Introduction to Java Programming

Chapter I - Introduction

By

Leon Andretti Abdillah

INTRODUCTION

- Syllabus
- Introduction to Rules (Component & Score)
- Introduction to Computers and Programming Languages
- OOP Concepts
- Java History, Platform, & Technology
- References

SYLLABUS

1. Introduction
2. Getting Started
3. Object-Oriented Programming
4. Language Basics – Variables & Literals, Identifiers, Keywords, and Types
5. Language Basics – Operators, Expressions, Statements
6. Flow Control
7. Arrays
8. Class Design, Advanced Class Features
9. Exceptions and Assertions
10. Text-Based Applications
11. Building Java GUIs, GUI Event Handling, GUI-Based Applications
12. Presentations

RULES [1]: (Component & Score)

- **Komponen Penilaian**

1. Ujian Harian (UH) → 015%
2. Ujian Tengah Semester (UTS) → 030%
3. Tugas Mandiri (TM) → 015%
4. Ujian Akhir Semester (UAS) → 040%
5. Total Score → 100%

- **Score Penilaian**

1. A → 085,00 – 100,00
2. B → 070,00 – 084,99
3. C → 055,00 – 069,99
4. D → 040,00 – 054,99
5. E → 000,00 – 039,99

RULES [2]

- **Kehadiran** mahasiswa pada tiap pertemuan memiliki toleransi keterlambatan 15 menit. Sedangkan untuk total pertemuan dalam 1 (satu) semester harus mencapai 80% dari total pertemuan perkuliahan. Dengan cacatan mahasiswa yang mengikuti perkuliahan harus tertera pada KRS (Kartu Rencana Studi) periode berjalan.
- Setiap **ujian** yang dilaksanakan harus dikerjakan sesuai dengan peraturannya. Pelanggaran yang terjadi dapat mengakibatkan ujian mengalami penyesuaian nilai atau bahkan dibatalkan. Dengan pertimbangan tertentu mahasiswa dapat mengikuti ujian perbaikan dengan izin dari dosen (kecuali UAS dengan syarat tertentu).
- Selama berada di **laboratorium**, mahasiswa harus menjaga kebersihan, kerapihan, dan ketertiban. Setiap selesai praktikum, mahasiswa diharuskan menyusun peralatan komputer sesuai dengan tempatnya. Mahasiswa tidak diperkenankan mengubah isi konfigurasi sistem komputer tanpa seizin dan sepengetahuan dosen pengasuh dan atau kepala laboratorium.

COMPUTER INTRODUCTION [1]

- Komputer berasal dari kata latin (*to compute*) yang berarti menghitung, mendapat *prefix* '*er*' → *computer*, yang berarti alat untuk menghitung.
- Namun sebuah komputer bukan saja sebagai alat hitung ia mempunyai kelebihan yang sangat penting, antara lain: 1) memiliki media penyimpanan (*memory*) dengan kapasistas (*capacity*) yang sangat besar, 2) dapat memroses data dengan kecepatan (*speed*) yang sangat tinggi, 3) dapat bekerja tanpa lelah/bosan/jemu (*continue*) secara berulang-ulang, 4) memiliki tingkat akurasi (*accuracy*) yang sangat tinggi, 5) dapat digunakan untuk membantu manusia dalam menyelesaikan berbagam masalah (*general purpose*) yang beragam dan kompleks dari berbagai bidang, dll.

COMPUTER INTRODUCTION [2]

- Komputer secara umum terbagi atas 3 (tiga) aspek/dimensi utama, yaitu: 1) Perangkat Keras (*Hardware*), 2) Perangkat Lunak (*Software*), dan 3) Sumber Daya Manusia (*Brainware*). Ketiga aspek tersebut harus ada agar aplikasi komputer dapat berjalan dengan baik.

COMPUTER INTRODUCTION [3]

- Perangkat Keras komputer merupakan perangkat yang secara harafiah/nyata dapat di-akses oleh panca indera manusia, seperti; dilihat, dipegang, disentuh, diraba, dll. Perangkat ini dapat dikelompokkan menjadi:

1. *Input Device*: alat yang digunakan untu memasukkan data/input ke dalam komputer, misalnya; *Keyboard, Mouse, Scanner, Microphone, Touch Screen*.
2. *Process Device*: alat yang digunakan mengolah/memproses data/input yang telah dimasukkan ke dalam komputer, misalnya; CPU (ALU & CU).
3. *Output Device*: alat yang digunakan untuk menampilkan / mengeluarkan hasil pengolahan terhadap data/input menjadi keluaran/output dengan format yang sesuai dengan keinginan *user*, misalnya; *Monitor, Printer, Flotter, Speaker*.

