LAPORAN PERTEMUAN 2

NAMA : AHMAD FIKRI ZAKARIA
NIM : H1D024062
MATERI : CONSTRUCTOR OVERLOADING & ENCAPSULATION

A. Alur Kerja (Workflow)

Pada pertemuan ini, saya menyelesaikan studi kasus sistem inventaris toko musik "NadaKita". Proses pengerjaan meliputi perbaikan kesalahan sintaks dan implementasi logika pemrograman berorientasi objek sebagai berikut:

1. **Refactoring & Debugging:**
   - Mengubah nama file Barangmusik.Java menjadi BarangMusik.java agar sesuai dengan konvensi penamaan class di Java.
   - Memperbaiki *typo* pada deklarasi class di file UjiBarang.java (dari lass menjadi class).
2. **Implementasi Constructor Overloading:**
   - Membuat tiga jenis konstruktor pada class BarangMusik untuk menangani variasi kelengkapan data saat input barang:
     1. Konstruktor 1: Menerima kode dan nama saja (harga & stok default 0).
     2. Konstruktor 2: Menerima kode, nama, dan harga (stok default 0).
     3. Konstruktor 3: Menerima data lengkap (kode, nama, harga, stok).
3. **Implementasi Encapsulation & Mutator:**
   - Menerapkan method mutator (setter) ubahHarga() untuk memperbarui harga barang.
   - Menerapkan method tambahStok() untuk menambah jumlah stok barang yang ada.
4. **Uji Coba (Testing):**
   - Menjalankan class UjiBarang untuk memverifikasi bahwa objek yang dibuat dengan konstruktor berbeda dapat menyimpan data dengan benar dan method mutator berfungsi sesuai harapan.

B. Penjelasan Kode & Fungsi

Berikut adalah konsep utama yang diterapkan dalam kode program:

1. **Constructor Overloading:** Fitur ini memungkinkan pembuatan objek BarangMusik dengan cara yang fleksibel. Java membedakan ketiga konstruktor tersebut berdasarkan *method signature* (jumlah dan tipe parameter). Ini sangat berguna ketika data awal barang tidak selalu lengkap.
2. **Encapsulation (Pembungkusan Data):** Atribut kodeBarang, namaBarang, harga, dan stok dikelola di dalam class. Akses perubahan data dilakukan melalui method khusus (ubahHarga dan tambahStok), bukan mengakses variabel secara langsung. Hal ini menjaga keamanan data agar perubahan nilai stok atau harga bisa dikontrol logic-nya di masa depan (misalnya mencegah harga negatif).
3. **Logic Mutator:**
   - ubahHarga(double hargaBaru): Mengganti nilai harga lama dengan harga baru.
   - tambahStok(int jumlah): Menggunakan operator += untuk mengakumulasi stok, bukan menimpanya.

C. Hasil Output Program

Berikut adalah tangkapan layar/teks hasil eksekusi program UjiPengguna.java: