Bab 2
Java dan Pola Pikir Obyek

Bab ini membahas tentang gambaran umum pola pikir yang dibutuhkan saat mendalami peranti pengembangan Java. Pada dasarnya ada pola pikir prosedural serta obyek dalam penyelesaian masalah pemrograman. Bab ini akan membahas keduanya serta perbedaannya.

2.1 Pemrograman dan Pola Pikir Penyelesaian Masalah

Secara sederhana, bisa dikatakan bahwa pemrograman merupakan aktivitas yang digunakan untuk membuat peranti lunak ("software"). Membuat peranti lunak bukan merupakan hal yang mudah, tetapi bisa dipelajari dan dilaksanakan. Perlu dipahami sejak awal bahwa pemrograman biasanya bertujuan untuk menghasilkan suatu peranti lunak yang bisa dijalankan oleh pemakai komputer biasa. Pemrograman bukan merupakan satu-satunya aktivitas yang dilaksanakan dalam menghasilkan suatu peranti lunak. Untuk pembahasan saat ini, yang perlu diketahui adalah, pemrograman merupakan suatu bagian dar proses pembuatan peranti lunak, bukan satu-satunya proses dan tidak bisa dikatakan bahwa pemrograman adalah fase yang paling penting. Materi ini akan kita bahas secara lebih mendalam pada saat kita membahas tentang metodologi pengembangan peranti lunak.

Proses pembuatan peranti lunak biasanya mengandung dua aspek:

1. Aspek teknis, yaitu sisi yang terkait dengan komputer dan peranti lunak. Teknik pemrograman (membuat antarmuka, implementasi rancangan basis data, penggunaan pustaka tertentu, dan lain-lain) merupakan aspek yang secara langsung berurusan dengan teknis.
2. Aspek non teknis, yaitu sisi yang tidak terkait secara langsung dengan komputer dan peranti lunak, biasanya lebih ke arah *inter human relationship* atau hubungan antar manusia dan antar peran yang berada pada pekerjaan pembuatan peranti lunak tersebut. Salah satu contoh sederhana dari sisi ini adalah sisi penetapan kebutuhan sistem / pemakai yang menentukan ruang lingkup ataupun fitur-fitur peranti lunak yang dibuat (fase tersebut sering disebut dengan *requirements engineering*).

Kedua aspek tersebut merupakan aspek yang menyatu dan menentukan tingkat keberhasilan pembuatan suatu peranti lunak.

Dalam hal teknis pemrograman, pada dasarnya ada beberapa pola pikir atau cara pandang dalam penyelesaian masalah atau seringkali disebut dengan paradigma pemrograman. Paradigma pemrograman menentukan bagaimana seorang pemrogram berpikir dalam menyelesaikan suatu masalah. Banyak kegagalan belajar pemrograman yang berasal dari ketidakpahaman hal tersebut. Secara umum, ada beberapa paradigma pemrograman, misalnya paradigma prosedural, paradigma obyek, paradigma fungsional, paradigma pemrograman logika, dan masih banyak lagi. Bab ini akan membahas dua paradigma yang pada umumnya ada di industri saat ini (meskipun demikian, tidak berarti paradigma pemrograman lainnya tidak digunakan di industri, tetapi penggunaannya masih relatif kurang umum dan sangat spesifik).

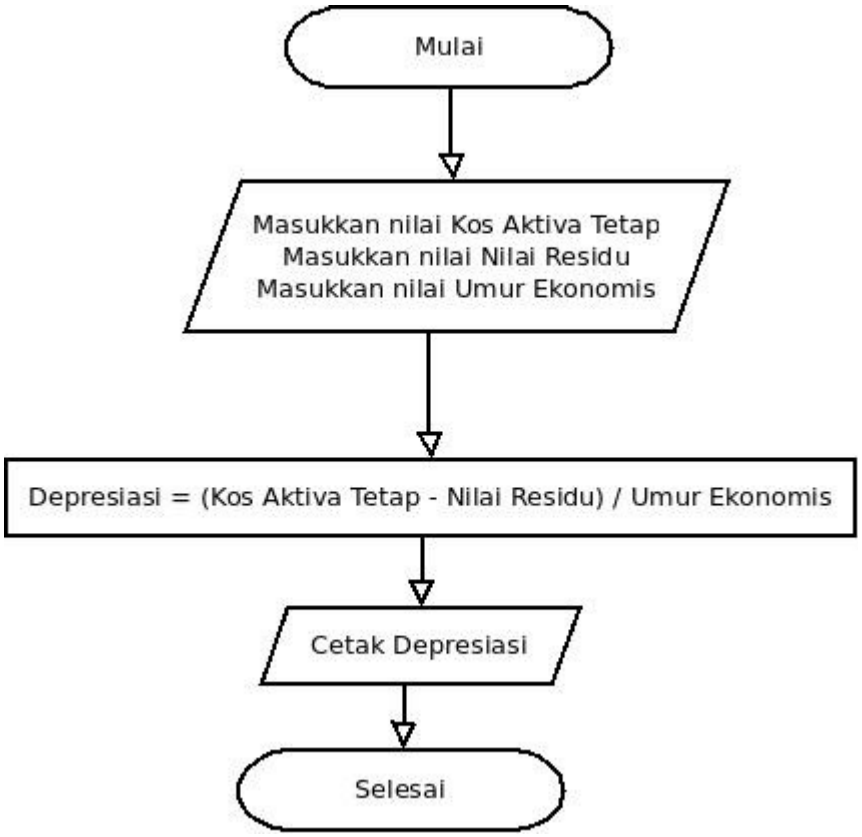
2.1.1 Pola Pikir Prosedural

Pola pikir prosedural merupakan pola pikir yang mengutamakan langkah-langkah prosedural dalam penyelesaian suatu masalah pemrograman. Langkah-langkah tersebut disebut dengan **algoritma**. Dengan menggunakan pola pikir ini, secara sederhana, penyelesaian masalah pemrograman dilaksanakan dengan langkah berikut:

1. Tetapkan tujuan akhir dari peranti lunak (misalnya “meminta masukan pemakai terhadap nilai aktiva tetap dan menghitung nilai depresiasi aktiva per periode kemudian menampilkan hasilnya di layar”).
2. Melakukan analisis masukan-masukan yang diperlukan untuk keperluan langkah pertama tersebut (untuk kasus di atas, masukan yang diperlukan dari pemakai adalah kos aktiva tetap, nilai residu, dan umur ekonomis aktiva tetap). Setelah itu menuliskan program untuk meminta masukan-masukan tersebut.
3. Melakukan analisis proses yang diperlukan (untuk kasus di atas adalah **depresiasi = (kos aktiva tetap – nilai residu) / umur ekonomis**). Setelah itu menuliskan program yang mengimplementasikan perumusan tersebut.
4. Memproses keluaran yang diinginkan (dalam kasus di atas, yang perlu dilakukan adalah mengambil hasil proses dan kemudian menuliskan hasilnya ke layar).

Tentu saja, langkah dan contoh di atas merupakan contoh yang sangat sederhana. Untuk proyek pengembangan peranti lunak yang besar, tentu usaha yang dilakukan lebih dari hal-hal di atas (misalnya penyimpanan data serta pengambilan data dari peranti lunak basis data, memproses masukan berupa *touch screen*, dan lain-lain). Meskipun demikian, setidaknya contoh di atas bisa memberikan gambaran umum pola pikir prosedural dalam pemrograman.

Algoritma pemrograman secara prosedural ini biasanya digambarkan dengan menggunakan suatu bagan alir (*flowchart*) atau *pseudo code* atau *structured english*. Dalam pengembangan peranti lunak yang besar, seringkali diperlukan diagram-diagram dan peranti lainnya seperti *system flowchart*, DAD (Diagram Alir Data / *Data Flow Diagram* / *DFD*), E-R Diagram, *Data Dictionary*, dan lain-lain. Berikut adalah contoh bagan alir untuk algoritma program di atas:

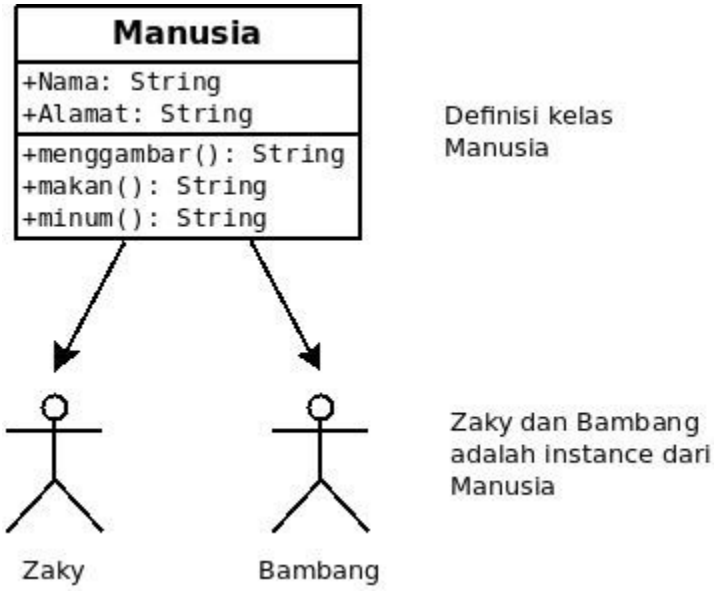


Gambar 2.1: Bagan alir mencari nilai depresiasi aktiva tetap

Pola pikir prosedur sering dikritisi mempunyai kekurangan dalam hal *reusable components* artinya pola pikir prosedural seringkali menggunakan langkah-langkah spesifik, sehingga jika terdapat masalah yang mirip, tidak mudah untuk mengambil bagian dari program yang sudah pernah dibuat oleh pemrogram. Hal ini sebenarnya merupakan hal yang kurang begitu tepat karena kebanyakan peranti pemrograman prosedural sudah menyediakan fasilitas untuk pembuatan pustaka (*library*) yang berisi berbagai prosedur yang dirancang cukup umum sehingga bisa digunakan di proyek pengembangan software lainnya. Meskipun demikian, prosedur tidak bisa diperluas (di-*extend*), dan ini merupakan kelemahan dari pola pikir prosedural.

2.1.2 Pola Pikir Obyek

Sama halnya dengan pola pikir prosedural, pola pikir obyek ini juga dimulai dengan perumusan masalah yang akan diselesaikan oleh peranti lunak yang akan dibuat. Hal yang membedakan terletak pada pola pikir penyelesaian masalah. Dengan menggunakan pola pikir obyek, seorang pemrogram harus mengenali berbagai komponen obyek yang membentuk suatu sistem dan bagaimana setiap obyek tersebut bertindak dan berinteraksi dengan obyek lainnya untuk menyelesaikan masalah. Obyek-obyek tersebut seringkali disebut juga dengan *instance*, dibuat dari kelas yang sudah tersedia atau jika belum terdapat kelas yang sudah ada, maka tugas pemrogram membuat kelas yang bersangkutan kemudian baru membuat suatu *instance* berdasarkan kelas yang ada atau yang telah dibuat tersebut. Secara sederhana, hubungan antara kelas dengan obyek (*instance*) adalah sebagai berikut:

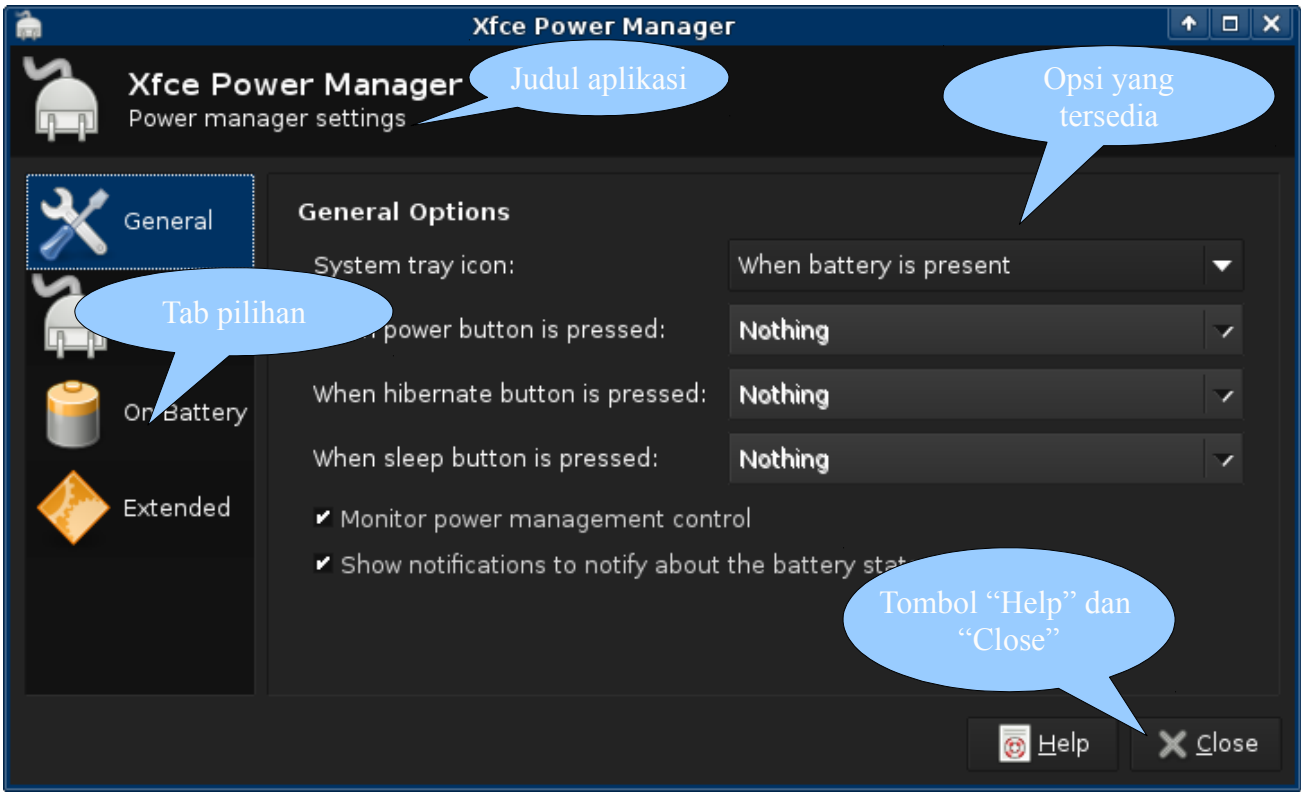


Gambar 2.2: Kelas dan *instance*/obyek

Kelas adalah cetak biru dari obyek / *instance*, setiap kali dibuat obyek, maka obyek yang dibuat tersebut akan mempunyai sifat / atribut serta bisa beraksi seperti yang telah di definisikan pada kelas tersebut. Jadi, jika dilihat pada gambar 2.2 tersebut, Zaky maupun Bambang mempunyai Nama dan Alamat (yang mungkin masing-masing berbeda), serta bisa mempunyai aksi menggambar, makan, dan minum (dan yang digambar, dimakan, serta

diminum masing-masing obyek berlainan).

Dalam kasus peranti lunak dengan komputer, seorang pemrogram harus bisa memahami obyek apa saja yang mungkin terdapat pada sistem. Sebagai contoh, saat membuat peranti lunak “XFCE Power Manager”, pemrogram seharusnya sudah memahami bahwa kemungkinan obyek yang akan terdapat dalam aplikasi adalah sebagai berikut:



Gambar 3.3: Komponen obyek dalam suatu aplikasi

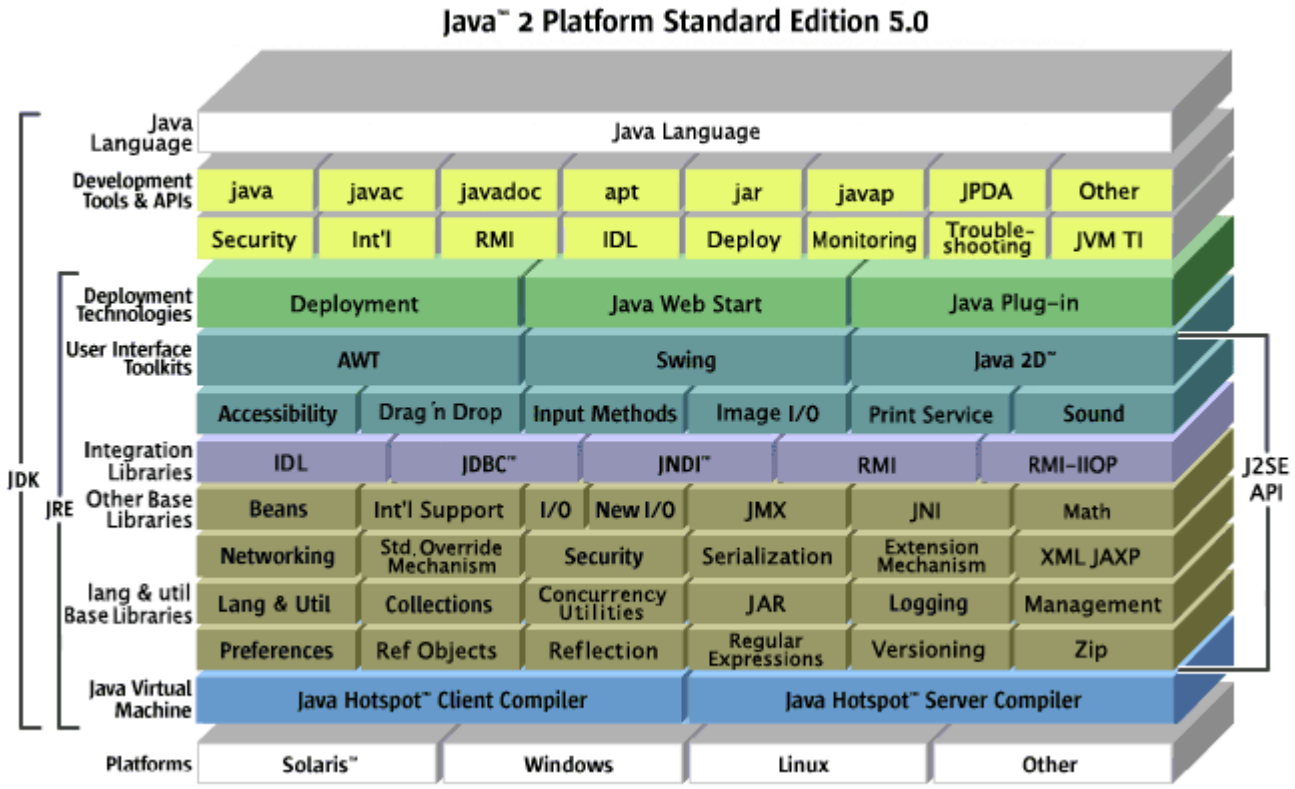
Sebagai contoh pola pikir obyek, tombol “Help” dan “Close” merupakan *instance*/obyek dari kelas **tombol**. Masing-masing tombol tersebut mempunyai aksi yang berbeda-beda jika diklik menggunakan mouse.

2.2 Algoritma Pemrograman dan PBO

PBO mempunyai suatu karakteristik yang disebut dengan pengkapsulan (*encapsulation*). Pengkapsulan merupakan fitur dari PBO yang menyatukan dan menyembunyikan atribut serta *method* dari akses publik (sama halnya dengan kap mesin mobil yang mengenkapsulasi berbagai peranti di dalam mesin mobil dan hanya bisa diakses dengan mekanisme gas, rem, persnelling, dan kopling untuk menjalankan mobil). Meskipun dalam pembahasan di depan, algoritma merupakan fitur dari pola pikir prosedural, tetapi setiap definisi kelas tetap memerlukan algoritma dalam setiap *method* yang terdapat dalam kelas tersebut. Apapun yang terjadi, pola pikir algoritmis ataupun prosedural dalam penyelesaian masalah tetap tidak bisa dihindari. Hal yang perlu dilakukan dalam pola pikir obyek adalah mengusahakan agar setiap kelas bisa didefinisikan seumum mungkin, dan jika terjadi kebutuhan yang lebih spesifik baru dibuat kelas anak (proses ini disebut dengan *inheritance* atau pewarisan).

2.3 Java Sebagai Bahasa Pemrograman Berorientasi Obyek

Java merupakan suatu bahasa pemrograman yang mendukung paradigma pemrograman berorientasi obyek (PBO). Dengan demikian, bangunan utama dari Java adalah kelas (*class*) yang secara minimal terdiri atas atribut (*property*) serta perilaku atau aksi (*action* / *method*). Sebagai bahasa yang mendukung paradigma PBO, selain berbagai sintaksis standar untuk keperluan pemrograman, Java juga menyediakan berbagai kelas standar (sering disebut *standard library* atau *builtin classes*) yang bisa secara langsung digunakan. Berikut ini adalah gambar berbagai *builtin classes* yang terdapat dalam Java versi standar:



Gambar 3.4: Berbagai *builtin classes* di Java 5.