



Rekayasa Peranti Lunak Menggunakan Java Modern

Dr. Bambang Purnomosidi D. P. <bambangpdp@gmail.com>

Version 0.0.1-rev-2025-08-24 10:07:50 +0700

Daftar Isi

List of figures	iv
List of tables	v
List of code snippets	vi
Kata Pengantar	vii
Bagian I: Pendahuluan.....	2
1. Teknologi Informasi dan Java.....	3
1.1. Memahami Teknologi Informasi	3
1.2. Peranti Pengembangan, Bahasa Pemrograman, Interpreter dan Kompilator	6
1.3. Kategori Bahasa Pemrograman	7
1.4. Paradigma Pemrograman	10
1.5. Posisi Java dalam Dunia Pemrograman.....	14
2. Another Chapter.....	16
2.1. Sub Seciton 1.....	16
2.2. See the Source Code Here	16
2.3. Rust Compilation Process.....	17
Bagian II: Bahasa Pemrograman Java	18
3. Another Chapter.....	19
3.1. Sub Section 1.....	19
4. Another Chapter.....	20
4.1. Sub Section 1.....	20
Bagian III: Paradigma Pemrograman di Java.....	21
5. Another Chapter.....	22
5.1. Sub Section 1.....	22
6. Another Chapter.....	23
6.1. Sub Section 1.....	23
Bagian IV: API di Java.....	24
7. Another Chapter.....	25
7.1. Sub Section 1.....	25
8. Another Chapter.....	26

8.1. Sub Section 1.....	26
Bagian V: Konsep Tingkat Lanjut di Rekayasa Peranti Lunak dan Java Modern .	27
9. Another Chapter.....	28
9.1. Sub Section 1.....	28
10. Another Chapter.....	29
10.1. Sub Section 1.....	29
Daftar Pustaka	30
Lampiran A: First Appendix.....	31
A.1. Part 1 of first appendix.....	31
A.2. Part 2 of first appendix.....	31
Lampiran B: Second Appendix	32
B.1. Part 1 of second appendix	32
B.2. Part 2 of second appendix	32
Ringkasan	33
Indeks.....	34

List of figures

Gambar 1. contoh bahasa mesin

Gambar 2. TIOBE index bulan Agustus 2025

Gambar 3. Rust Compilation Process

List of tables

[Tabel 1.](#) First table

[Tabel 2.](#) Second table

[Tabel 3.](#) First table

[Tabel 4.](#) Second table

[Tabel 5.](#) First table

[Tabel 6.](#) Second table

[Tabel 7.](#) First table

[Tabel 8.](#) Second table

List of code snippets

Kode Sumber 1. `src/01-02/hello-plain/hello.rs`

Kode Sumber 2. `rustup`

Kata Pengantar

Buku ini ditulis bagi pembaca yang ingin mendalami bahasa pemrograman Java dengan *kompilator / compiler* yang tersedia bebas dalam konteks perekayasaan peranti lunak. Buku ini unik dalam hal kelengkapan dan kedalaman pembahasan serta keterkaitannya dengan rekayasa peranti lunak. Lebih khusus lagi, buku ini membahas tentang platform pengembangan serta *deployment* terkini dengan menggunakan JDK modern.

Selama ini, banyak orang yang mengasosiasikan Java dengan *bloated / memerlukan sumber daya / resources* yang berlebihan. Selain itu, kompleksitas Java dalam pembuatan kode sumber maupun ekosistem juga merupakan sisi lain dari Java. Buku ini berusaha untuk menjadi solusi terhadap hal-hal tersebut. Setelah membaca buku ini, pembaca akan mendapatkan gambaran umum tentang dunia rekayasa peranti lunak menggunakan Java modern pada lingkungan pengembangan dan *deployment* yang modern. Buku ini juga menyiapkan para pembaca untuk berkarir dalam dunia industri peranti lunak sebagai pengembang peranti lunak. Dengan membaca buku ini, pembaca tidak hanya akan memahami Java, tetapi juga akan mendapatkan pengetahuan berikut:

1. Memahami berbagai pola pikir dalam pengembangan peranti lunak / kode sumber (obyek, imperatif, fungsional, dan lain-lain)
2. Memahami berbagai terminologi serta berbagai isu sentral rekayasa peranti lunak yang melingkupi Java
3. Memahami cara membuat dan mengoptimalkan program menggunakan Java
4. Memahami berbagai cabang yang bisa didalami di industri peranti lunak.

Buku ini tentu saja tidak akan terwujud tanpa peran serta dari berbagai pihak. Untuk itu, penulis mengucapkan terima kasih mendalam kepada:

1. Keluarga penulis

2. Rekan-rekan di Yayasan Jamali

3. Rekan-rekan dosen maupun tendik di Universitas Teknologi Digital Indonesia

Akhir kata, buku ini tentu saja masih jauh dari kekurangan. Saran dan masukan bisa disampaikan ke penulis secara langsung di bambangpdp@gmail.com. Selamat membaca, berlatih, dan terima kasih!

Yogyakarta, 2025

Bambang Purnomosidi D. P.

Bagian I: Pendahuluan

Bagian ini membahas tentang hal-hal yang bersifat mendasar dan merupakan latar belakang serta kondisi ekosistem Java sebagai suatu bahasa pemrograman. Bagian ini diperlukan untuk memberikan wawasan terhadap bahasa pemrograman Java dalam konteks pengembangan peranti lunak. Dengan mempelajari bagian ini, pembaca akan mempunyai wawasan yang cukup tentang dunia pengembangan peranti lunak serta keterkaitannya dengan Java modern.

Bab 1. Teknologi Informasi dan Java

Bab ini membahas tentang gambaran umum dunia teknologi informasi dengan maksud untuk memberikan pemahaman tentang posisi Java dalam teknologi informasi. Dengan memahami bab ini, anda bisa mengetahui berbagai komponen teknologi informasi secara umum dan kemudian memahami posisi dan ruang lingkup pekerjaan anda jika anda memutuskan untuk mendalami Java

1.1. Memahami Teknologi Informasi

1.1.1. Pengertian Teknologi Informasi

Teknologi informasi sering disingkat dengan TI dan merupakan penyerapan istilah dari bahasa Inggris Information Technology (IT). Dalam praktik dan berbagai artikel serta buku, banyak istilah lainnya yang merujuk ke penyebutan TI, misalnya:

- Telematika (merujuk pada konvergensi antara telekomunikasi, media, dan informatika).
- TIK (Teknologi Informasi dan Komunikasi), berasal dari istilah ICT (*Information and Communication Technology*), sinonim dengan TI tetapi lebih menekankan pada telekomunikasi.

Berbagai istilah tersebut sebenarnya menunjukkan adanya kerancuan pemahaman. Istilah IT/TI/TIK/ICT masih bisa dipahami dalam kerangka industri, tetapi penggunaan istilah Informatika (dari bahasa Inggris: Informatics) seharusnya hanya merujuk pada suatu disiplin keilmuan seperti halnya Mathematics – Matematika, Statistics – Statistika, dan lain-lain. Istilah Informatika lebih sesuai digunakan dalam kerangka proses pendidikan.

Buku ini akan menggunakan istilah teknologi informasi dan akan menggunakan TI jika akan disingkat. Penggunaan istilah TI akan merujuk ke hal yang sama seperti yang dimaksud oleh berbagai jargon tersebut di atas. Meskipun demikian,

perlu diketahui bahwa tidak ada satu badan atau organisasi di dunia yang mempunyai “hak” untuk membuat standar istilah sehingga diferensiasi istilah ini akan tetap ditemui.

Definisi dari **TechTarget** adalah sebagai berikut.^[1]

Information technology (IT) is the use of computers, storage, networking and other physical devices, infrastructure and processes to create, process, store, secure and exchange all forms of electronic data. Typically, IT is used in the context of business operations, as opposed to the technology used for personal or entertainment purposes. The commercial use of IT encompasses both computer technology and telecommunications.

— TechTarget

Terjemahan bebas dalam bahasa Indonesia adalah sebagai berikut:

Teknologi informasi (TI) adalah penggunaan komputer, penyimpanan, jaringan, dan perangkat fisik lainnya, infrastruktur, serta proses untuk membuat, memproses, menyimpan, mengamankan, dan bertukar segala bentuk data elektronik. Umumnya, TI digunakan dalam konteks operasional bisnis, bukan untuk keperluan pribadi atau hiburan. Penggunaan TI secara komersial mencakup teknologi komputer dan telekomunikasi.

— TechTarget

1.1.2. Komponen Teknologi Informasi

Dari pengertian di atas, kita akan bergerak ke bagian yang lebih spesifik. TI pada dasarnya terdiri atas berbagai komponen:

1. Peranti keras komputer, meliputi berbagai peranti yang saat ini bisa dikategorikan sebagai komputer: PC, workstation, NetBook, Laptop, dan lain-lain.

2. Peranti telekomunikasi dan jaringan, meliputi berbagai peranti yang membentuk infrastruktur telekomunikasi dan jaringan seperti router, switch, wireless access point, kabel, server, satelit, dan berbagai teknologi konektivitas jaringan lainnya.
3. Peranti lunak, yaitu peranti yang diinstalasi atau di-embed dalam peranti keras dan berbagai peranti telekomunikasi dan jaringan lainnya sehingga memungkinkan peranti-peranti tersebut dapat diprogram dan bekerja.

Peranti lunak (diterjemahkan dari kata software dan seringkali juga disebut sebagai program) biasanya bisa dikategorikan menjadi 3 bagian:

1. Sistem operasi, yaitu peranti lunak yang menjadi perantara antara peranti keras dan peranti lunak lainnya. Fungsi utama dari sistem operasi adalah pengelolaan berbagai sumber daya komputasi serta menyediakan abstraksi dalam bentuk API (Application Programming Interface) yang bisa digunakan oleh peranti lunak lainnya untuk mengendalikan kerja dari peranti keras. Contoh dari sistem operasi adalah Linux, FreeBSD, NetBSD, OpenBSD, MacOSX, Minix, Windows, dan lain-lain.
2. Aplikasi, yaitu peranti lunak yang digunakan untuk membantu manusia menyelesaikan pekerjaan dalam bidang tertentu, misalnya OpenOffice (<http://www.openoffice.org>) digunakan untuk keperluan aplikasi perkantoran dan bisa digunakan untuk menyelesaikan berbagai masalah komputasi perkantoran (pengolah kata, spreadsheet, dan presentasi).
3. Utilitas, yaitu peranti lunak yang menyediakan fasilitas-fasilitas untuk memperbaiki kinerja dari komputer, misalnya utilitas antivirus, utilitas pengatur partisi harddisk, dan lain-lain.
4. Peranti pengembangan (software development tools), yaitu peranti yang digunakan untuk membuat peranti lunak (termasuk diantaranya untuk membuat sistem operasi, aplikasi, utilitas, maupun peranti pengembangan). Contoh dari kategori ini adalah JDK (Java Development Kit), GNU C++, Python, Ruby, Nasm (Netwide Assembler), Microsoft Visual Studio, Anjuta, KDevelop,

dan lain-lain.

1.2. Peranti Pengembangan, Bahasa Pemrograman, Interpreter dan Kompilator

Jika membicarakan bahasa pemrograman (komputer) dan keterkaitannya dengan industri software, maka berarti kita bicara tentang spesifikasi bahasa pemrograman serta implementasinya. Penyebutan “bahasa pemrograman” sebenarnya hanya mengacu pada spesifikasi, yaitu aturan sintaksis serta tata bahasa dan kelengkapan bahasa pemrograman tersebut. Tentu saja hal itu tidak akan lengkap tanpa “implementasi”. Implementasi dari spesifikasi bahasa pemrograman tersebut terletak pada peranti lunak interpreter / kompilator. Jadi, jika kita bicara tentang “Java”, berarti sebenarnya kita membicarakan tentang spesifikasi bahasa pemrograman Java dan implementasi kompilator Java (JDK, OpenJDK, dan lain-lain).^[2]

Fungsi dari interpreter / kompilator adalah menerjemahkan instruksi yang ditulis sesuai dengan spesifikasi bahasa pemrograman tertentu ke dalam bentuk target yang bisa dijalankan oleh mesin komputer. Bentuk hasil akhirnya bisa berupa berkas executable atau berkas yang bisa dijalankan secara langsung oleh sistem operasi, atau format tertentu (biasanya disebut sebagai object code) yang masih memerlukan mesin virtual untuk menjalankannya.

Peranti pengembangan yang lengkap terdiri atas banyak bagian, meskipun secara minimal bisa dikatakan bahwa peranti pengembangan adalah kompilator / interpreter serta pustaka (library) standar. Kompilator / interpreter biasanya berkaitan dengan sintaksis standar sesuai spesifikasi bahasa pemrograman, misalnya untuk if ... else, while ... do, pengaturan variabel, dan lain-lain. Pustaka standar merupakan konstruksi program siap pakai yang langsung bisa digunakan, misalnya:

1. Java mempunyai API (Application Programming Interface) yang berisi pustaka standar untuk membangun aplikasi GUI dengan Swing, Kriptografi, operasi

String, operasi Matematika, koneksi basis data dengan JavaDB, dan lain-lain.

2. Python mempunyai kelas siap pakai yang bisa digunakan untuk XMLRPC, JSON, dan lain-lain.

Pustaka standar ini yang membedakan “fasilitas” yang dimiliki oleh setiap peranti pengembangan. Semakin lengkap pustaka standar ini, semakin mudah kita menggunakan peranti pengembangan tersebut. Jika tidak terdapat pada pustaka standar, biasanya pemrogram harus mencari dari pihak ketiga, misalnya untuk koneksi Java menggunakan JDBC ke basis data PostgreSQL pemrogram harus menggunakan API yang bisa diperoleh di <http://jdbc.postgresql.org> karena tidak disediakan di pustaka standarnya.

1.3. Kategori Bahasa Pemrograman

Pada dasarnya pemrogram menggunakan bahasa pemrograman untuk menuliskan sintaksis dan kemudian menggunakan kompiler / interpreter untuk menterjemahkan ke bentuk bahasa yang dipahami mesin dan bisa dijalankan oleh mesin. Dengan demikian, kita bisa memahami bahwa mesin komputer itu sendiri mempunyai “bahasa”. Ada banyak cara manusia untuk memprogram mesin supaya tidak langsung berurusan dengan bahasa mesin yang rumit. Pada bagian ini kita akan melihat kategorisasi bahasa pemrograman berdasarkan kedekatannya dengan bahasa mesin.

1.3.1. Bahasa Mesin

Instruksi bahasa mesin adalah satu-satunya instruksi yang dipahami oleh komputer. Instruksi ini merupakan suatu pola bit yang hanya terdiri atas angka 0 dan 1 (binary digit). Setiap perintah mempunyai suatu pola tersendiri. Bagian dari bahasa mesin yang menyebutkan operasi yang harus dilaksanakan disebut sebagai opcode (operation code). Bahasa ini berbeda-beda untuk setiap prosesor, sehingga untuk mendalami bahasa ini, berarti harus mendalami tipe prosesor tertentu dan tidak bisa diaplikasikan ke tipe prosesor lain. Gambar tentang [contoh bahasa mesin](#) berikut ini adalah contoh bahasa mesin untuk melompat ke

alamat memory 1024.^[3]

[op target address]		
2	1024	decimal
000010 00000 00000 00000 10000 000000		binary

Gambar 1. contoh bahasa mesin

1.3.2. Bahasa Aras Rendah (Assembly)

Bahasa assembly merupakan penyederhanaan dari bahasa mesin dengan cara mengubah pola bit angka 0 dan 1 tersebut menjadi suatu kode yang bisa dibaca. Kode tersebut dikenal dengan istilah mnemonic. Suatu peranti lunak yang digunakan untuk mengubah instruksi bahasa assembly (mnemonic) ke dalam opcode serta mengatur alokasi memory dari nama-nama simbolis dan entitas lainnya adalah assembler. Untuk mengubah ke bentuk yang siap dijalankan oleh mesin, biasanya dibutuhkan linker. Bahasa ini juga sangat tergantung pada mesin dan platform sistem operasi. Di Linux, bisa digunakan Nasm, GNU As sebagai assembler dan GNU ld sebagai linker. Di Windows, bisa digunakan TASM dan MASM untuk keperluan ini. Berikut adalah contoh dari bahasa assembly:

```
...  
...  
mov al, 61h  
...  
...
```

Instruksi di atas digunakan untuk mengisi register **al** dengan nilai hexadecimal 61. Jika diterjemahkan ke bahasa mesin, mnemonic MOV di atas akan menjadi “B0 61” dalam hexadecimal atau 10110000 01100001 dalam biner atau bahasa mesin.

Untuk bisa memprogram dalam bahasa mesin dan assembly, pemrogram perlu membaca manual prosesor yang bersangkutan. Sebagai contoh, jika menggunakan prosesor dari Intel, instruksi-instruksi di atas bisa diperoleh di website dari Intel di URL <http://www.intel.com/products/processor/manuals/>.

1.3.3. Bahasa Tingkat Menengah

Bahasa tingkat menengah (middle level language) adalah bahasa yang memungkinkan untuk membuat aplikasi di level pemakai maupun untuk melakukan pemrograman yang mengakses sistem dan sumber daya komputer melalui perantara sistem operasi. Bahasa ini memang dikatakan bahasa tingkat menengah karena mempunyai kemampuan baik untuk mengakses sistem secara langsung (biasanya melalui inline assembly atau langsung menuliskan mnemonic di kode sumber) dan di sisi lainnya bahasa ini mempunyai tingkat kemudahan yang mendekati bahasa tingkat tinggi (menggunakan istilah bahasa manusia dalam pemberian perintahnya). Bahasa C adalah bahasa yang masuk dalam kategori ini. Berikut ini adalah contoh dari Bahasa C menggunakan kompiler GCC yang menyertakan bahasa assembly.^[4]

```
#include <stdio.h>
#include <stdint.h>
#define inf_int uint64_t
int main(int argc, char *argv[]){
    inf_int zero = 0;
    inf_int one = 1;
    inf_int infinity = ~0;
    printf("value of zero, one, infinity = %lu, %lu, %lu\n", zero, one,
infinity);
    __asm__ (
        "addq $1, %0 \n\t"
        : "+r" (zero)
    );
    __asm__ (
        "addq $1, %0 \n\t"
        : "+r" (one)
    );
    __asm__ (
        "addq $1, %0 \n\t"
        : "+r" (infinity)
    );
    printf("value of zero, one, infinity = %lu, %lu, %lu\n", zero, one,
infinity);
    return 0;
}
```


1.3.4. Bahasa Tingkat Tinggi

Bahasa tingkat tinggi adalah bahasa pemrograman yang mempunyai sintaks mirip dengan bahasa manusia dan mempunyai tingkat abstraksi penyelesaian masalah yang jauh dari bahasa mesin ataupun assembly. Bahasa ini biasanya bisa dikategorikan ke dalam:

1. General purpose high level programming language, yaitu bahasa pemrograman dengan tujuan penggunaan umum atau bisa digunakan untuk membuat berbagai jenis aplikasi. Contoh dari bahasa pemrograman ini antara lain adalah Java, Pascal, Python, Ruby, dan lain-lain.
2. Special purpose high level programming language, yaitu bahasa pemrograman dengan tujuan khusus dan spesifik untuk menyelesaikan masalah dalam suatu bidang tertentu, misalnya Prolog untuk pemrograman logika atau kecerdasan buatan, PHP untuk aplikasi web (meskipun ada yang non web, yaitu PHP-GTK, tetapi penggunaan utamanya untuk web),

1.4. Paradigma Pemrograman

Paradigma pemrograman berkaitan dengan cara pandang dalam menyelesaikan masalah dengan bahasa pemrograman. Cara pandang ini akan mengakibatkan berbagai perbedaan antar paradigma dalam hal abstraksi dan berbagai konsep yang merepresentasikan elemen dari suatu program (seperti obyek, fungsi, variabel, dan lain-lain) serta berbagai langkah atau proses komputasi (penugasan, evaluasi ekspresi, alur kendali, dan lain-lain). Bagian ini akan menguraikan sedikit dari beberapa paradigma tersebut.

1.4.1. Pemrograman Prosedural

Pemrograman prosedural sering diasosiasikan dengan pemrograman terstruktur dan mengacu pada cara menyelesaikan masalah dengan membuat langkah-langkah terstruktur (disebut algoritma) dan kemudian mengimplementasikan langkah-langkah tersebut dalam berbagai perintah dan prosedur yang akan

dipanggil sesuai dengan urutan eksekusi program. Setiap prosedur tersebut berisi rangkaian perintah dalam bahasa pemrograman yang bersangkutan. Biasanya rangkaian langkah-langkah tersebut dijabarkan dalam suatu pseudo code. Sebagai contoh, untuk menghitung biaya depresiasi tiap periode dilakukan dengan melihat pada rumus “Biaya Depresiasi = (Kos Aset – Nilai Residu)/Umur Ekonomis” dan diwujudkan dalam algoritma berikut:

1. Mulai
2. Masukkan / input kos aset
3. Masukkan / input nilai residu
4. Masukkan / input umur ekonomis
5. Hitung biaya depresiasi = (kos aset – nilai residu) / umur ekonomis
6. Tampilkan hasil perhitungan biaya depresiasi ke layar

Setelah itu, setiap bagian dan langkah di atas diterjemahkan ke dalam kode bahasa pemrograman. Contoh-contoh bahasa pemrograman yang bisa digunakan untuk mengimplementasikan algoritma di atas antara lain adalah Pascal, Python, BASIC, dan lain-lain. Berikut ini adalah hasil penerjemahan ke dalam bahasa pemrograman Python untuk algoritma di atas:

```
kos_aset = float(raw_input("Masukkan kos aset = "))
nilai_residu = float(raw_input("Masukkan nilai residu = "))
umur_ekonomis = float(raw_input("Umur ekonomis = "))
biaya_depresiasi = (kos_aset - nilai_residu) / umur_ekonomis
print "Biaya depresiasi setiap periode = " + str(biaya_depresiasi)
```

1.4.2. Pemrograman Berorientasi Obyek

Pemrograman berorientasi obyek (diterjemahkan dari “Object-Oriented Programming”) adalah paradigma pemrograman yang berusaha untuk menyelesaikan masalah pemrograman dengan cara membuat abstraksi dari berbagai obyek yang ada dalam suatu masalah kemudian mendefinisikan interaksi antar obyek dalam menyelesaikan masalah tersebut. Setiap obyek

merupakan suatu manifestasi dari kelas yang merupakan cetak biru dari obyek yang bersangkutan. Suatu kelas terdiri atas berbagai atribut / karakteristik umum dan berbagai perilaku / behaviour / method dari kelas tersebut. Kedudukan antara kelas dengan obyek adalah kedudukan cetak biru serta instance dari cetak biru tersebut. Sebagai contoh, terdapat kelas manusia dan Bambang adalah manusia, untuk contoh tersebut kita bisa mendefinisikan kelas manusia dan Bambang adalah salah satu instance dari kelas manusia tersebut. Beberapa bahasa pemrograman yang mendukung PBO sebagai paradigma utamanya adalah Java, C++, C#, Ruby, Smalltalk, dan lain-lain. Paradigma ini yang akan menjadi pembahasan utama dari buku ini.

1.4.3. Pemrograman Fungsional

Pemrograman fungsional (diterjemahkan dari “Functional Programming”) adalah paradigma pemrograman yang berusaha menyelesaikan masalah pemrograman dengan melalui fungsi (function) matematis dan menghindari mutable data. Jadi, tidak seperti paradigma pemrograman prosedural yang mengutamakan berbagai tipe data dan kemudian membuat prosedur atau fungsi untuk memanipulasi data tersebut. Paradigma ini memang cenderung lebih “matematis” dan akan lebih mudah dipahami jika pemrogram yang belajar paradigma ini memahami matematika. Paradigma ini berasal dari suatu sistem formal yang disebut dengan Lambda Calculus. Spreadsheet adalah salah satu contoh dari pola pikir fungsional ini, terutama dengan melihat pada data yang terdapat pada spreadsheet serta berbagai elemen rumus-rumus di berbagai sel di spreadsheet. Contoh bahasa pemrograman yang mempunyai paradigma pemrograman fungsional sebagai paradigma utamanya adalah Haskell, Erlang, OCaml, Lisp, Scheme, Scala, Clojure, F#, dan lain-lain

1.4.4. Beberapa Pandangan Lain dalam Paradigma Pemrograman

Perlu diketahui bahwa pembagian menjadi beberapa paradigma ini merupakan pembagian yang bersifat subyektif. Selain paradigma-paradigma di atas, masih ada beberapa lagi yang sering dianggap sebagai paradigma tersendiri.

1. Pemrograman Imperatif. Sering kali dianggap sama dengan pemrograman prosedural, meskipun demikian, ada juga yang menyatakan lain. Perbedaan dengan pemrograman prosedural terutama terletak pada penggunaan prosedur (atau sub program atau subroutines atau functions) yang banyak untuk memperbaiki keterbacaan (readability) dan kemudahan pemeliharaan program (maintainability). Model yang lebih tegas lagi dengan menggunakan aturan penggunaan variabel lokal di level prosedur sebenarnya juga merupakan bagian dari pemrograman imperatif tetapi lebih sering dimasukkan dalam pemrograman terstruktur (structured programming). Beberapa bahasa pemrograman yang mendukung paradigma ini sama dengan bahasa pemrograman yang mendukung paradigma pemrograman prosedural hanya teknik pemrogramannya yang berbeda.
2. Pemrograman Deklaratif. Berbeda dengan pemrograman imperatif yang secara eksplisit cenderung untuk menetapkan algoritma untuk menyelesaikan suatu masalah pemrograman, pemrograman deklaratif cenderung untuk mengekspresikan logika komputasi tanpa mendefinisikan alur kendali program. Pemrograman logika menggunakan Prolog merupakan salah satu contoh paradigma pemrograman deklaratif. Paradigma pemrograman ini cenderung lebih sesuai dipelajari jika ranah aplikasi yang dibangun melibatkan matematika logika. Contoh dari bahasa pemrograman yang mempunyai paradigma utama pemrograman deklaratif antara lain adalah Prolog.
3. Pemrograman Event-Driven. Pemrograman event-driven adalah suatu paradigma pemrograman yang mengatur aliran program melalui berbagai event / kejadian. Paradigma ini kadang kala juga disebut dengan paradigma pemrograman visual. Pemrogram biasanya membuat form dan meletakkan berbagai komponen (menu, pushbutton, radio button, text area, dan lain-lain) ke dalam form tersebut kemudian mengisikan rangkaian perintah yang dijalankan jika terdapat kejadian yang berkaitan dengan komponen-komponen tersebut (misalnya menu dipilih, push button di klik, form di tutup, dan lain-lain). Contoh peranti pengembangan yang masuk dalam kategori ini

antara lain adalah Borland Delphi (dengan bahasa Pascal), QT Designer (dengan bahasa C++), Swing di Apache NetBeans (dengan bahasa Java), dan lain-lain.





















1.4.5. Kondisi Saat Ini

Pada dasarnya, saat ini jarang ada peranti pengembangan yang hanya menggunakan satu paradigma saja. Saat ini kebanyakan peranti pengembangan merupakan peranti pengembangan yang multiparadigm atau menganut lebih dari satu paradigma meskipun biasanya ada satu paradigma yang dijadikan dasar dari peranti pengembangan tersebut. Beberapa contoh dari multiparadigm tersebut adalah sebagai berikut: 1. Python, mempunyai dasar paradigma pemrograman berorientasi obyek, tetapi juga mendukung prosedural maupun fungsional. 2. OCaml, mempunyai dasar paradigma pemrograman fungsional, tetapi juga mendukung pemrograman berorientasi obyek. 3. Java, mempunyai dasar paradigma pemrograman berorientasi obyek, tetapi juga mendukung pemrograman event-driven.

1.5. Posisi Java dalam Dunia Pemrograman

Java sebenarnya bisa dimasukkan dalam kategori bahasa pemrograman tingkat tinggi dan secara umum termasuk dalam kelompok paradigma pemrograman berorientasi obyek. Peranti pengembangan Java merupakan bagian dari peranti pengembangan yang digunakan untuk membangun berbagai aplikasi maupun berbagai kategori peranti lunak lain (misalnya sistem operasi dengan Jnode). Dengan demikian, jika tujuan anda adalah membangun aplikasi, maka peranti pengembangan Java ini akan menjadi wilayah pekerjaan anda.

Perlu diketahui, di Internet terdapat beberapa situs web yang memang dibangun untuk mengetahui posisi berbagai peranti pengembangan di dunia. Salah satu dari situs web tersebut antara lain adalah indeks menurut TIOBE. TIOBE bisa diakses melalui URL <https://www.tiobe.com/tiobe-index/>. Gambar posisi masing-masing bahasa pemrograman bisa dilihat pada [TIOBE index bulan Agustus 2025](#).

Aug 2025	Aug 2024	Change	Programming Language		Ratings	Change
1	1			Python	26.14%	+8.10%
2	2			C++	9.18%	-0.86%
3	3			C	9.03%	-0.15%
4	4			Java	8.59%	-0.58%
5	5			C#	5.52%	-0.87%
6	6			JavaScript	3.15%	-0.76%
7	8	▲		Visual Basic	2.33%	+0.15%
8	9	▲		Go	2.11%	+0.08%
9	25	▲		Perl	2.08%	+1.17%
10	12	▲		Delphi/Object Pascal	1.82%	+0.19%
11	10	▼		Fortran	1.75%	-0.03%
12	7	▼		SQL	1.72%	-0.49%
13	30	▲		Ada	1.52%	+0.91%
14	19	▲		R	1.37%	+0.26%
15	13	▼		PHP	1.27%	-0.19%
16	11	▼		MATLAB	1.19%	-0.53%
17	20	▲		Scratch	1.15%	+0.06%
18	14	▼		Rust	1.13%	-0.15%
19	18	▼		Kotlin	1.10%	-0.04%
20	17	▼		Assembly language	1.03%	-0.19%

Gambar 2. TIOBE index bulan Agustus 2025

[1] <https://www.techtarget.com/searchdatacenter/definition/IT>

[2] Ada juga kompilator / interpreter yang namanya sama dengan spesifikasi bahasa pemrogramannya, misalnya Python, Ruby

[3] https://en.wikipedia.org/wiki/Machine_code

[4] Diambil dari <https://stackoverflow.com/questions/31688987/why-is-this-simple-c-program-with-gcc-clang-inline-assembly-exhibiting-undefin>

Bab 2. Another Chapter

2.1. Sub Seciton 1

[illegible]

In this chapter, I will give you an example of how to format source code using AsciiDoctor.

2.2. See the Source Code Here

Kode Sumber 1. `src/01-02/hello-plain/hello.rs`

```
1 fn main() { ①  
2  
3     println!("Hello World!"); ②  
4  
5 }
```

① Explanation - callout number 1.

② Explanation - callout number 2.

For any other source which doesn't relate to source code in programming language, use this:

Kode Sumber 2. rustup

```
$ curl --proto 'https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

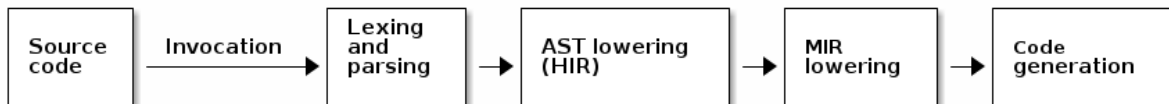
```
info: downloading installer
```

```
...
```

```
...
```

2.3. Rust Compilation Process

Rust compilation process can be seen at [Rust Compilation Process](#).



Gambar 3. Rust Compilation Process

Bagian II: Bahasa Pemrograman Java

Bagian ini membahas tentang pengetahuan pemrograman menggunakan Java. Bab-bab pada bagian ini membahas tentang sintaksis dari bahasa pemrograman Java. Dengan memahami bagian ini, diharapkan pembaca bisa membuat program / kode sumber Java secara efektif.

Bab 3. Another Chapter

3.1. Sub Section 1

Tabel 1. First table

Column 1, Header Row	Column 2, Header Row
Cell in column 1, row 1	Cell in column 2, row 1
Cell in column 1, row 2	Cell in column 2, row 2

Bab 4. Another Chapter

4.1. Sub Section 1

Tabel 2. Second table

Column 1, Header Row	Column 2, Header Row
Cell in column 1, row 1	Cell in column 2, row 1
Cell in column 1, row 2	Cell in column 2, row 2

Bagian III: Paradigma Pemrograman di Java

Paradigma dalam bahasa pemrograman merupakan cara pandang untuk menghasilkan solusi terhadap berbagai masalah pemrograman. Saat ini, bisa dikatakan bahwa Java modern merupakan bahasa pemrograman yang mendukung berbagai paradigma. Bagian ini terutama digunakan untuk membentuk pola pikir sesuai paradigma pemrograman yang dibahas serta implementasinya menggunakan Java modern. Bagian ini tidak membahas secara khusus tentang PBO (Pemrograman Berorientasi Obyek) karena PBO merupakan paradigma default dari Java dan telah dibahas di bagian II.

Bab 5. Another Chapter

5.1. Sub Section 1

Tabel 3. First table

Column 1, Header Row	Column 2, Header Row
Cell in column 1, row 1	Cell in column 2, row 1
Cell in column 1, row 2	Cell in column 2, row 2

Bab 6. Another Chapter

6.1. Sub Section 1

Tabel 4. Second table

Column 1, Header Row	Column 2, Header Row
Cell in column 1, row 1	Cell in column 2, row 1
Cell in column 1, row 2	Cell in column 2, row 2

Bagian IV: API di Java

Bagian ini membahas tentang berbagai API (*Application Programming Interface*) standar di Java. Java mempunyai beberapa API standar yang menyediakan berbagai fungsionalitas tertentu. Dengan memahami bagian ini, diharapkan pembaca mengetahui berbagai API standar di Java dan bisa menggunakannya dalam kode sumber Java modern secara efektif.

Bab 7. Another Chapter

7.1. Sub Section 1

Tabel 5. First table

Column 1, Header Row	Column 2, Header Row
Cell in column 1, row 1	Cell in column 2, row 1
Cell in column 1, row 2	Cell in column 2, row 2

Bab 8. Another Chapter

8.1. Sub Section 1

Tabel 6. Second table

Column 1, Header Row	Column 2, Header Row
Cell in column 1, row 1	Cell in column 2, row 1
Cell in column 1, row 2	Cell in column 2, row 2

Bagian V: Konsep Tingkat Lanjut di Rekayasa Peranti Lunak dan Java Modern

Bagian ini membahas berbagai konsep tingkat lanjut di Java modern. Konsep tingkat lanjut ini diperlukan terutama terkait dengan proses pengembangan dan *deployment* pada berbagai platform dan lingkungan komputasi modern. Dengan memahami bagian ini, diharapkan pembaca bisa menerapkan berbagai materi yang telah dibahas untuk pengembangan peranti lunak menggunakan Java modern secara utuh.

Bab 9. Another Chapter

9.1. Sub Section 1

Tabel 7. First table

Column 1, Header Row	Column 2, Header Row
Cell in column 1, row 1	Cell in column 2, row 1
Cell in column 1, row 2	Cell in column 2, row 2

Bab 10. Another Chapter

10.1. Sub Section 1

Tabel 8. Second table

Column 1, Header Row	Column 2, Header Row
Cell in column 1, row 1	Cell in column 2, row 1
Cell in column 1, row 2	Cell in column 2, row 2

Daftar Pustaka

- [PragProg1999] Andy Hunt & Dave Thomas. The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley. 1999.
- [RustBook2023] Steve Klabnik and Carol Nichols. The Rust programming language. No Starch Press, 2023.

Lampiran A: First Appendix

A.1. Part 1 of first appendix

This is just an example of first - first appendix.

A.2. Part 2 of first appendix

And this on is an example of second - first appendix.

Lampiran B: Second Appendix

B.1. Part 1 of second appendix

This is just an example of second appendix. first subsection

B.2. Part 2 of second appendix

And this on is an example of second appendix. second subsection

Ringkasan

terminology 1

terminology no 1 is an example of glossary

terminology 2

terminology no 2 is an example of glossary

Indeks

R

Rust compilation process, [17](#)

S

source code

- formatting

 - callout, [16](#)

 - shell display, [16](#)