Nama : Alan Aditya

NIM : 2304130087

Rombel : 2 Teknik Informatika

**PRINSIP-PRINSIP YANG HARUS DIJAGA SEORANG PENGEMBANG PERANGKAT LUNAK**

1. **KISS (Keep It Simple, Stupid):** Prinsip ini menekankan untuk menjaga kode tetap sederhana dan tidak rumit. Kode yang sederhana lebih mudah dipahami, di-debug, dan dimodifikasi oleh developer lain. Tujuannya adalah menghindari kompleksitas yang tidak perlu dan hanya menggunakan apa yang dibutuhkan.
2. **DRY (Don't Repeat Yourself):** DRY menganjurkan untuk menghindari pengulangan kode. Kode yang sama seharusnya hanya ditulis sekali dan dapat digunakan kembali. Ini mengurangi panjang kode, memudahkan pemeliharaan (perubahan hanya perlu dilakukan di satu tempat), dan menghindari inkonsistensi. Lawan dari DRY adalah WET ("write everything twice" atau "waste everyone’s time").
3. **YAGNI (You Aren't Gonna Need It):** Prinsip ini menyarankan untuk tidak mengimplementasikan sesuatu sampai benar-benar dibutuhkan. Terlalu sering, programmer menambahkan kode untuk fitur yang mungkin diperlukan di masa depan, padahal belum tentu. Ini bisa membuat kode menjadi besar dan kompleks tanpa alasan. Fokuslah pada apa yang dibutuhkan saat ini.
4. **SOLID:** SOLID adalah akronim dari lima prinsip desain berorientasi objek:
   * **Single Responsibility Principle (SRP):** Setiap kelas harus memiliki satu tanggung jawab.
   * **Open/Closed Principle:** Entitas perangkat lunak (kelas, modul, fungsi, dll.) harus terbuka untuk perluasan, tetapi tertutup untuk modifikasi.
   * **Liskov's Substitution Principle (LSP):** Objek dari kelas turunan harus dapat menggantikan objek dari kelas induk tanpa mengubah kebenaran program.
   * **Interface Segregation Principle (ISP):** Antarmuka yang spesifik lebih baik daripada antarmuka yang umum.
   * **Dependency Inversion Principle (DIP):** Modul tingkat tinggi tidak boleh bergantung pada modul tingkat rendah. Keduanya harus bergantung pada abstraksi. Abstraksi tidak boleh bergantung pada detail, detail harus bergantung pada abstraksi.
5. **Separation of Concerns (SoC):** SoC memisahkan aplikasi yang kompleks menjadi bagian-bagian atau domain yang berbeda, di mana setiap bagian memiliki tugas atau tanggung jawab spesifik. Setiap bagian independen satu sama lain, sehingga memudahkan pemeliharaan, pembaruan, dan penggunaan kembali kode. Contohnya adalah pola MVC (Model-View-Controller).
6. **Avoid Premature Optimization:** Optimasi memang penting, tetapi jangan dilakukan terlalu dini. Optimasi prematur dapat membuat kode sulit dipahami dan dipelihara, dan seringkali tidak efektif karena kita belum tahu di mana bottleneck program sebenarnya. Optimasi sebaiknya dilakukan setelah kode berfungsi dengan benar dan diidentifikasi area mana yang perlu ditingkatkan.
7. **Law of Demeter:** Prinsip ini, juga dikenal sebagai prinsip pengetahuan paling sedikit, membatasi interaksi antar objek. Setiap unit (objek) hanya boleh memiliki pengetahuan terbatas tentang unit lain, yaitu hanya unit yang "berhubungan dekat" dengannya. Setiap unit hanya boleh berbicara dengan "teman"-nya, bukan dengan "orang asing". Tujuannya adalah untuk menjaga kelas tetap independen dan mengurangi ketergantungan.