Erdianti Wiga Putri Andini

13522053 – K1

TUGAS BACA

SECTION 1.1 – 1.3 HOLGER GAST

1.1. The Core: Objects as Small and Active Entities

Pembahasan menguraikan prinsip-prinsip inti dari pemrograman berorientasi objek (OOP), menekankan keutamaan objek dibandingkan kelas dan pentingnya memahami perilaku dan kolaborasi. OOP menganjurkan penggunaan banyak objek kecil yang berkolaborasi untuk menangani tugas-tugas kompleks, meninggalkan struktur monolitik dari pemrograman prosedural. Objek dikarakterisasi sebagai entitas yang ringan, aktif, dan terenkapsulasi, menyajikan desain modular dengan antarmuka yang jelas untuk meningkatkan keberlanjutan kode. Selain itu, OOP mendorong pola pikir kolaboratif, memperlakukan objek sebagai pemain tim yang mampu delegasi tugas dan berkomunikasi melalui pertukaran pesan, sehingga mendorong arsitektur kode yang dapat digunakan kembali dan modular.

Pembahasan mengatasi kekhawatiran efisiensi dalam OOP, menganjurkan pendekatan seimbang antara usaha pengembang dan kinerja saat runtime. Ini menyarankan untuk memusatkan upaya optimisasi pada titik-titik bottleneck yang diidentifikasi melalui bukti empiris, seperti profil, dan menekankan peran enkapsulasi dalam lokalisisasi optimisasi dalam kelas-kelas tertentu. Pembahasan ini menekankan pentingnya memahami perilaku objek, merancang untuk kolaborasi, dan menjaga sikap pragmatis terhadap efisiensi dalam pengembangan berorientasi objek untuk memajukan kode yang lebih jelas, dapat dipertahankan, dan efisien.

1.2. Developing with Objects

Membahas mengenai pentingnya praktik pengembangan perangkat lunak yang efisien, terutama fokus pada penggunaan alat seperti Eclipse untuk menyederhanakan tugas pemrograman dan meminimalkan upaya manual. Dengan menekankan peningkatan kode yang kontinu melalui pemformatan, mengorganisir impor, dan mematuhi konvensi kode, pengembang dapat mempertahankan konsistensi dan keberbacaan dalam basis kode mereka. Pendekatan ini memungkinkan pengembang untuk lebih banyak menghabiskan waktu untuk pemecahan masalah dan kreativitas, daripada terbebani oleh tugas rutin.

Konsep refactoring sebagai praktik penting untuk meningkatkan struktur perangkat lunak sambil mempertahankan fungsionalitas. Ini menyoroti sifat iteratif pengembangan, di mana wawasan yang diperoleh selama pemrograman mengarah pada penyesuaian desain. Refactoring disajikan sebagai strategi kunci untuk meningkatkan kemudahan pemeliharaan kode, keberbacaan, dan akhirnya mengurangi biaya produksi. Dengan merangkul refactoring, pengembang dapat beradaptasi dengan persyaratan yang berkembang dan meningkatkan kualitas keseluruhan kode mereka.

Membahas peran penting penamaan dalam pengembangan perangkat lunak, menekankan dampak nama yang dipilih dengan baik pada pemahaman dan kemudahan pemeliharaan kode. Hal ini memberikan pedoman untuk memilih nama yang deskriptif yang mempertimbangkan konteks, keberbacaan, dan menghindari ambiguitas. Memanfaatkan alat seperti Eclipse untuk mengubah nama memastikan konsistensi dan keberbacaan kode. Secara keseluruhan, teks menganjurkan nama-nama yang memfasilitasi pemahaman kode dan penceritaan, sambil memperingatkan tentang pilihan yang dapat membingungkan atau menyesatkan pembaca, yang pada akhirnya berkontribusi pada praktik pengembangan perangkat lunak yang lebih baik.

1.3. Fields

Dalam pemrograman berorientasi objek, fields memainkan peran penting dalam menyimpan informasi yang dioperasikan oleh objek. Fields ini membentuk inti dari sebuah objek, mengandung data dari mana objek menghitung jawaban untuk panggilan metode dan membuat keputusan. Namun, dari sudut pandang desain sistem, detail-detail fields objek biasanya dijaga privat, tersembunyi dari objek lain. Penyembunyian ini memastikan bahwa objek lain tidak dapat membuat asumsi tentang keberadaan atau konten dari fields ini. Fields diinisialisasi saat objek dibuat dan mempertahankan maknanya sepanjang masa hidup objek. Penting untuk memahami dan memelihara konten masing-masing fields serta hubungannya dengan fields lain untuk memastikan perilaku yang konsisten.

Objek sering menggunakan fields untuk mempertahankan dan mengatur data mereka, membentuk struktur data yang memungkinkan operasi yang efisien. Struktur ini dapat bervariasi mulai dari array sederhana hingga pengaturan yang lebih kompleks seperti linked list atau hash map. Fields dalam struktur data berfungsi sebagai wadah pasif untuk informasi, dengan antarmuka objek menyediakan akses yang disederhanakan untuk memanipulasi data yang mendasarinya. Selain itu, objek sering bekerja sama satu sama lain, mempercayakan tanggung jawab tertentu kepada rekan-rekannya. Rekan-rekan sering disimpan dalam fields, memungkinkan objek untuk merujuk pada mereka sepanjang masa hidupnya dan menugaskan tugas secara efektif.

Fields juga dapat mewakili properti objek, menyediakan akses seperti getter dan setter untuk klien berinteraksi dengan data tertentu yang dipegang oleh objek. Properti memungkinkan objek untuk memegang dan mengelola informasi yang penting untuk fungsinya, membuatnya dapat diakses oleh klien sesuai kebutuhan. Selain itu, fields juga dapat mencakup properti konfigurasi atau flag yang mempengaruhi perilaku atau keputusan sebuah objek. Cache, data bersama antara metode, dan fields statis adalah aspek lain dari fields yang dapat memengaruhi desain dan fungsionalitas sebuah objek, masing-masing memiliki tujuan yang berbeda dalam pemrograman berorientasi objek.