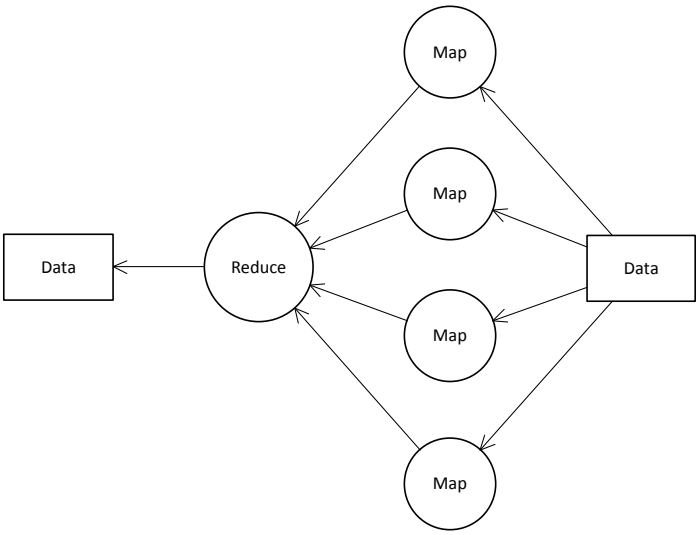
BAB II

LANDASAN TEORI

2.1 MapReduce

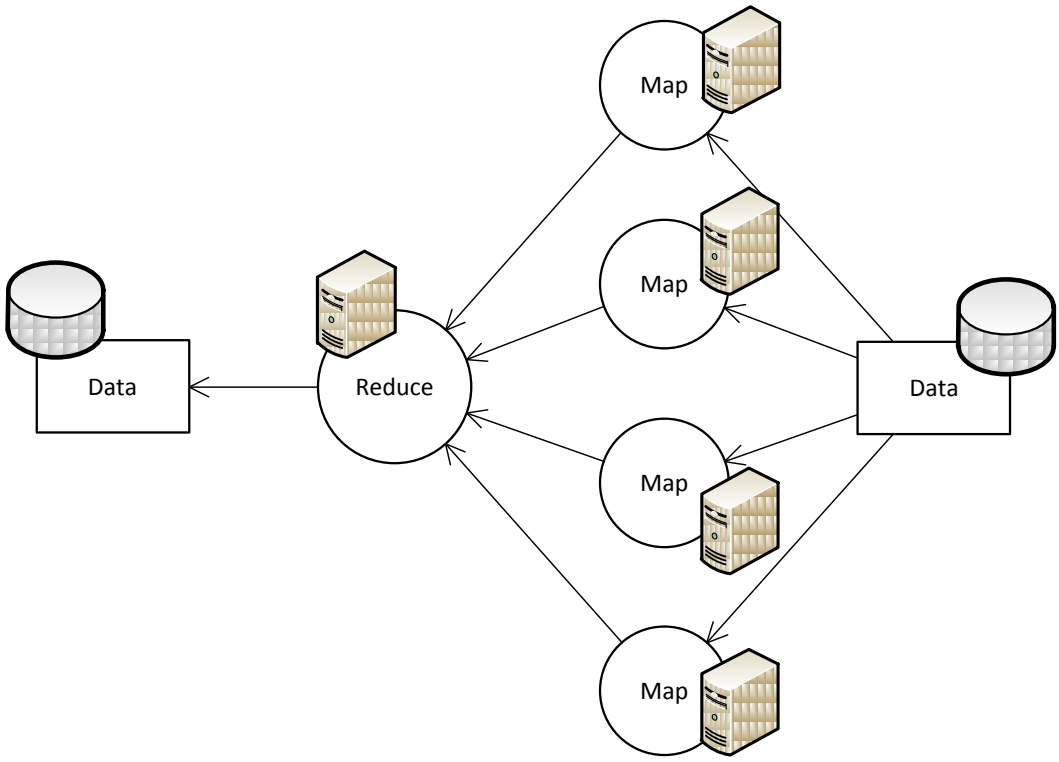
MapReduce adalah model pemrograman terdistribusi yang digunakan untuk melakukan pengolahan data besar. *MapReduce* pertama kali dikenalkan oleh Jeffrey Dean dan Sanjay Ghemawat dari Google, Inc. Saat ini model pemrograman MapReduce digunakan pada sistem pengolahan data di Google dan juga di Yahoo.

Model pemrograman *MapReduce* membagi proses menjadi dua tahapan, yaitu tahapan *Map* dan tahapan *Reduce*. *Map* merupakan proses yang berjalan secara paralel, sedangkan *Reduce* merupakan proses penggabungan hasil dari proses *Map* seperti yang terlihat pada gambar 2.1.



Gambar 2.1 Tahapan Map dan Reduce

Secara otomatis proses *MapReduce* berjalan secara paralel, sehingga model pemrograman *MapReduce* sangat mudah diimplementasikan pada sistem terdistribusi. Pada sistem terdistribusi, sebuah proses *Map* akan dijalankan pada sebuah sistem, biasanya pada sebuah perangkat komputer atau *cluster* seperti yang telihat pada gambar 2.2.



Gambar II.2 Proses MapReduce di Banyak Komputer

Saat ini model pemrograman *MapReduce* hampir diterapkan di setiap perusahaan besar yang menangani pengolahan data besar, seperti Google, Yahoo, Detik.com dan lain-lain.

2.2 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Dikembangkan oleh Sun Microsystems dan

diterbitkan tahun 1995. Java tidak boleh disalahpahami sebagai JavaScript. JavaScript adalah bahasa *scripting* yang digunakan oleh *web browser*.

2.2.1 Sejarah perkembangan Java

Bahasa pemrograman Java pertama lahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran Sand Hill Road di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program Java Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era Star Seven selesai, sebuah anak perusahaan Tv kabel tertarik ditambah beberapa orang dari proyek The Green Project. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Perusahaan baru ini bertambah maju: jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (browser) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot Java.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar San Jose Mercury News pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling.

2.2.2 Versi Awal

Versi awal Java ditahun 1996 sudah merupakan versi release sehingga dinamakan Java Versi 1.0. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

1. java.lang: Peruntukan kelas elemen-elemen dasar.
2. java.io: Peruntukan kelas input dan output, termasuk penggunaan berkas.

3. java.util: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas kelas penanggalan.
4. java.net: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
5. java.awt: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI)
6. java.applet: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

2.2.3 Kelebihan

1. Multiplatform. Kelebihan utama dari Java ialah dapat dijalankan di beberapa platform / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / bytecode) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas operating system Linux tetapi dijalankan dengan baik di atas Microsoft Windows. Platform yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mac OS dan Sun Solaris. Penyebanya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs Java) untuk meninterpretasikan bytecode tersebut.
2. OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di Java adalah Objek. Java merupakan salah satu bahasa pemrograman berbasis objek secara murni.

Semua tipe data diturunkan dari kelas dasar yang disebut Object. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir. Kelebihan ini menjadikan Java sebagai salah satu bahasa pemograman termudah, bahkan untuk fungsi fungsi yang advance seperti komunikasi antara komputer sekalipun.

3. Perpustakaan Kelas Yang Lengkap, Java terkenal dengan kelengkapan library/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.
4. Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.
5. Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan

memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.2.4 Kekurangan

1. Tulis sekali, perbaiki di mana saja - Masih ada beberapa hal yang tidak kompatibel antara platform satu dengan platform lain. Untuk J2SE, misalnya SWT-AWT bridge yang sampai sekarang tidak berfungsi pada Mac OS X.
2. Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan bytecode yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/direverse-engineer.
3. Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena trend memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.2.5 Tahap Kompilasi Java

1. Tulis / Ubah. Pemrogram menulis program dan menyimpannya di media dalam bentuk berkas '.java'.
2. Kompilasi. Pengkompilasi membentuk bytecodes dari program menjadi bentuk berkas '.class'.
3. Muat. Pemuat kelas memuat bytecodes ke memori.
4. Verifikasi. Peng-verifikasi memastikan bytecodes tidak mengganggu sistem keamanan Java.
5. Jalankan. Penerjemah menerjemahkan bytecodes ke bahasa mesin.tidak bisa di pakai‘’

2.3 UML

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya.[1] UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut.[1] UML mulai diperkenalkan oleh Object Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an.[1] Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP.[1] UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM.[1]

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi.

UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

2.3.1 Sejarah UML

Sampai era tahun 1990 puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (method war) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan kelompok/perusahaan lain yang menggunakan metodologi yang berlainan.

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8)[3]. Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG – http://www.omg.org).

2.3.2 Diagram UML

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. Use Case Diagram untuk memodelkan proses bisnis.
2. Conceptual Diagram untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. Sequence Diagram untuk memodelkan pengiriman pesan (message) antar objects.
4. Collaboration Diagram untuk memodelkan interaksi antar objects.
5. State Diagram untuk memodelkan perilaku objects di dalam sistem.
6. Activity Diagram untuk memodelkan perilaku Use Cases dan objects di dalam system.
7. Class Diagram untuk memodelkan struktur kelas.
8. Object Diagram untuk memodelkan struktur object.
9. Component Diagram untuk memodelkan komponen object.
10. Deployment Diagram untuk memodelkan distribusi aplikasi.

2.3.3 Use Case Diagram

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. Use case diagram terdiri atas diagram untuk use case dan actor. Actor merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi.

Use case merepresentasikan operasi-operasi yang dilakukan oleh actor. Use case digambarkan berbentuk elips dengan nama operasi dituliskan di

dalamnya. Actor yang melakukan operasi dihubungkan dengan garis lurus ke use case.

2.3.4 Sequence Diagram

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari use case: interaksi yang terjadi antar class, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

2.3.5 Collaboration Diagram

Collaboration diagram dipakai untuk memodelkan interaksi antar object di dalam sistem. Berbeda dengan sequence diagram yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan, collaboration diagram lebih fokus pada pemahaman atas keseluruhan operasi yang dilakukan oleh object.

2.3.6 Class Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. Class diagram menunjukkan hubungan antar class dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan.

2.4 Teori Bahasa dan Otomata

Ilmu komputer memiliki dua komponen utama; *pertama*, model dan gagasan mendasar mengenai komputasi, *kedua*, teknik rekayasa untuk perancangan sistem komputasi, meliputi perangkat kerasn dan perangkat lunak, khususnya penerapan rancangan rancangan dan teori. Teori komputer diawali dari sejumlah disiplin ilmu, antara lain ahli biologi mempelajari *neural network*,

insinyur elektro mengembangkan *switching* sebagai tool untuk mendesain *hardware*, matemartikawan bekerja mendasarkan logika, dan ahli bahasa menyelidiki tata bahasa untuk *narutal language*.

Finite State Automata dan ekspresi regular awalnya dikembangkan berdasarkan pemikiran *neural network* dan *switching circuit*. *Finite State Automata* merupakan *tool* yang sangat berguna dalam pengembangan *lexical analyzer*, yaitu bagian dari kompilator yang mengelompokkan karakter-karakter ke dalam *token*, yang berupa unit terkecil seperti nama, *variabel* dan *keyword*. Dalam sistem penrn mulisan kompilator secara otomatis akan mentraksformasikan ekspresi regular ke dalam *Finite State Automata* untuk dipakai sebagai pengalalisis leksikal. *Finite State Automata* dan ekspresi regular dipakai pula dalam teks editor, pattern matching, sejumlah pemrosesan teks, dan program, file searching, dan sebagai konsep matematis untuk aplikasi di disiplin lain seperti logika.

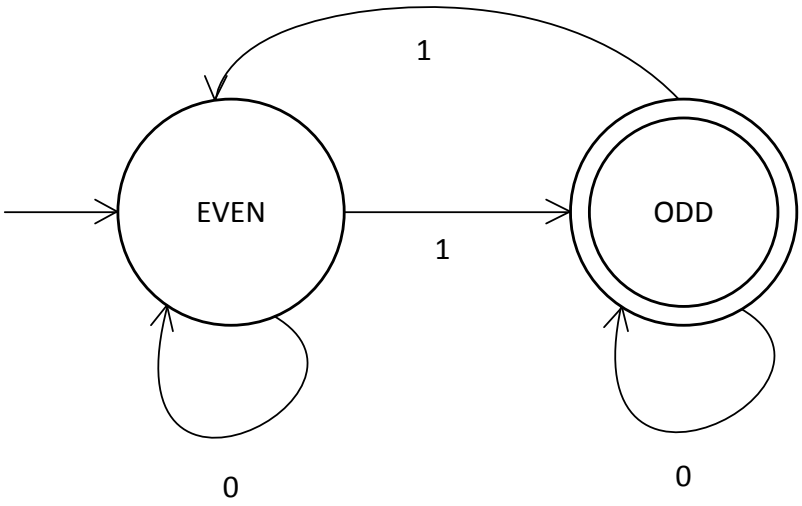
Suatu bahasa pemrograman harus didefinisikan secara tepat. Spesifikasi dari sebuah bahasa pemrograman meliputi hal-hal berikut.

1. Himpunan simbol-simbol yang bisa dipakai untuk membentuk program yang benar.
2. Himpunan program yang benar secara sintaksis.
3. Makna dari program tersebut.

2.4.1 Finite State Automata

Finite State Automata / state otomata berhingga bukanlah mesin fisik, tetapi suatu model matematika dari suatu sistem yang menerima input dan output

diskrit. FSA merupakan mesin otomata dari bahasa regular. Suatu FSA memiliki state yang banyaknya berhingga, dan dapat berpindah-pindah dari satu state ke state yang lain. Perubahan *state* ini dinyatakan oleh fungsi transisi. Jenis otomata ini tidak memiliki tempat penyimpanan sehingga kemampuan mengingatnya terbatas. Mekanisme kontrol pada suato *evelator/lift* adalah contoh yang bagus untuk suatu otomata. Mekanisme tersebut tidak mengingat semua permintaan sebelumnya tetapi hanya posisi lift pada saat itu pada suatu lantai, pergerakan keatas dan kebawah, dan sekumpulan permintaan yang belum terpenuhi.



Gambar II.3 Mesin otomata untuk pencet pariti ganjil

Pada FSA, arti dari bentuk-bentuk seperti pada gambar 2.3 adalah sebagia berikut.

1. Lingkaran menyatakan *state*/kedudukan.
2. Label pada lingkaran adalah nama *state* tersebut.
3. Busur menyatakan transisi yaitu perpindahan kedudukan/*state*.
4. Label pada busur adalah simbol input.
5. Lingkaran didahului sebuah busur tanpa label menyatakan *state* awal.

6. Lingkaran ganda menyatakan state akhir/*final*.

2.4.2 Parsing

Proses parsing merupakan tahapan analisis sintaksis yang berguna untuk memeriksa urutan kemunculan token. Di dalam mengimplementasikan sebuah metode parsing ke dalam program perlu diperhatikan tiga hal sebagai berikut.

1. Rentang waktu eksekusi.
2. Penanganan kesalahan.
3. Penanganan kode.

Metode parsing bisa digolongkan sebagai berikut.

1. *Top Down*

   Jika dilihat dari terminologi pohon penurunan, metode ini melakukan proses penelurusan dari *root*/puncak menuju ke *leaf*/daun (simbol awal sampai simbol terminal). Metode *top down* sendiri meliputi:

   a. *Backtrack/backup : Brute Force.*
   b. *No backtrack : Recursive Descent Parser.*

2. *Bottom Up*

   Metode ini melakukan penelurusan dari *leaf*/daun menuju root/puncak.

2.4.2.1 Parsing dengan Brute Force

Metode ini memilih aturan produksi mulai dari paling kiri, dan melakukan expand semua non terminal pada atuan produksi sampai yan tertinggal adalah simbol terminal. Kemungkinan pertama string masukan sukses di parsing, bisa juga bila terjadi expansi yang salah untuk suatu simbol variabel maka akan

dilakukan backtrack. Algoritma ini membangun pohon parsing *top down*, yaitu mencoba segala kemungkinan untuk setiap simbol non terminal.

Kelamahan dari metode *bruce force* adalah :

1. Mencoba untuk semua aturan produksi yang ada sehingga lambat (rentan waktu eksekusi tidak jelas).
2. Menyulitkan untuk melakukan pemulihan kesalahan.
3. Memakan banak memori karena perlu mencatat *backup* lokasi *backtrack*.

2.4.2.2 Parsing dengan Recursive Descent Parser

Recursive Descent Parser adalah salah satu cara untuk mengaplikasikan bahasa bebas konteks untuk melakukan analisis sintaksis suatu *source code*. Di sini simbol terminal maupun simbol variabelnya sudah bukan sebuah karakter, tetapi berupa besaran leksi sebagai simbol terminalnya dan besaran sintaks sebagai simbol variabelnya. Ciri dari *recursive descent parser* yant menonjol adalah secara rekursive menurunkan sebuah *variabel* dari awal sampai bertemu terminal dan tidak pernah mengambil token secara mundur. Berbeda dengan mesin turing yang dapat maju dan mundur dalam melakukan parsing. Ciri lain dari *rekursif descent parser* adalah sangat tergantung pada algoritma *scan* dalam mengambil token.