model tabel (**TableProductView**) berdasarkan pencarian.
- **mapToShoppingChart**: Mengubah daftar item pembelian (**purchaseItemDtos**) menjadi model tabel (**ShoppingCartView**).

- **reloadTables:** Mengisi ulang data pada tabel produk dan tabel keranjang (melakukan refresh pada table saat ada perubahan).
5. Selanjutnya masih di file **CashierView**, kita akan melakukan inisialisasi pada setiap component yang ada pada file **CashierView**. Pada component **productTable** karena kita akan melakukan klik pada table nantinya (untuk memilih products yang akan kita beli), maka kita perlu mendefinisikan events yang akan kita lakukan. Klik kanan pada **productTable**, klik events, dan pilih **mouseClicked** untuk mendefinisikan events.



Kemudian isikan pengkodean dibawah ini setelah mengklik events **mouseClicked**

```

private void productTableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    inputQuantity.setText("");
    setComponent(true);

    // Mendapatkan indeks baris yang diklik oleh pengguna pada tabel produk.
    int clickedRow = productTable.getSelectedRow();
    System.out.println("click row: " + clickedRow);
    System.out.println("Products List: " + products);

    if (products == null || products.isEmpty()) {
        // Menampilkan pesan dialog kepada pengguna jika produk tidak ditemukan atau kosong.
        JOptionPane.showMessageDialog(this, "Produk tidak ditemukan atau kosong.");
        return;
    }

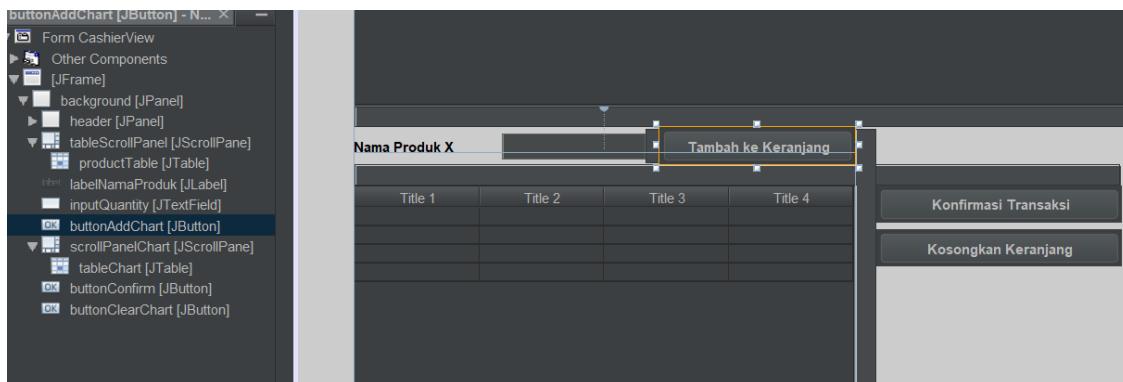
    // Mengambil produk yang dipilih berdasarkan indeks baris yang diklik pengguna.
    selectedProduct = products.get(clickedRow);
    System.out.println("selected product: " + selectedProduct);

    // Menampilkan nama produk yang dipilih pada label
    labelNamaProduk.setText(selectedProduct.getName());
}

```

- **clickedRow:** Mengambil indeks baris tabel yang diklik oleh pengguna.
- **selectedProduct:** Menggunakan indeks baris untuk mendapatkan produk yang dipilih dari daftar produk.

