LAPORAN PRAKTKUM

CLEAN CODE APLIKASI POIN OFF SALE

Dibuat untuk memenuhi salah satu tugas mata kuliah Pemrograman Berbasis Objek yang diampu oleh Bapak Ardhian Ekawijana

Disusun Oleh:

NIM	: 241511094
NAMA	: ZAHRA ALDILA
KELAS	: 2C
PROGRAM STUDI	: D3 – TEKNIK INFORMATIKA
JURUSAN	: TEKNIK KOMPUTER DAN INFORMATIKA

Program Studi D-3 Teknik Informatika

Jurusan Teknik Komputer dan Informatika

Politeknik Negeri Bandung

2025

DAFTAR ISI

A. Pengertian

1. Graphical User Interface

Graphical User Interface (GUI) adalah antarmuka pengguna yang memungkinkan interaksi antara manusia dan komputer melalui elemen-elemen grafis seperti tombol, jendela, ikon, menu, dan teks. Dengan GUI, pengguna tidak perlu mengetikkan perintah secara manual di command line, tetapi cukup menggunakan mouse dan keyboard untuk mengoperasikan aplikasi.

GUI berfungsi untuk mempermudah penggunaan program sehingga lebih intuitif dan efisien. Dalam konteks pemrograman, GUI dibangun menggunakan komponen visual (seperti tombol, label, dan kotak teks) yang dapat diatur dan diatur perilakunya melalui kode program.

Ciri-ciri utama GUI antara lain:

a. Menggunakan elemen visual seperti window, menu, icon, dan button.
b. Memungkinkan interaksi secara langsung dengan elemen tersebut (event-driven).
c. Memberikan tampilan yang interaktif dan mudah digunakan.

2. Java Swing

Java Swing adalah salah satu library GUI di bahasa pemrograman Java yang disediakan dalam paket javax.swing. Swing digunakan untuk membuat antarmuka grafis berbasis desktop (desktop application) yang berjalan di berbagai sistem operasi karena sifat Java yang platform independent.

Swing merupakan pengembangan dari Abstract Window Toolkit (AWT) dengan tampilan dan fitur yang lebih lengkap. Komponen-komponen dalam Swing seperti JFrame, JPanel, JButton, JLabel, JTable, dan JTextField dapat digunakan untuk membangun jendela aplikasi yang interaktif.

Keunggulan Java Swing:

a. Memiliki banyak komponen siap pakai untuk membangun antarmuka.
b. Dapat dikustomisasi (ubah warna, ukuran, font, dan tata letak).
c. Mendukung berbagai layout manager seperti BorderLayout, FlowLayout, dan GridLayout.

B. Pengerjaan

1. Link GitHub

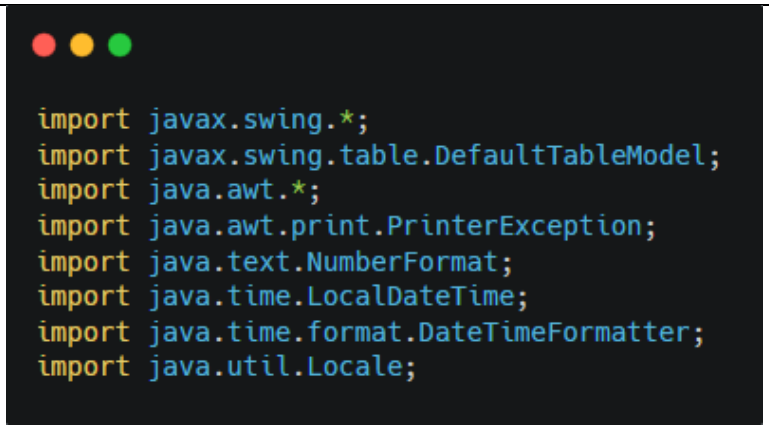
https://github.com/zahraldila/TugasPBOWeek12.git

2. Deskripsi Program

Program ini merupakan tampilan awal aplikasi Point of Sales (POS) yang dibuat menggunakan Java Swing. Antarmuka dirancang menggunakan komponen seperti JFrame, JPanel, JTable, JButton, JLabel, dan JTextArea untuk menampilkan daftar produk, keranjang belanja, serta area struk. Layout disusun dengan kombinasi BorderLayout dan FlowLayout agar tampilan rapi dan menyerupai sistem kasir pada umumnya.

Saat ini, program hanya berfokus pada pembuatan antarmuka grafis (GUI) tanpa logika perhitungan atau transaksi. Tujuannya adalah untuk melatih pemahaman mahasiswa dalam menempatkan dan mengatur komponen GUI di dalam jendela aplikasi Java sebelum ditambahkan fungsi interaktif seperti “Add to Cart”, “Checkout”, dan “Cetak”.

