saya sebetulnya cukup kebingungan di kode java anda, kenapa anda membuat getter an setternya public

Pertanyaan **bagus banget** 😄 — dan ini sebenarnya menyentuh salah satu prinsip paling penting dalam **pemrograman berorientasi objek (OOP)**, yaitu **enkapsulasi (encapsulation)**.

Kamu benar: di contoh kode Java yang saya tulis sebelumnya, **getter** dan **setter** dibuat dengan **akses public**, dan wajar kamu bertanya *“kenapa nggak dibuat private juga aja sekalian?”*

Yuk kita bahas pelan-pelan biar jelas:

**🧩 1. Tujuan utama getter dan setter**

Getter dan setter itu dibuat untuk:

* **Mengontrol akses** ke data (variabel) yang bersifat private.
* Memberi **jalan yang aman** bagi kode luar (class lain) untuk membaca atau mengubah nilai atribut.

Jadi dalam OOP:

* **Variabel (field)** → biasanya private (disembunyikan)
* **Getter & Setter** → biasanya public (pintu akses yang dikontrol)

Contohnya di Java:

public class Mobil {

private String merk; // disembunyikan dari luar

private int tahun;

// Getter (public) -> untuk membaca nilai merk

public String getMerk() {

return merk;

}

// Setter (public) -> untuk mengubah nilai merk secara aman

public void setMerk(String merk) {

this.merk = merk;

}

}

**⚙️ 2. Kenapa variabelnya private, tapi getter/setter-nya public?**

**Karena konsep enkapsulasi**

Enkapsulasi = menyembunyikan data internal dari akses langsung luar kelas.

**Analogi sederhana:**

Bayangkan sebuah **mesin mobil** 🚗.  
Kamu (pengguna) tidak boleh langsung menyentuh mesin (data internal = private),  
tetapi kamu bisa **menyalakan mobil lewat tombol Start** (getter/setter = public).

💡 Jadi:

* private = data disembunyikan, supaya tidak sembarangan diubah dari luar.
* public getter/setter = cara resmi (aman) untuk membaca atau mengubah data tersebut.

Kalau getter/setter dibuat private, maka:

* Class luar (misalnya Main atau Pemilik) **tidak bisa mengakses data sama sekali**,
* Akibatnya, atribut merk dan tahun jadi “terkunci” di dalam kelas Mobil.

**🚫 3. Apa yang terjadi kalau getter dan setter dibuat private?**

Contoh kesalahan:

public class Mobil {

private String merk;

private int tahun;

private String getMerk() { // private getter

return merk;

}

private void setMerk(String merk) { // private setter

this.merk = merk;

}

}

Lalu di kelas lain:

public class Main {

public static void main(String[] args) {

Mobil m = new Mobil();

m.setMerk("Toyota"); // ❌ ERROR: setMerk() has private access

}

}

❌ Error karena method setMerk() bersifat private,  
artinya **tidak bisa diakses dari luar kelas Mobil**.

**✅ 4. Jadi, kapan sebaiknya getter dan setter dibuat public?**

Biasanya:

* **public** kalau data itu perlu dibaca/diubah dari luar kelas (seperti di Main atau Pemilik).
* **private atau protected** kalau data hanya boleh diubah dari dalam kelas itu sendiri (misalnya logika internal saja).

**🔒 5. Bisa juga dikontrol: hanya getter publik, setter tidak**

Kadang kamu ingin **data bisa dibaca tapi tidak bisa diubah** dari luar.  
Maka:

* Getter → public
* Setter → private (atau tidak dibuat sama sekali)

Contoh:

public class Mobil {

private String merk;

private final int tahun; // misalnya tahun mobil tidak boleh diubah

public Mobil(String merk, int tahun) {

this.merk = merk;

this.tahun = tahun;

}

public String getMerk() {

return merk;

}

// setter tetap boleh

public void setMerk(String merk) {

this.merk = merk;

}

// getter untuk tahun (tanpa setter)

public int getTahun() {

return tahun;

}

}

Artinya:

* Dari luar kelas, kita bisa baca getTahun(), tapi **tidak bisa ubah tahun**.
* Hanya merk yang bisa diubah lewat setMerk().

**💬 6. Kesimpulan**

| **Elemen** | **Akses Umum** | **Tujuan** |
| --- | --- | --- |
| **Field (variabel)** | private | Melindungi data dari akses langsung luar kelas |
| **Getter** | public | Mengizinkan pembacaan data dengan cara yang aman |
| **Setter** | public | Mengizinkan pengubahan data dengan validasi |
| **Getter/Setter private** | Hanya dipakai di dalam kelas itu sendiri (jarang digunakan) |  |

🧠 **Intinya:**  
Getter dan setter dibuat public karena mereka adalah **“pintu resmi”** untuk keluar-masuk data dari kelas.  
Sedangkan atribut dibuat private agar **tidak bisa diakses langsung** tanpa izin.

Kalau kamu mau, aku bisa tunjukkan contoh **versi getter/setter dengan validasi** (misalnya tidak boleh isi kosong atau tahun lebih besar dari tahun sekarang), supaya kamu bisa lihat kenapa kontrol lewat public method itu berguna banget.  
Apakah mau saya tambahkan contohnya?