Selanjutnya untuk menginisialisasi **buttonAddChart**, dapat dilakukan dengan melakukan **double klik** pada componentnya secara langsung, button ini digunakan untuk menambahkan item ke keranjang sesuai dengan item yang sedang dipilih.



```
private void buttonAddChartActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    buttonConfirm.setEnabled(true);
    buttonClearChart.setEnabled(true);

    // Membuat objek 'PurchaseItemDto' berdasarkan produk yang dipilih dan jumlah yang dimasukkan pengguna.
    // 'inputQuantity.getText()' mengambil jumlah yang diinput oleh pengguna, dan diubah menjadi integer.
    PurchaseItemDto purchaseItemDto = new PurchaseItemDto(selectedProduct, Integer.valueOf(inputQuantity.getText()));
    purchaseItemDtos.add(purchaseItemDto);

    System.out.println("Items in cart: " + purchaseItemDtos.size());
    System.out.println("Last added item: " + purchaseItemDto.getProduct().getName());

    // Memperbarui tabel di UI untuk mencerminkan perubahan pada data.
    reloadTables();
}
```

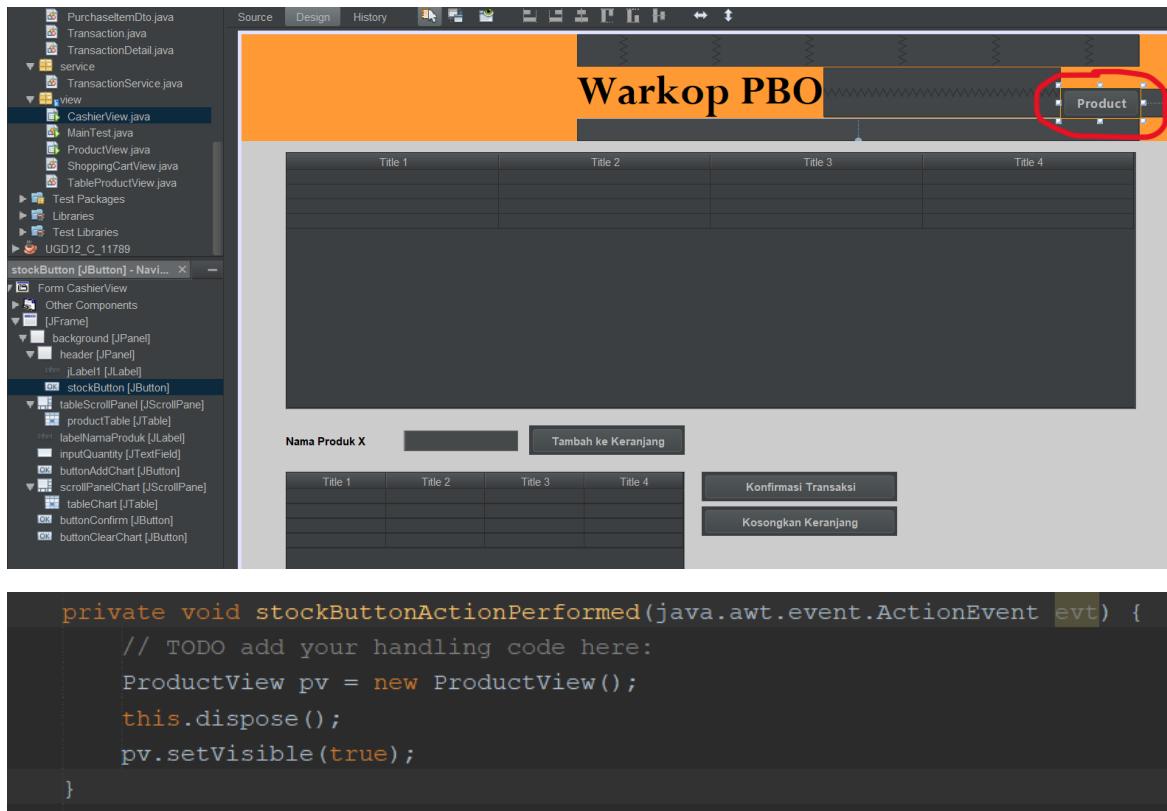
Selanjutnya kita akan melakukan inisialisasi pada component **buttonConfirm**, button ini akan digunakan untuk mengkonfirmasi pembelian. Untuk melakukan inisialisasi sama seperti sebelumnya, lakukan dengan **double klik** pada component.

```
private void buttonConfirmActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String total = transactionController.create(purchaseItemDtos);
    purchaseItemDtos.clear();
    reloadTables();
    JOptionPane.showMessageDialog(
        null,
        "Transaksi Berhasil, Total Transaksi: " + total,
        "Success",
        JOptionPane.INFORMATION_MESSAGE
    );
}
```

