26 Agustus 2025

**LAPORAN TUGAS WORKSHOP**

“Resume dan Implementasi code convention”

Mata Kuliah Aplikasi Workshop Rekayasa Ulang Kode



Nama dosen pengampu:

Adam Shidqul Aziz S.ST., M.T

Dikerjakan oleh:

Nama : Kansa Adeneva

NRP : 3123600009

Kelas : 2 D4 IT A

* Resume Java Code Convention (Oracle Link)

1. Introduction

Konvensi kode dibuat agar kode lebih mudah dibaca, dipelihara, dan dibagikan. Mayoritas biaya software ada di pemeliharaan, jadi konsistensi gaya kode penting.

1. File Names

Nama file harus sama dengan nama kelas public.

Contoh:

public class MyApp { ... }

→ file bernama MyApp.java (Gunakan huruf besar di awal kata (CamelCase))

1. File Organization

Urutan isi file sumber:

1. Header comment → informasi pembuat, tanggal, hak cipta
2. Package statement
3. Import statement
4. Class/interface declaration:

* Variabel statis
* Variabel instans
* Konstruktor
* Method

1. Indentation

* Gunakan 4 spasi (jangan tab)
* Panjang baris maksimum 80 karakter (70 karakter untuk dokumentasi)
* Jika baris terlalu Panjang, lakukan line wrapping:
  + Pecah setelah koma atau operator
  + Selaraskan indentasi agar tetap terbaca
  + Jika terlalu dalam, tambahkan identasi 8 spasi ekstra

1. Comments
   1. Implementation Comments

* // → komentar satu baris
* /\* . . . \*/ → blok komentar beberapa baris
* Trailing comments: int age = 18; // minimal age
* End of line comments: beberapa baris // berturut, tapi jangan untuk teks Panjang.
  1. Documentation Comments (Javadoc)
* /\*\* . . . \*/ → untuk API (kelas, method, variabel public)
* Format: baris pertama /\*\*, baris tengah diawali \*, baris terakhir \*/.
  1. Best Practices
* Jangan menulis komentar yang sekadar mengulang isi kode.
* Gunakan untuk menjelaskan maksud/desain, bukan hal yang jelas.

1. Declatations
   1. Satu variabel per baris (hindari int a, b; )
   2. Deklarasi diletakkan di awal blok {}
   3. Variabel ‘for’ boleh dideklarasikan dalam ‘for’
   4. Inisialisasi variabel sedini mungkin
   5. Hindari menimpa variabel dengan nama sama di scope dalam
2. Statements
   1. Satu statement per baris
   2. Selalu gunakan {} walau hanya satu statement di if/for/while
   3. Switch:

* Gunakan ‘break’ setiap case
* Jika ‘fall through’, beri komentar /\* falls through \*/
  1. Try – catch : selalu gunakan blok {} yang rapi

1. White Space
   1. Gunakan spasi agar kode terbaca:

* Setelah koma → int x, y;
* Setelah keyword → if (true)
* Sekitar operator → a + b, tapi object.method() tanpa spasi
  1. Baris kosong:
* 2 baris antar bagian besar (antar kelas, antar import dan deklarasi)
* 1 baris antar method, sebelum komentar blok, antara variabel lokal dan statement pertama

1. Naming Conventions
   1. Kelas/Interface: CamelCase, huruf besar awal (StudentData)
   2. Method/Variabel: camelCase, huruf kecil awal (getName, studentName)
   3. Konstanta: HURUF\_BESAR dengan underscore (MAX\_SIZE)
   4. Nama harus deskriptif, hindari singkatan kecuali umum (i, j untuk loop)
2. Programming Practices
   1. Jangan jadikan variabel kelas / instans sebagai ‘public’ → gunakan getter / setter
   2. Akses static melalui nama kelas, bukan objek
   3. Hindari literal angka kecuali -1, 0, 1 → pakai konstanta
   4. Jangan gunakan penugasan berantai (a = b = c).
   5. Gunakan kurung pada ekspresi kompleks
   6. Simplifikasi return:

return isValid;

* 1. Gunakan komentar marker:
* FIXME → ada bug / harus diperbaiki
* XXX → kode ambigu tapi masih jalan
* Implementasikan code convention
  1. Code kotor / blm menggunakan clean code

class nilaiMhs{

  int x;

  int y;

  public nilaiMhs(int a,int b){

    x=a; y=b;

  }

  int hitung(int z){

    return (x+y+z)/3;

  }

}

public class mainn{

  public static void main(String args[]){

    nilaiMhs nm = new nilaiMhs(80,70);

    int r=nm.hitung(90);

    System.out.println("hasil:"+r);

  }

}

* 1. Analisa code kotor
     1. Nama kelas tidak sesuai konvensi
* nilaiMhs → harusnya pakai PascalCase → NilaiMahasiswa
* main → tidak jelas, harusnya Main atau App
  + 1. Nama variabel tidak jelas
* x, y, z → tidak deskriptif, bikin bingung
* r → tidak jelas maksud nya apa

* + 1. Format kode berantakan
* Spasi , identasi, dan {} tidak rapi
  + 1. Tidak ada JavaDocs
* Tidak menjelaskan apa fungsi class atau method
  1. Clean Code

/\*\*

 \* Class untuk menghitung nilai akhir mahasiswa berdasarkan 3 komponen.

 \*/

class NilaiMahasiswa {

    private int nilaiTugas;

    private int nilaiUts;

    /\*\*

     \* Konstruktor NilaiMahasiswa.

     \*

     \* @param nilaiTugas nilai tugas mahasiswa

     \* @param nilaiUts nilai UTS mahasiswa

     \*/

    public NilaiMahasiswa(int nilaiTugas, int nilaiUts) {

        this.nilaiTugas = nilaiTugas;

        this.nilaiUts = nilaiUts;

    }

    /\*\*

     \* Menghitung nilai akhir berdasarkan nilai tugas, UTS,dan UAS.

     \*

     \* @param nilaiUas nilai UAS mahasiswa

     \* @return nilai akhir mahasiswa

     \*/

    public int hitungNilaiAkhir(int nilaiUas) {

        return (nilaiTugas + nilaiUts + nilaiUas) / 3;

    }

}

public class Main {

    public static void main(String[] args) {

        NilaiMahasiswa mahasiswa = new NilaiMahasiswa(80, 70);

        int nilaiAkhir = mahasiswa.hitungNilaiAkhir(90);

        System.out.println("Nilai Akhir: " + nilaiAkhir);

    }

}

* 1. Analisa Clean Code dengan code yg belum clean code

|  |  |  |
| --- | --- | --- |
| Aspek | Code Kotor | Clean Code |
| Nama Kelas | nilaiMhs, mainn | NilaiMahasiswa, Main |
| Nama Variabel | x, y, z, r | nilaiTugas, nilaiUts, nilaiUas, nilaiAkhir |
| Komentar/Dokumen | Tidak ada | Ada JavaDoc @param, @return |
| Akses Modifier | Default (package-private) | private untuk atribut, public untuk method |
| Format & Spasi | Berantakan | Indentasi konsisten (4 spasi), {} rapi |
| Keterbacaan | Sulit dipahami | Mudah dipahami siapa pun |

* 1. Kesimpulan

Clean code **memperjelas maksud program** tanpa harus membaca detail logika.  
Dengan Java Code Conventions + JavaDoc (@param, @return), kode jadi lebih profesional, mudah dipelihara, dan siap dipakai di tim besar.