# Tugas Pendahuluan: Modul 9

**NIM: 105223002** **Nama: Gerald Eberhard**

**Instruksi:** Jawablah pertanyaan-pertanyaan berikut dengan jelas dan rinci. Gunakan contoh kode jika diperlukan untuk memperjelas jawaban Anda. Kumpulkan dengan format **TP9_NIM_PrakPBO.pdf.** Deadline, Kamis 22 Mei 2025, pukul 13:00 (sebelum praktikum). Link pengumpulan

## Soal

1. Apa gunanya anotasi @Override dalam Java? Apa yang terjadi jika kita lupa menuliskannya saat melakukan overriding?
   **Jawab:**

   Anotasi @Override digunakan untuk menandai bahwa sebuah method menggantikan (override) method dari superclass atau interface. Manfaatnya: memastikan method benar-benar meng-override method yang ada (mencegah kesalahan penulisan nama, parameter, atau tipe return). Kompiler akan memeriksa kesesuaian method.

   Jika kita lupa menuliskan @Override, method tetap bisa meng-override selama tanda tangan method (nama, parameter, tipe return) sama dengan method di superclass. Namun, tanpa anotasi, kompiler tidak akan mendeteksi kesalahan jika method tidak benar-benar meng-override (misalnya, karena salah nama atau parameter), sehingga dapat menyebabkan bug logika.

2. Dapatkah constructor dioverride dalam Java? Jelaskan alasan teknisnya.
   **Jawab:**

   Constructor tidak dapat dioverride di Jawa. Hal ini itu dikarenakan:
   1. Constructor bersifat spesifik untuk kelas tertentu dan tidak diwarisi oleh subclass, sehingga tidak ada method constructor di subclass yang bisa diganti (override).
   2. Overriding berlaku untuk method instance, sedangkan constructor bukan method, melainkan blok khusus untuk inisialisasi objek.
      Namun, subclass dapat memanggil constructor superclass menggunakan super() untuk memperluas inisialisasi.

3. Jelaskan bagaimana method super digunakan dalam konteks method overriding!
   **Jawab:**

   Kata kunci super digunakan untuk memanggil method superclass yang telah dioverride oleh subclass. Dalam konteks overriding, contohnya seperti super.methodName() memungkinkan subclass menjalankan implementasi method dari superclass sebelum atau setelah menambahkan logika spesifik di subclass. Hal

ini berguna untuk mempertahankan fungsionalitas superclass sambil menambahkan perilaku baru.

4. Sebutkan dan jelaskan dua jenis polymorphism dalam Java beserta waktu terjadinya
**Jawab:**

1. Compile-time Polymorphism (Static Polymorphism):
   - Terjadi saat kompilasi, melalui method overloading atau operator overloading (meski Java tidak mendukung operator overloading secara langsung).
   - Contoh: Method dengan nama sama tetapi parameter berbeda dipilih oleh kompiler berdasarkan tanda tangan method.
   - Waktu: Saat kode dikompilasi.

2. Runtime Polymorphism (Dynamic Polymorphism):
   - Terjadi saat runtime, melalui method overriding menggunakan inheritance dan reference type.
   - Contoh: Method yang dioverride di subclass dipanggil berdasarkan tipe objek aktual, bukan tipe referensi.
   - Waktu: Saat program dijalankan (dengan mekanisme dynamic method dispatch).

5. Bagaimana polymorphism mendukung prinsip abstraction, flexibility, dan code reusability dalam pemrograman berorientasi objek?
**Jawab:**

- Abstraction: Polymorphism memungkinkan penggunaan tipe umum (superclass/interface) untuk merujuk ke objek spesifik (subclass), menyembunyikan detail implementasi. Contoh: Menggunakan interface List untuk ArrayList atau LinkedList.

- Flexibility: Polymorphism memungkinkan sistem menangani objek dari berbagai subclass secara seragam melalui referensi superclass, memudahkan penambahan tipe baru tanpa mengubah kode yang ada.

- Code Reusability: Method di superclass dapat digunakan kembali oleh subclass, dan overriding memungkinkan penyesuaian perilaku tanpa menduplikasi kode.

**Referensi**


[1]. Belajar Java OOP: Memahami Inheritance dan Method Overriding
[2]. Belajar Java OOP: Memahami Prinsip Polimorfisme dalam OOP
[3]. Polimorfisme di Java: Konsep, Implementasi, dan Keuntungannya - Pemburu Kode