Untuk **buttonClearChart**, kita akan gunakan ini untuk menghapus seluruh barang yang ada dikeranjang dan melakukan reset ke keadaan **default**. Dapat dilakukan menggunakan pengkodean dibawah ini:

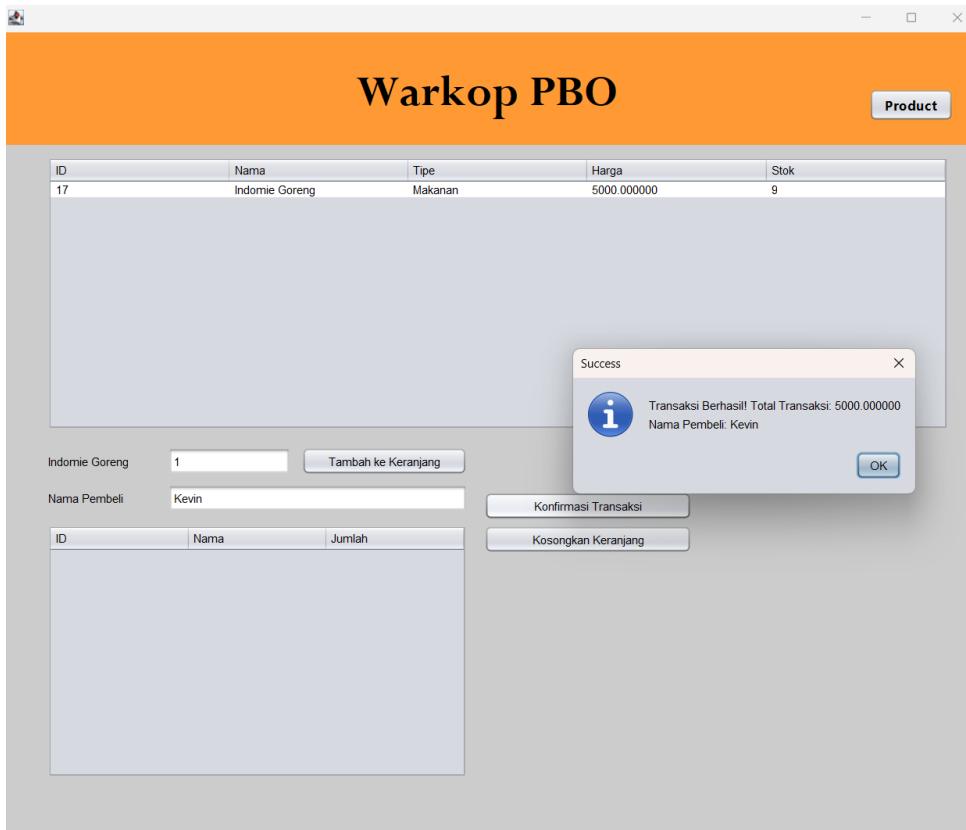
```
private void buttonClearChartActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    buttonConfirm.setEnabled(false);
    buttonClearChart.setEnabled(false);
    purchaseItemDtos.clear();
    reloadTables();
}
```

Yang terakhir, lakukan inisialisasi pada **stockButton** untuk dapat berpindah kembali ke menu product. Pengkodean dapat dilakukan seperti dibawah ini:



Challenge 🎉

Silahkan buat 1 inputan baru berupa nama pembeli di view kalian, kemudian tampilkan nama pembeli ketika transaksi berhasil dilakukan. Pastikan data nama pembeli masuk ke database
(Hint: dilakukan modifikasi pada controller dan service transaksi)



	id	date	total	cashier	buyer
1	1	2025-04-29 12:42:09	5000	cashier-dummy	Kevin

---- Selamat Belajar, GBU ----