3. Problem yang Muncul

Warning
Move this file to a named package
Code awal

Penyebab
File POSFrame.java tidak memiliki deklarasi package di bagian awal kode sehingga berada dalam *default package* (tanpa nama). Kondisi ini ditandai oleh SonarQube karena

penggunaan *default package* tidak direkomendasikan dalam pengembangan proyek Java. Hal tersebut dapat menyebabkan kesulitan dalam pengelolaan struktur kode, membatasi kemampuan kelas untuk di-*import* dari package lain, serta tidak sesuai dengan standar konvensi struktur proyek Java yang menggunakan hierarki package → folder → file.
Solusi
Menambahkan deklarasi package id.ac.polban.week12; di awal file agar file berada di dalam package yang bernama dan sesuai standar Java.
Code setelah diperbaiki

```java
package id.ac.polban.week12;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.print.PrinterException;
import java.text.NumberFormat;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Locale;
```

Warning
Remove the "tblKeranjang" field and declare it as a local variable in the relevant methods.
Code awal

```java
    private JTable tblKeranjang;
    private JLabel lblTotal, lblPoints;
    private JButton btnCheckout, btnCetak;
    private JTextArea txtStruk;
```

Penyebab
Variabel tblKeranjang dideklarasikan sebagai field kelas padahal hanya digunakan di dalam satu metode, sehingga SonarQube menilai scope-nya terlalu luas dan tidak efisien.
Solusi

Menghapus field tblKeranjang dari kelas karena hanya digunakan secara lokal di metode GUI, sehingga peringatan SonarQube hilang.

Code setelah diperbaiki

```
    private JLabel lblTotal, lblPoints;
    private JButton btnCheckout, btnCetak;
    private JTextArea txtStruk;
```

```
        JTable tblKeranjang = new JTable(modelCart);
        tblKeranjang.setFillsViewportHeight(true);
        wrapper.add(new JScrollPane(tblKeranjang), BorderLayout.CENTER);
```

Warning

Declare "lblPoints" on a separate line.

Code awal

```
    private JLabel lblTotal, lblPoints;
    private JButton btnCheckout, btnCetak;
    private JTextArea txtStruk;
```

Penyebab

Variabel lblTotal dan lblPoints dideklarasikan dalam satu baris yang sama, sehingga menurunkan keterbacaan kode.

Solusi

Memisahkan deklarasi variabel menjadi dua baris: private JLabel lblTotal; dan private JLabel lblPoints; agar lebih jelas dan sesuai standar penulisan kode Java.

Code setelah diperbaiki

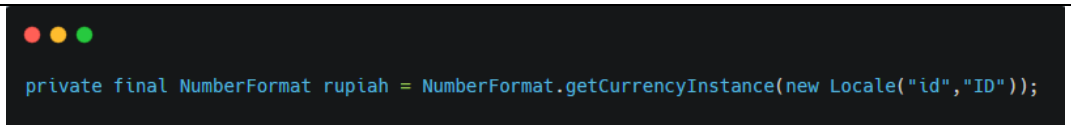
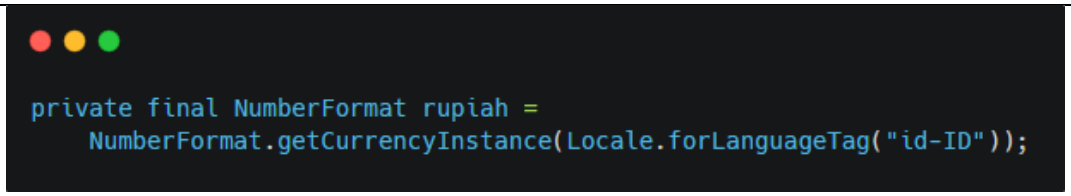
```
    private JLabel lblTotal;
    private JLabel lblPoints;
    private JButton btnCheckout, btnCetak;
    private JTextArea txtStruk;
```

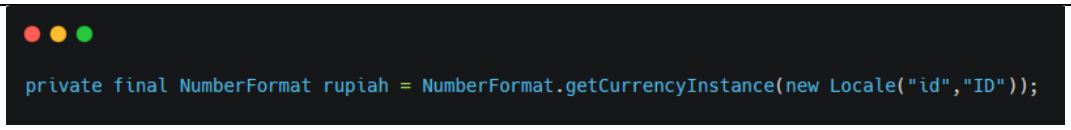
Warning
Declare "btnCetak" on a separate line.
Code awal

```
    private JLabel lblTotal;
    private JLabel lblPoints;
    private JButton btnCheckout, btnCetak;
    private JTextArea txtStruk;
```

Penyebab
Variabel btnTambah, btnHapus, dan btnCetak dideklarasikan dalam satu baris, membuat kode kurang rapi dan sulit dibaca.
Solusi
Memisahkan deklarasi menjadi tiga baris terpisah: private JButton btnTambah;, private JButton btnHapus;, dan private JButton btnCetak; agar lebih mudah dibaca dan sesuai standar konvensi Java.
Code setelah diperbaiki

```
    private JLabel lblTotal;
    private JLabel lblPoints;
    private JButton btnCheckout;
    private JButton btnCetak;
    private JTextArea txtStruk;
```

Warning
The constructor Locale(String, String) is deprecated since version 19
Code awal

Penyebab
Penggunaan konstruktor new Locale("id", "ID") sudah tidak disarankan karena API tersebut *deprecated* pada Java versi 19.
Solusi
Mengganti konstruktor new Locale("in", "ID") dengan Locale.forLanguageTag("id-ID") agar sesuai dengan standar Java 19 dan menghilangkan peringatan deprecated.
Code setelah diperbaiki


Warning
Remove this use of "Locale"; it is deprecated.
Code awal

Penyebab
Penggunaan kelas Locale dengan konstruktor lama (new Locale("id","ID")) sudah tidak direkomendasikan pada Java versi 19 ke atas, sehingga menimbulkan peringatan deprecated.
Solusi
Mengganti penggunaan new Locale("id","ID") dengan Locale.forLanguageTag("id-ID") agar sesuai standar API baru. Setelah diperbaiki, peringatan deprecated pada Locale ikut hilang.
Code setelah diperbaiki

```
private final NumberFormat rupiah =
    NumberFormat.getCurrencyInstance(Locale.forLanguageTag("id-ID"));
```

Warning

Define a constant instead of duplicating this literal "==========" 4 times. [+4 locations]

Code awal

```
StringBuilder sb = new StringBuilder();
        sb.append("==========\n");
        sb.append("              STRUK BELANJA              \n");
        sb.append("==========\n");
        sb.append("Waktu : ")
          .append(LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm")))
          .append("\n\n");
```

Penyebab

String literal "==========\n" digunakan berulang kali di beberapa baris kode, menyebabkan duplikasi dan menurunkan maintainability.

Solusi

Membuat konstanta LINE_SEPARATOR untuk menggantikan string "==========\n" yang berulang, sehingga kode lebih mudah dirawat dan sesuai standar SonarQube.

Code setelah diperbaiki

```
        sb.append(LINE_SEPARATOR);
        sb.append("              STRUK BELANJA              \n");
        sb.append(LINE_SEPARATOR);
        sb.append("Waktu : ")
          .append(LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm")))
          .append("\n\n");
```

```
        sb.append(LINE_SEPARATOR);
        sb.append("Terima kasih atas kunjungan Anda!\n");
        sb.append(LINE_SEPARATOR);
```

Warning
Remove this block of code, fill it in, or add a comment explaining why it is empty.
Code awal

```
    public static void main(String[] args) {
        try { UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName()); }
        catch (Exception ignored) {}
        SwingUtilities.invokeLater(() -> new POSFrame().setVisible(true));
    }
```

Penyebab
Terdapat blok kode {} yang kosong tanpa isi maupun komentar penjelasan (misalnya pada catch block atau event handler).
Solusi
Mengisi blok catch dengan ex.printStackTrace(); agar exception tidak diabaikan dan dapat ditelusuri jika terjadi error.
Code setelah diperbaiki

```
    public static void main(String[] args) {
        try { UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName()); }
        catch (Exception ex) {
            ex.printStackTrace();
        }
        SwingUtilities.invokeLater(() -> new POSFrame().setVisible(true));
    }
```

C. Lesson Learned

Pada praktikum kali ini, saya mempelajari cara memanfaatkan SonarQube untuk melakukan analisis kualitas kode pada aplikasi Java Swing (GUI). Jika sebelumnya saya hanya berfokus pada bagaimana program dapat dijalankan dengan benar, setelah menggunakan SonarQube saya menyadari bahwa kualitas program juga sangat dipengaruhi oleh keteraturan struktur kode, tingkat keterbacaan, serta penerapan standar penulisan yang baik.

Beberapa temuan seperti deklarasi variabel yang disatukan dalam satu baris, penggunaan konstruktor yang sudah tidak direkomendasikan (deprecated), dan blok kode kosong tanpa penjelasan membuat saya lebih memahami pentingnya prinsip clean code. Dari proses perbaikan tersebut, saya belajar bagaimana menulis kode yang lebih rapi, mudah dikelola, dan sesuai dengan praktik profesional di dunia industri.

Selain itu, saya juga memahami fungsi Look and Feel pada Java Swing yang berguna untuk menyesuaikan tampilan aplikasi dengan sistem operasi pengguna. Hal ini membuat saya sadar bahwa aspek tampilan pun dapat memunculkan peringatan jika tidak diatur dengan tepat.

Secara keseluruhan, praktikum ini memberikan pengalaman berharga tentang bagaimana alat bantu seperti SonarQube dapat meningkatkan kualitas perangkat lunak. Tidak hanya membantu memperbaiki kesalahan teknis, tetapi juga menumbuhkan kebiasaan berpikir lebih teliti, terstruktur, dan profesional dalam menulis program..