Tugas Pendahuluan: Modul 9

NIM: 105222033 **Nama:Mohamad Dandung**

Soal

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

Jawab:

Anotasi @Override dalam Java berfungsi sebagai penanda bagi compiler bahwa sebuah method dimaksudkan untuk meng-override method dari superclass atau mengimplementasikan method dari sebuah interface. Tujuannya adalah untuk membantu deteksi kesalahan: jika Anda lupa menuliskannya dan kemudian membuat kesalahan penulisan (typo) pada nama method atau parameter, sehingga tanda tangan method tidak sama persis dengan method di superclass, compiler tidak akan menganggapnya sebagai overriding.

2. Dapatkah constructor dioverride dalam Java? Jelaskan alasan teknisnya.

Jawab:

Constructor tidak dapat dioverride dalam Java. Alasan teknisnya adalah bahwa constructor tidak diwarisi oleh subclass; setiap subclass memiliki constructornya sendiri untuk menginisialisasi objeknya.

3. Jelaskan bagaimana method super digunakan dalam konteks method overriding!

Jawab:

Dalam konteks method overriding, kata kunci super digunakan untuk memanggil dan menjalankan implementasi method yang sama dari superclass (kelas induk) yang telah di-override di subclass

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

Jawab:

Ada dua jenis polymorphism dalam Java: *Compile-time Polymorphism* (juga dikenal sebagai Static Polymorphism atau Method Overloading) dan *Runtime Polymorphism* (juga dikenal sebagai Dynamic Polymorphism atau Method Overriding).

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

Jawab:

Polymorphism sangat mendukung prinsip abstraction, flexibility, dan code reusability dalam pemrograman berorientasi objek. Dalam hal abstraction, polymorphism memungkinkan kita untuk berinteraksi dengan berbagai jenis objek melalui antarmuka umum (interface atau superclass), memungkinkan kita untuk fokus pada "apa" yang dapat dilakukan objek tanpa harus peduli "bagaimana" detail implementasinya di setiap subclass.