BAB I
PENDAHULUAN

A. Latar Belakang Masalah

Dewasa ini pemograman berorientasi objek / *Object Oriented Programming* (OOP) telah marak digunakan, salah satu yang terkenal adalah bahasa pemograman JAVA yaitu pemograman berorientasi obyek murni. Pemograman menggunakan metode ini mempermudah kalaborasi antar *programmer* dalam satu tim dan dengan metode ini pula seorang *programmer* dapat menggunakan *class* (produk yang dihasilkan dalam pemogramman OOP) programmer lain dari belahan dunia lain (tidak dalam satu tim/perusahaan, tidak dalam satu wilayah dan bahkan tidak saling kenal). Hal ini dikarenakan oleh *class* yang dibuat. Class dalam hal ini adalah sebuah file yang berisi suatu kesatuan proses-proses yang terbungkus (enkapsulasi/salah satu ciri dari OOP) yang diberi nama tertentu (untuk bahasa pemograman JAVA, nama class sama dengan nama file sedangkan untuk bahasa pemograman PHP tidak dan isi file boleh lebih dari satu *class*) misal class Database yang isinya berisikan proses-proses yang ada di dalam satu database seperti koneksi, queri eksekusi/ eksekusi perintah, pengambilan hasil dari queri. *Class-class* yang dibuat digabung sehingga menjadi produk. Misalnya di dalam satu tim ada 3 *programmer*, pekerjaan dibagi menjadi tiga bagian dalam contoh kali ini menggunakan pola MVC (*Model View Controller*/salah satu pola dalam OOP). Programmer pertama membuat *class-class* bagian *model* kemudian programmer yang lainnya membuat bagian *view* dan *controller*. Untuk membuat suatu produk utuh, yang perlu dilakukan hanya menggabungkan class-class yang mereka miliki dan apabila ingin mengembangkan produk, yang dilakukan hanya perlu menggunakan ulang *class-class* yang dibuat dan

menggembangkannya. Sedangkan penggunaan *class* dari *programmer* lain dapat diambil contoh pada penggunaan *class calendar*. Seorang *programmer* yang membuat produk yang menggunakan kalender misalnya dalam sebuah program laporan berbasis web, *programmer* ini tidak perlu membuat *class calendar* lagi karena sudah banyak *class calendar* yang bertebaran di Internet yang dibuat oleh *programmer* lain dan yang perlu dilakukannya adalah mempelajari cara penggunaan *class*-nya (*class calendar*) saja atau mengembangkan class tersebut sesuai dengan kebutuhan tergantung syarat yang diberikan oleh pembuatnya. Dari beberapa contoh di atas dapat di ambil gambaran bahwa *class* dibuat dan dikembangkan sesuai dengan kebutuhan, *class-class* dapat digabungkan dengan suatu produk atau membentuk satu produk. *Class-class* dapat digunakan ulang (*reuse class*) dan dapat dikembangkan menjadi lebih kompleks.

Sebagai manusia tidak luput dari yang namanya lupa, karena banyaknya *class* yang dibuat kadang-kadang suatu *class* (yang pernah dibuat sebelumnya) dapat terlupakan bagaimana fungsi dari *class* yang pernah dibuat, bagaimana membuat, struktur dan kegunaan *class*. Hal-hal semacam ini mungkin saja akan terlupakan sehingga hal ini dapat menghambat dalam pengembangan lebih lanjut karena harus mempelajari dari awal lagi. Begitu juga dengan *class* yang dibuat *programmer* lain, apabila ingin mempelajari dan mengembangkannya, perlu dibaca keseluruhan dari isi *class* tersebut sehingga ini akan menghabiskan banyak waktu. Masalah tersebut dapat ditanggulangi dengan membuat pembaca *class* secara otomatis dengan memilah *class* menjadi komponen-komponen yang mudah untuk dipelajari, dimanipulasi, dinavigasi dan dokumentasi. Dokumentasi terkadang menjadi hal yang membosankan dan merepotkan bagi seorang *programmer* yang terkadang lebih senang membuat program daripada mendokumentasikannya (yang terkadang menjadi hal yang

mebghabiskan waktu saja). *Programmer* lebih senang memberi komentar di dalam program mereka daripada menuliskannya di tempat terpisah seperti dokumentasi. Untuk masalah inilah diperlukan suatu pembangkit dokumentasi secara otomatis yang dibangkitkan dari kode program sebuah class dengan mengandalkan komentar-komentar yang biasanya ditulis oleh seorang programmer di dalam *class*-nya.

Adapun analisa yang akan digunakan adalah analisa PIECES yang terdiri dari

1. Analisa Kinerja
2. Analisa Informasi
3. Analisa Ekonomi
4. Analisa Pengendali
5. Analisa Efisiensi
6. Analisa Pelayanan

B. Rumusan Masalah

Adapun rumusan masalah yang dibahas pada skripsi ini adalah sebagai berikut:

1. Dapatkah sebuah *class* dari PHP(file yang berisikan *class* yang ditulis dalam bahasa PHP) dipilah menjadi komponen-komponen yang dapat dengan mudah dipelajari, dimanipulasi dan dinavigasi.
2. Dapatkah suatu *class* dari PHP(file yang berisikan *class* yang ditulis dalam bahasa PHP) di dokumentasikan secara otomatis dan dapat dimanipulasi sesuai kebutuhan.

C. Batasan Masalah

Pembahasan di dalam tulisan ini membahas seputar pemograman dari segi analisis, desain, pembahasan dan implementasi program pembaca dan "pembangkit dokumentasi' *class* untuk PHP. Adapun secara detail batasan masalahnya adalah sebagai berikut:

1. Program pembaca class (*class reader*) yang dibuat hanya membaca dengan baik file yang berisikan *class* yang sudah memenuhi aturan penulisan class yang sudah terhindar dari *sintaks error* (tidak termasuk *sintaks error* karena masalah depedensi terhadap suatu *class* lain). Untuk file yang masih mengandung *sintaks error*, tidak dijamin ketepatan pembacaannya. Adapun dalam aturan penulisan class terdapat aturan yang dipatuhi (agar bisa dibaca dengan baik oleh *class reader*) yaitu bagian posisi penulisan komentar pada atau untuk fungsi/*method*. Untuk aturan penulisan disertakan di dalam lampiran
2. Tidak ada pengecekan dependensi antara class (*class* yang memiliki ketergantungan akan *class* lain untuk dapat berjalan dengan baik). Pembacaan dilakukan terhadap file secara independensi.
3. Pada pembuatan program class reader lebih ditekankan pembuatannya (dalam memilah) menggunakan REGEX dan manipulasi string pada PHP
4. Bahasa pemograman yang digunakan adalah bahasa pemograman PHP versi 5. Sistem Operasi yang digunakan dalam percobaan kali ini adalah GNU/Linux.
5. Fitur program *class reader* adalah:
   a. Membuat komponen-komponen sebagai berikut: nama *class,* nama *superclass* (*class* orang tua / *parent*) jika ada, nama fungsi-fungsi /*method-method* dalam *class* (sesuai dengan *class*-nya masing-masing), suatu Array yang menyimpan fungsi-fungsi, nama *temporary* file yang dimasukkan, isi dari *method,* nama *class* selanjutnya (jika ditemukan *class* lebih dari satu). nama *class* sebelumnya (jika ditemukan *class* lebih dari satu), nama *method* selanjutnya (jika *method* lebih dari satu), nama *method* sebelumnya (jika *method* lebih dari satu), komentar untuk

sebuah *method,* variabel penunjuk apakah ada parameter atau tidak untuk suatu *nethod,* nama parameter untuk suatu *method* (jika ada), nilai *default* untuk parameter (jika ada), suatu array untuk variabel global, file yang di-*include*-kan dalam suatu *class,* variabel yang di-*define* dalam suatu *class* dan h*eader* suatu *class*.

b. Menampilkan komponen-komponen tersebut (di buat sebagai opsi).
c. Menyimpan ke dalam server file class yang dibaca (di buat sebagai opsi).
d. Meng-*overwrite* nama file di server yang sama dengan file yang dibaca (di buat sebagai opsi).
e. Membuat dokumentasi. Di simpan ke dalam format XML (di buat sebagai opsi).

6. Untuk file yang dibaca (dibaca oleh program *class reader*), dapat berasal dari mana saja (lokasi file-nya). Tidak harus file class yang berada di dalam server root. Artinya apabila server root berada di /var/www maka file class yang ingin dibaca boleh berada diluar server root tersebut. Misalnya dari direktori /home/awal.
7. Lokasi tempat penyimpanan file program baik pembaca class maupun "pembangkit dokumentasi" class berada pada direktori yang sama di dalam server.

   Misalnya server root-nya adalah /var/www kemudian file-file tersebut (file program pembaca class dan "pembangkit dokumentasi") di dalam direktori skripsi. Jadi lokasi penyimpananya di /var/www/skripsi.
8. Direktori tempat penyimpanan file-file program harus memiliki hak akses penulisan. (Untuk GNU/Linux posisi kepemilikan pada bagian other adalah bisa ditulisi).

   Misalnya server root-nya adalah /var/www kemudian file-file tersebut (file program pembaca class dan "pembangkit

dokumentasi") di dalam direktori skripsi. Jadi lokasi penyimpananya di /var/www/skripsi. Untuk memberikan akses tersebut gunakan perintah:

```
chmod o+x   /var/www/skripsi
```

9. Pada program "pembangkit dokumentasi" *class*. Untuk bahasa dokumentasi yang dibangkitkan yaitu diambil dari komentar-komentar di dalam file yang dibaca, tergantung dari bahasa yang ditulis di dalam komentar tersebut. Komentar yang dimaksud dalam poin ini adalah komentar yang ditulis dalam bahasa PHP.
10. Fitur program dari "pembangkit dokumentasi" adalah
    a. Dapat di navigasi. Adapun navigasinya adalah sebagai berikut:
        1) Link dan nama *class* yang dipilih.
        2) Link untuk dekripsi *class* yang dipilih.
        3) Link untuk *Method-method* yang terdapat di dalam *class* yang dipilih.
        4) Link untuk Global variabel *class* yang dipilih.
        5) Link untuk file yang dimasukkan di dalam *class* yang dipilih.
        6) Link untuk variabel yang didefinisikan di luar *class* dalam file *class* yang dipilih
        7) Menampilkan semua *method* yang ada pada *class* yang dipilih dengan memanggil *class* xmlgetmethod dan setiap nama-nama tersebut dapat dilink.
    b. Dapat diberi komentar pada *method* untuk suatu *class*.
11. Untuk pola-pola pemograman PHP yang digunakan dapat dilihat di bagian lampiran
12. Program Classreader dan Documentation Generator dapat berjalan di GNU/Linux dengan PHP versi 5. Untuk versi lain PHP3 dan PHP4 belum diadakan percobaan. Begitu juga dengan sistem operasi yang digunakan hanya digunakan di GNU/Linux.

D. Tujuan Penelitian

1. Sebagai syarat kelulusan dari starta satu (S1).
2. Membuat sebuah program pembaca *class* (*class reader)* untuk PHP baik PHP4 maupun PHP5 yang memilah sebuah file yang berisikan *class* menjadi komponen-komponen sehingga mempermudah di dalam mempelajari, memanipulasi dan navigasi.
3. Membuat program dokumentasi *class* untuk PHP baik PHP4 maupun PHP5 yang dibuat secara otomatis dari program pembaca *class* (*class reader*) dan dari dokumentasi yang dibuat dapat dimanipulasi sesuai dengan kebutuhan.

E. Metode Penelitian

Metode yang digunakan dalam penulisan skripsi ini adalah:

1. Metode Observasi

   Metode yang digunakan di dalam pembuatan program pembaca *class* adalah dengan melakukan penelitian terhadap pola-pola penulisan dalam membuat *class* (*class* yang dibuat oleh kebanyakan programmer) seperti gaya(*style*) penulisan, sintaksis (untuk PHP4 dan PHP5 memiliki beberapa perbedaan), komentar-komentar, header (yang bisa nama class, *copyright*, tahun pembuatan, pembuatnya/penulis class (*author)*, lisensinya atau syarat yang diberikan oleh penulis). Melakukan penyaringan terhadap pola-pola yang didapatkan dan memanipulasinya. Hasil manipulasi adalah berupa komponen-komponen yang nantinya untuk membentuk sebuah dokumentasi.

2. Metode Sampling

   Mengambil beberapa sampling dari *class-class* yang pernah dibuat oleh programmer (baik lokal maupun interlokal) dalam membuat

class reader yang nantinya dilanjutkan dengan pembuatan dokumentasinya.

3. Kepustakaan

   Mempelajari beberapa buku yang berkaitan dengan pemograman yang dilakukan.

F. Sistematika Penulisan

1. BAB I PENDAHULUAN

   Pada bagian ini akan dibahas latar belakang masalah, rumusan masalah, batasan masalah, maksud dan tujuan, metode penelitian dan sistematika penulisan

2. BAB II DASAR TEORI

   Pada bagian ini akan dibahas dasar-dasar teori yang berkaitan dengan skripsi seperti teori Pemograman Beorientasi Obyek, PHP, REGEX dan XML.

3. BABA III ANALISA DAN DESAIN

   Pada bagian ini akan dibahas analisis program yang dibuat dan desain program yang dibuat.

4. BAB IV PEMBAHASAN

   Pada bagian ini akan dibahas bagaimana kedua program tersebut dibuat.

5. BAB V IMPLEMENTASI

   Pada bagian ini akan dijelaskan implementasi dari program yang dibuat.

6. BAB VI PENUTUP

   Pada bagian ini berisi saran dan kesimpulan.