

# LAPORAN JAVA CLEAN CODE

Diajukan untuk memenuhi salah satu tugas Mata kuliah PBO Praktek



Disusun Oleh:  
Gema Adzan Firdaus (241511075)  
D3 2C  
Jurusan Teknik Komputer dan Informatika  
Program Studi D-3 Teknik Informatika Politeknik Negeri Bandung 2025

## **Daftar Isi**

Smell Code .....	3
Penjelasan & Penerapan Clean Code .....	3

# Smell Code

The screenshot shows the SonarQube Java plugin integrated into the Visual Studio Code interface. The code editor displays the file `PointOfSales.java`. The left sidebar shows a project structure with a folder named `pos` containing `PointOfSales.java`. The right sidebar shows a detailed list of code smells found by SonarQube, such as unused imports and static access. The bottom status bar indicates the code is ready for SonarQube analysis.

## Penjelasan & Penerapan Clean Code

### A. Move this file to a named package

Penerapan clean code disini Adalah menempatkan kode ini dalam sebuah folder lalu memasukkan package dari nama folder tersebut

This screenshot shows the same setup as the previous one, but with a key difference: the code editor now contains a `package pos;` statement at the top of `PointOfSales.java`. This change is intended to move the code from a unnamed package to a named package, which is a common practice in clean code principles. The SonarQube analysis results remain the same, indicating various code smells like unused imports and static access.

## B. Menghapus kode yang unused dan not used

Ada banyak import yang ditulis dalam kode namun tidak digunakan maka harus dihapus agar kode menjadi clean

Dalam kode saya banyak menggunakan import import yang ternyata pada akhirnya tidak terpakai dalam implementasi

The screenshot shows the SonarQube IDE interface with the Java project 'pos' open. The code editor displays the 'PointOfSales.java' file. The right-hand panel shows a 'Todos' list with one item: 'Completed (3/3) Restore cartTable field'. A detailed note explains that the field 'cartTable' was removed and restored, and provides guidance on best practices for field usage.

## C. cartTable

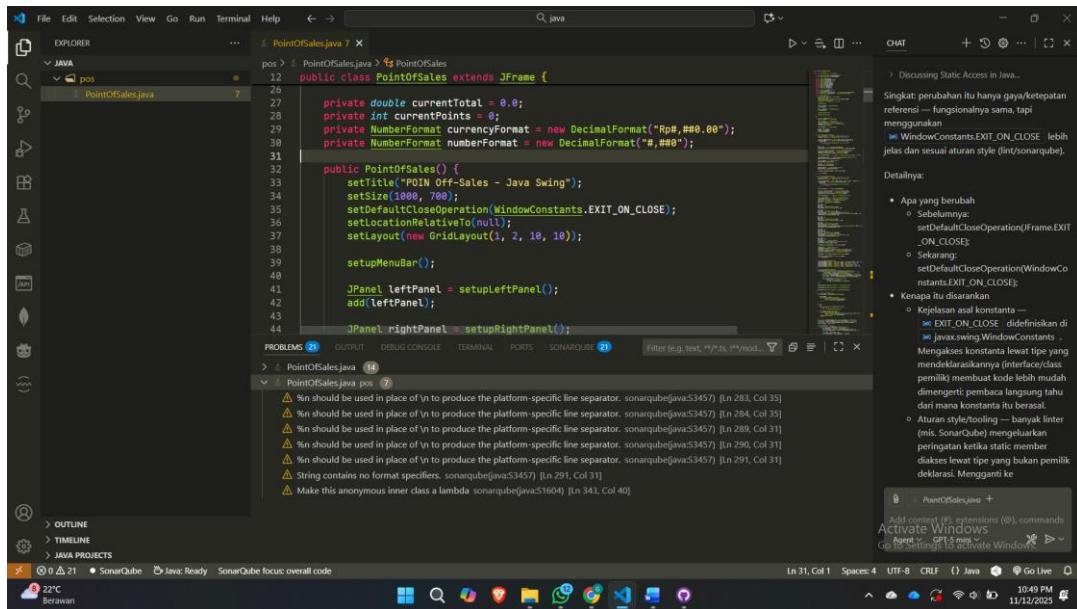
Penggunaan cartTable disini hanya digunakan dalam sebuah method maka mendefinisikan cartTable sebagai field merupakan bukan langkah yang baik jadi lebih baik untuk mengubahnya menjadi sebuah lokal variabel dalam method tersebut

The screenshot shows the SonarQube IDE interface with the Java project 'pos' open. The code editor displays the 'PointOfSales.java' file. The right-hand panel shows a 'Todos' list with one item: 'Make cartTable local again'. A detailed note explains that the field 'cartTable' was made local again and provides guidance on best practices for field usage.

#### D. Penggunaan EXIT\_ON\_CLOSE

Pada SonarCube lint dari sonarcube itu akan memberi peringatan untuk penggunaan konstanta dengan tidak langsung memanggil class/interface dimana konstanta itu didefinisikan atau tidak memanggil pemilik konstanta secara langsung

Untuk masalah apakah ada perbedaan diantara memanggil langsung class dengan bukan seperti implementasi sebelumnya itu tidak ada hanya karena linter sonarcube ini yang menyebabkan terbacanya kode ini sebagai smell code



E. `%n` should be used in place of `\n` to produce the platform-specific line separator.

Penggunaan /n menurut sonarcube merupakan smell code yang masih bisa dibuat lebih baik dengan menggunakan placeholder %n lalu menggunakan (nl) untuk membuat code selanjutnya ada pada next line dan juga digunakan untuk membuat varibel teks dengan mendefinisikan isi dari variable untuk dipakai secara langsung dalam sebuah method, dengan car aini mendukung adanya clean code

```

public class PointOfSales extends JFrame {
    private double currentTotal = 0.0;
    private int currentPoints = 0;
    private NumberFormat currencyFormat = new DecimalFormat("Rp#,##0.00");
    private NumberFormat numberFormat = new DecimalFormat("#,##0");
    private static final String RECEIPT_SEPARATOR = "=====";
    private static final String RECEIPT_DIVIDER = "-----";

    public PointOfSales() {
        setTitle("POINT Off-Sales - Java Swing");
        setSize(1000, 700);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new GridLayout(1, 2, 10, 10));
        JPanel leftPanel = setupLeftPanel();
        add(leftPanel);
    }

    JPanel setupLeftPanel() {
        ...
    }
}

```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | SONARQUBE | Filter (e.g. text, \*\*/\*, node\_modules/\*)

PointOfSales.java (14)

Make this anonymous inner class a lambda sonarqube(java:S1604) [Ln 346, Col 40]

CHAT | CHAT + | CHAT ... | CHAT perubahan?

ini benarin PointOfSales.java:274

Thought for a few seconds PointOfSales.java +7

Berhasil — saya mengekstrak string pemisah struk yang duplikat menjadi konstanta.

- RECEIPT\_SEPARATOR = "=====
- RECEIPT\_DIVIDER = "-----"

dan mengganti semua pemakaian literal duplikat di metode actionCheckout() .

Saya melihat satu peringatan gaya lagi (anonymous inner class di main ); mau saya ubah menjadi lambda juga untuk menghilangkan peringatan itu dan menyederhanakan kode?

OUTLINE | TIMELINE | JAVA PROJECTS

23°C Klat di sktr

In 43, Col 45 | Spaces: 4 | UTF-8 | CR/LF | Java | Go Live | 11:00 PM | 11/12/2023

## F. Make this anonymous inner class a lambda

Menyederhanakan code agar peringatan sonar code ini hilang

```

public class PointOfSales extends JFrame {
    private void updateTotal() {
        currentPoints = (int) (currentTotal / 1000);
        totalValueLabel.setText(currencyFormat.format(currentTotal));
        pointsValueLabel.setText(String.valueOf(currentPoints));
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new PointOfSales().setVisible(true));
    }
}

```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | SONARQUBE | Filter (e.g. text, \*\*/\*, node\_modules/\*)

PointOfSales.java (14)

Todos (7/7) | Todos (7/7) | Created 7 todos | Created 7 todos

menyederhanakan kode; outcome:  
SwingUtilities.invokeLater() akan memakai lambda dan fungsionalitas tidak berubah. Sekarang saya terapkan perubahan.

PointOfSales.java +14

Lama: SwingUtilities.invokeLater(new Runnable() { public void run() { new PointOfSales().setVisible(true); } }); Sekarang: SwingUtilities.invokeLater(() -> new PointOfSales().setVisible(true));

Kenapa dibalik:

- Singkatnya: kode lebih ringkas, lebih modern (Java 8+) dan menghilangkan peringatan SonarQube tentang anonymous inner class.
- Detalii:
  - Lambda mengurangi boilerplate (tidak perlu kelas anonim dan method run).
  - Fungsionalitas tidak berubah: lambda masih menjalankan pembuatan dan tampilan frame di Event Dispatch Thread.

OUTLINE | TIMELINE | JAVA PROJECTS

0, 14 • SonarQube | Java Ready | SonarQube focus: overall code

23°C Berawan

In 346, Col 24 | Spaces: 4 | UTF-8 | CR/LF | Java | Go Live | 11:07 PM | 11/12/2023

Github repo: