Full Stack Enterprise Application Developer

Materi Pembelajaran

Dr. Bambang Purnomosidi D. P.

Daftar Isi

1 Tentang Buku Ini

Versi

```
1.0.0-RC--28-Mei-2019--21:58:44
```

Buku ini merupakan buku pegangan program "**Full Stack Enterprise Application Developer**" untuk para mentor, siswa, serta pihak-pihak lain yang berkepentingan dengan proses pendidikan di Praxis Academy. Dengan panduan buku ini, diharapkan proses belajar menjadi lebih efektif dan efisien serta mempunyai pedoman yang jelas.

```
Praxis Academy
Jl.Garuda no 67, Manukan
Condong Catur
Sleman
Yogyakarta 55283
Indonesia
Email: hello@praxisacademy.id
```

```
8 Web: https//praxisacademy.id
```

2 Minggu 01

Hari 1

Tujuan

1. Memahami teknologi informasi secara umum dan kaitannya dengan software.
2. Memahami ranah pendidikan yang terkait dengan teknologi informasi: istilah teknik informatika / *informatics*, ilmu komputer / *computer science*.
3. Memahami software paling mendasar: *operating system*
4. Memahami ekosistem sistem operasi saat ini
5. Memahami komponen-komponen sistem operasi - khususnya berbasis Linux
6. Memahami UI di Linux (*GUI* dan *text mode* / *shell*)
7. Memahami dan bisa menggunakan berbagai utilitas dasar dalam Linux
8. Mehamami dasar-dasar dari *shell script*.

Pembahasan

1. Teknologi informasi dan teknologi software
2. Sistem Operasi: UI, shell, utilities, shell script

Pembelajaran

```
1 Materi dan Penjelasan
```

1. Halaman Wikipedia - Information Technology, serta komponen-komponen dari TI dengan penekanan lebih pada software.
2. Cara software menempati komputer kita serta bagaimana user berinteraksi dengan *tasks* komputasi. Tekankan pemahaman pada bagaimana sistem operasi bekerja.
3. *Overview* dari daftar sistem operasi yang ada di dunia ini.
4. Komponen-komponen SO Linux.
5. Windowing System yang berfungsi sebagai antarmuka grafis (GUI) di Linux serta impelementasinya di Linux menggunakan X Window System.

6. Keterkaitan X Window System dengan Display Manager, Window Manager, serta Desktop Environment.
7. Linux console dan keterkaitannya dengan Linux shell.
8. Ringkasan dari berbagai command shell terutama yang digunakan di Linux.
9. Beberapa utilitas yang biasanya digunakan pada saat berada di *console* / *shell*. Beberapa utilitas tersebut biasanya berada dalam kelompok GNU Core Utilities dan util-linux. Beberapa petunjuk lainnya adalah Guru99 dan E-Guide - format PDF.
10. Cara membuat shell script.

```
Latihan
```

1. Boot laptop masing-masing, perhatikan proses booting dari awal, jika muncul Grub, usahakan melihat parameter dari Grub tersebut. Cari informasi di Internet tentang parameter Grub tersebut.
2. Cari informasi tentang software yang ada di laptop anda: display manager yang digunakan, window manager yang digunakan, desktop environment yang digunakan, serta shell apa yang digunakan. Dari mana bisa mengetahui informasi tersebut?
3. Cari lokasi dari kernel Linux, sebutkan file-file yang terkait dan kegunaannya.
4. Masuk ke *terminal / comsole* dan kerjakan beberapa perintah berikut melalui *command line / shell*:
    - buat direktori $HOME/praxis/minggu-01/hari-01
    - silahkan coba beberapa perintah di Guru99 dan E-Guide - PDF.
    - tulis hasil dari masing-masing perintah tersebut ke dalam file `cmdline.txt` (gunakan copy paste dari shell)
5. Silahkan coba 30 contoh shell script, masukkan semua file-file yang dihasilkan di direktori $HOME/praxis/minggu-01/hari-01/30shellscript.

```
Kasus
```

Referensi: 1. Wikibooks - Bash Shell Scripting 2. Bash Reference Manual 3. Bash Handbook

Selesaikan kasus-kasus berikut.

0. Semua hasil disimpan pada $HOME/praxis/minggu-01/hari-01/kasus
1. Buat shell script untuk melihat daftar file pada suatu direktori (termasuk direktori anak-anaknya) dan jika terdapat file dengan ekstensi .java - tampilkan tulisan “Ada file Java pada direktori {nama direktori}”. Hasil eksekusi (misalnya):

```
$ cari.sh $HOME/src
Ada file Java pada direktori /home/bpdp/src/hari-01
```

2. Buat shell script untuk menanyakan suatu nama program (misalnya `firefox`), setelah itu mencari PID dari program tersebut dan jika PID program tersebut ada, maka program tersebut akan dimatikan. Saran: gunakan perintah-perintah `ps`, `grep`, `awk`, dan `kill`.

Hari 2

Tujuan

1. Siswa memahami peran Git dalam software engineering serta memahami keterkaitan Git dengan istilah-istilah yang biasanya digunakan:
   - software engineering
   - software configuration management
   - version control system
   - distributed version control system
2. Siswa memahami berbagai vendor yang menyediakan fasilitas untuk *remote repository*.
3. Siswa memahami dasar-dasar penggunaan Git serta GitHub dan mampu menggunakan Git serta GitHub untuk *single person development*.
4. Siswa memahami dan mampu membuat file *markdown* untuk dokumentasi di GitHub.
5. Siswa memahami dan mampu menggunakan Git serta GitHub untuk *team* pengembang aplikasi.

Pembahasan

1. Software engineering, software configuration management, version control. dan distributed version control.
2. Git dan perintah-perintah dasarnya
3. *Markdown* sebagai format untuk dokumentasi
4. Git dan *remote repository* (GitHub, GitLab, Assembla, BitBucket)
5. Git untuk single person development
6. Git untuk tim pengembang aplikasi

Pembelajaran

```
Materi dan Penjelasan
```

1. Ruang lingkup software engineering.
2. Keterkaitan software engineering dengan SCM - Software Configuration Management.
3. Keterkaitan SCM dengan Version Control dan Distributed Version Control.
4. Git dan keterkaitannya dengan Distributed Version Control
5. Men-*setup* Git - Chapter 1 - Getting Started.
6. Membuat account GitHub serta membuat repo di GitHub.
7. README.md serta mampu menggunakan pemformatan file markdown untuk menuliskan dokumentasi.
8. Penggunaan `git status`, `git add`, `git commit`, `git push` untuk menyimpan ke *remote repo*
9. Penggunaan branching and merging.
10. Pull request untuk repo di GitHub milik sendiri, merging, kemudian sinkronisasi ke lokal repo di komputer.
11. Menggunakan Github untuk kolaborasi tim.

```
Latihan
```

1. Buat repo sesuai keterangan pada panduan umum Praxis Academy untuk hari 1, masukkan semua file-file yang dibuat pada pertemuan hari pertama.
2. Praktikkan Getting Started ini bersama rekan.

```
Kasus
```

1. Praktikkan Team Collaboration with GitHub bersama rekan.

Hari 3

Tujuan

1. Siswa bisa memahami keterkaitan antara bahasa pemrograman dengan *compiler/interpreter*.
2. Siswa memahami komponen dari peranti pengembangan (*development tools*) dan bisa mencari komponen-komponen untuk suatu bahasa pemrograman tertentu.
3. Siswa memahami keterkaitan antara Java, JVM, JRE, JDK, JSE, JEE, serta JME.

4. Siswa memahami keterkaitan antara JCP (Java Community Process) dengan spesifikasi Java dan spesifikasi-spesifikasi lainnya di dunia Java.
5. Siswa memahami keterkaitan antara OpenJDK dengan JDK dari berbagai vendor (Oracle, Corretto, dll).
6. Siswa mengetahui riwayat versi dari Java.
7. Siswa mampu menginstall peranti pengembangan Java di komputer masing-masing, setidaknya untuk OpenJDK dan Orache JDK.
8. Siswa mampu menginstall Visual Studio Code serta plugin untuk peranti pengembangan Java. Siswa juga dibebaskan menggunakan editor teks maupun IDE lainnya.
9. Siswa memahami dan mampu membuat *source code* dalam bahasa pemrograman Java, mengkompilasi, serta menjalankan hasilnya - menggunakan *shell command line* maupun dengan membuat file .JAR (dengan file manifest maupun tanpa file manifest).
10. Siswa memahami struktur dasar *source code* dalam bahasa pemrograman Java.
11. Siswa memahmai dan bisa menggunakan variabel, konstanta, operator, ekspresi, statement, dan tipe data dasar di Java.
12. Siswa memahami dan bisa menggunakan perintah-perintah Java untuk mengatur alur kendali program.

Pembahasan

1. Development tools dan ekosistemnya
2. Dasar-dasar Java:
    - Kompilasi, menjalankan, mempaket dalam bentuk `.jar`.
    - *Tools dan utilities* di JDK.
    - Konstruksi dasar bahasa pemrograman Java: variabel, konstanta, operator, ekspresi, *statement* / pernyataan, tipe data.
    - Pernyataan untuk mengatur kendali program

Pembelajaran

```
1 Materi dan Penjelasan
```

1. Keterkaitan antara bahasa pemrograman, compiler, dan interpreter.
2. Komponen dari peranti pengembangan (*development tools*) dan bisa mencari komponen-komponen untuk suatu bahasa pemrograman tertentu.
3. Keterkaitan antara Java sebagai bahasa pemrograman, Java sebagai platform software, serta Java Virtual Machine.
4. Perbedaan JRE dengan JDK.

5. Edisi dari JDK: JSE, JavaFX, JEE, Java Embedded, serta JME.
6. Keterkaitan antara JCP (Java Community Process) dengan spesifikasi Java dan spesifikasi-spesifikasi lainnya di dunia Java.
7. Keterkaitan antara OpenJDK dengan JDK dari berbagai vendor - Oracle, Corretto, dll.
8. Riwayat versi dari Java.
9. Cara menginstall JDK 8 (Oracle dan OpenJDK) dari file .tar.gz (bukan dari distro) serta membuat script untuk mengatur instalasi tersebut (berisi perintah `export` env variable yang diperlukan).
10. Instalasi Visual Studio Code serta plugin untuk peranti pengembangan Java.
11. Kompilasi dan menjalankan aplikasi yang diprogram menggunakan Java, lihat juga repo GitHub ini.
12. Struktur dasar *source code* dalam bahasa pemrograman Java.
13. Penggunaan variabel, konstanta, operator - dijelaskan disini dan disini, ekspresi, statement, block, dan tipe data dasar di Java.
14. Penggunaan pernyataan-pernyataan dalam Java untuk mengatur alur kendali program: looping, pengaturan kondisi, switch.
15. Penggunaan array, multidimensional arrays, dan fungsi matematika.
16. Penggunaan String: String: karakter tunggal, split dan join,

```
Latihan
```

1. Install JDK seperti pada point nomor 7 di materi dan penjelasan di atas.
2. Install Visual Studio Code atau editor / IDE apapun yang anda sukai, konfigurasikan supaya bisa digunakan untuk menulis program Java.
3. Kerjakan point nomor 9 di materi dan penjelasan di atas.
4. Kerjakan point nomor 11 - 14 di materi dan penjelasan di atas, buat semuanya masing-masing dalam bentuk *executable* JAR.

```
Kasus
```

Referensi: Bagaimana meminta input dari user di Java?

1. Buat program untuk penjumlahan matriks. Matriks sudah didefinisikan dalam program menggunakan array multidimensi.
2. Perbaiki program penjumlahan matriks tersebut dengan nama lain. Matriks tidak didefinisikan di dalam program, tetepi merupakan hasil input dari pemakai program.

Hari 4

Tujuan

1. Siswa memahami pola pikir obyek.
2. Siswa memahami dan mampu membuat *source code* menggunakan dasar-dasar pemrograman berorientasi obyek.
3. Siswa memahami dan mampu menggunakan *package* dalam pembuatan *source code*.
4. Siswa memahami arti penting *build tool* dan mampu menggunakan Gradle untuk proses pengembangan aplikasi menggunakan Java.
5. Siswa memahami perbedaan *dynamic* dengan *static typing* serta arti penting *type system* dalam mencegah *runtime error*.
6. Siswa mampu menggunakan *static typing* di Java.
7. Siswa memahami kemungkinan terjadinya *exception* di Java serta cara menanganinya.

Pembahasan

1. Pola Pikir Obyek
2. Dasar-dasar Pemrograman Berorientasi Obyek di Java
3. Package di Java
4. *Build tool*: Gradle
5. *Dynamic typing* dan *static typing*
6. Penanganan error / *exception*

Pembelajaran

```
Materi dan Penjelasan
```

1. Konsep dan contoh dasar OOP - object, class, inheritance, interface
2. Packages di Java, serta tambahan materi ini dan materi ini
3. Build automation tool serta contoh pemanfaatan Gradle sebagai *build automation tool*.
4. Dynamic type - static typing.
5. Exception Handling. Lihat juga materi ini dan materi ini.

```
Latihan
```

1. Salah satu cara untuk menumbuhkan pola pikir obyek adalah penggunaan CRC Card. Pelajari dan buat 1 contoh sistem lain.

2. Kerjakan Java OOP Concepts Tutorial terutama pada bagian *object, class, inheritance, interface*. Buat dalam bentuk *executable .JAR file*, jelaskan dalam README di repo GitHub anda.
3. Kerjakan point 2 di *materi dan penjelasan* di atas.
4. Install Gradle dan kerjakan panduan singkat ini. Jelaskan dalam README.md.
5. Kerjakan point 5 di *materi dan penjelasan* di atas.

```
Kasus
```

Lengkapi / perbaiki *CRC Cards* untuk sistem yang sudah anda buat pada *Latihan point 1*. Implementasikan dalam bentuk source code.

Hari 5

Tujuan

1. Siswa memahami arti penting *annotations* dan mampu menggunakannya dalam *source code* dengan baik.
2. Siswa mampu menggunakan *javadoc* untuk membuat dokumentasi dari *source code*.
3. Siswa memahami arti penting dari *generics* dan mampu menggunakannya dalam *source code* dengan baik.

Pembahasan

1. Annotations
2. Javadoc
3. Generics

Pembelajaran

```
Materi dan Penjelasan
```

1. Tutorial tentang annotations: dari Oracle JDK, beginnersbook.com, jenkov.com, howtodoinjava.com.
2. Panduan Javadoc dari Oracle dan dari Wikipedia.
3. Tutorial tentang Generics: dari Oracle JDK, Gilad Bracha

```
Latihan
```

1. Kerjakan annotations examples dari Data Flair. Setelah selesai, perbaiki semua *source code* dengan anotasi untuk dokumentasi, setelah itu buat dokumentasi menggunakan Javadoc.
2. Kerjakan tutorial generics dari geeksforgeeks.org.

```
Kasus
```

1. Cari salah satu project kecil yang menggunakan Java di GitHub, buatlah dokumentasinya menggunakan Javadoc.
2. Perhatikan gist untuk generics ini. Kompilasi, buat file .JAR, jalankan. Buat penjelasan tentang generics pada source code tersebut.
3. Kerjakan java2novice.com Java Example Programs tentang Generics.

3 Minggu 02

Hari 1

Tujuan

1. Annotations
2. Object Oriented Programming Lanjut di Java

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 2

Tujuan

1. Generics
2. Functional Programming di Java

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 3

Tujuan

1. Unit Testing di Java

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 4

Tujuan

1. Serialisasi data: XML, JSON, dll
2. Struktur data di Java

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 5

Tujuan

1. Concurrent Programming di Java

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

4 Minggu 03

Hari 1

Tujuan

1. Pustaka standar dan pustaka pihak ketiga di Java
2. Proses *build* dan pengelolaan paket pustaka di Java: Gradle

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 2

Tujuan

1. Database management system
2. Akses ke database menggunakan Java: JDBC dan ORM

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 3

Tujuan

1. NoSQL dan NewSQL

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 4

Tujuan

1. RESTful endpoint
2. GraphQL

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 5

Tujuan

1. Arsitektur Software: microservices dan *full stack applicatio*
2. Framework: Dasar-dasar Micronaut

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

5 Minggu 04

Hari 1

Tujuan

1. Berpindah development tools: step-by-step
2. Studi kasus perpindahan development tools: dari Java ke peranti pengembangan lainnya (JavaScript, Kotlin, Go, PHP)
3. Dasar-dasar JavaScript dan Node.js

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 2

Tujuan

1. JavaScript

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 3

Tujuan

1. Vue.js

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```

Hari 4

Tujuan

1. Aplikasi Full stack: Micronaut + Vue.js
2. Aplikasi Full stack: Micronaut + Vue.js + Database

Pembahasan

Pembelajaran

```
Materi dan Penjelasan
```

```
Latihan
```

```
Kasus
```