COMPUTER INTRODUCTION [4]

- Perangkat Sumber Daya Manusia komputer merupakan orang-orang yang berhubungan dengan koputer baik yang memberikan / memasukkan input serta dapat juga memberikan perintah kepada komputer. Ia dapat dikelompokkan menjadi:

1. *Analyst System*: orang bertanggung jawab atas uraian kemampuan dari program yang akan dibuat.
2. *Programmer*: orang yang tugasnya menerjemahkan rancangan *analyst system* menjadi suatu kode/bahasa yang dimengerti oleh komputer.
3. *Operator*: orang yang bertugas untuk mengoperasikan / memberikan perintah kepada komputer untuk mengerjakan suatu pekerjaan berdasarkan kriteria tertentu dengan meng-gunakan program aplikasi tertentu.
4. *Data Entry*: orang yang bertanggung jawab untuk memasukkan data ke dalam suatu sistem komputer.

COMPUTER INTRODUCTION [5]

- Perangkat Lunak komputer merupakan perangkat yang secara nyata tidak dapat di-akses oleh panca indera manusia, namun ia ada dan sangat penting peranannya. Ia dapat dikelompokkan menjadi :

1. Sistem Operasi (*Operating System*): DOS, UNIX, Linux, Windows, dll.
2. **Bahasa Pemrograman (*Programming Language*).**
3. Program Aplikasi (*Application Program*): Aplikasi Penggajian, Aplikasi Penjualan Barang, Aplikasi Persediaan Barang, dll.
4. Program Paket (*Package Program*): Ms. Word, Ms. Excel, Ms. Access, Ms. Power Point, dll.
5. Program Bantu (*Utility*): Norton.

PROGRAM [1]

- Program berupa kumpulan instruksi (dalam bentuk perangkat lunak) yang ditulis dengan suatu susunan atau tata cara (*syntax*) tertentu. Program merupakan suatu cara bagi *brainware* untuk memberi perintah kepada *hardware* untuk mengerjakan suatu pekerjaan dari manusia (mendapat hasil/keluaran, dapat berupa informasi, aksi, dsb). Program (seperti halnya bahasa manusia) memiliki sejumlah varian yang beragam sesuai vendors dan dengan kegunaannya.
- program
  - a set of directions telling a computer exactly what to do
- programming languages
  - languages for specifying sequences of directions to a computer
- algorithm
  - a sequence of language independent steps which may be followed to solve a problem

  (S.N. Kamin, D. Mickunas, E. Reingold)

PROGRAM [2]

- Secara umum **bahasa pemrograman** terdiri dari 5 (lima) golongan / tingkatan / generasi:
- Generasi 1: pemrograman bahasa mesin. Contohnya; hasil kompail yang hanya berupa angka 0 dan 1.
- Generasi 2: pemrograman bahasa rakitan. Contohnya; Assembler.
- Generasi 3: pemrograman prosedural / terstruktur. Contohnya; BASIC, COBOL, Pascal, C, dll.
- **Generasi 4: pemrograman visual / berorientasi objek. Contohnya; Microsoft Visual BASIC, Borland Delphi, Microsoft Visual Foxpro, JavaSwing, dll.**
- Generasi 5: pemrograman kecerdesan buatan (*Artificial Intellegence*). Contohnya; LISP, Prolog, dll.

OOP CONCEPTS [0]

- Java mendukung semua konsep utama OO termasuk: 1) ***polymorphism*** (dapat merespon berbagai situasi dengan beragam aksi), 2) ***inheritance*** (pewarisan sifat), serta 3) ***encapsulation*** yang menyem-bunyikan kompleksitas pengembangan perangkat lunak berbasis *windows* secara fleksibel. Dan konsep yang terakhir adalah ***re-use***, komponen berbasis ***windows*** baru yang bisa diciptakan secara cepat dan aman.

OOP CONCEPTS [1]: Class (a)

- Class merupakan suatu tipe data (*data type*) yang mengkapsulasi data dan *operation* pada data dalam suatu unit tunggal.
- Sebelum *object-oriented programming*, data dan *operations* (*functions*) dibuat secara menjadi elemen-elemenyang terpisah.
- Suatu *object* merupakan *instance* dari suatu *class*. Yakni, ia memiliki nilai yang berjenis suatu *class*.

OOP CONCEPTS [1]: Class (b)

- Class — kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing.
- Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi object. *Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada,* dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP).
- Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

OOP CONCEPTS [1]: Class (c)

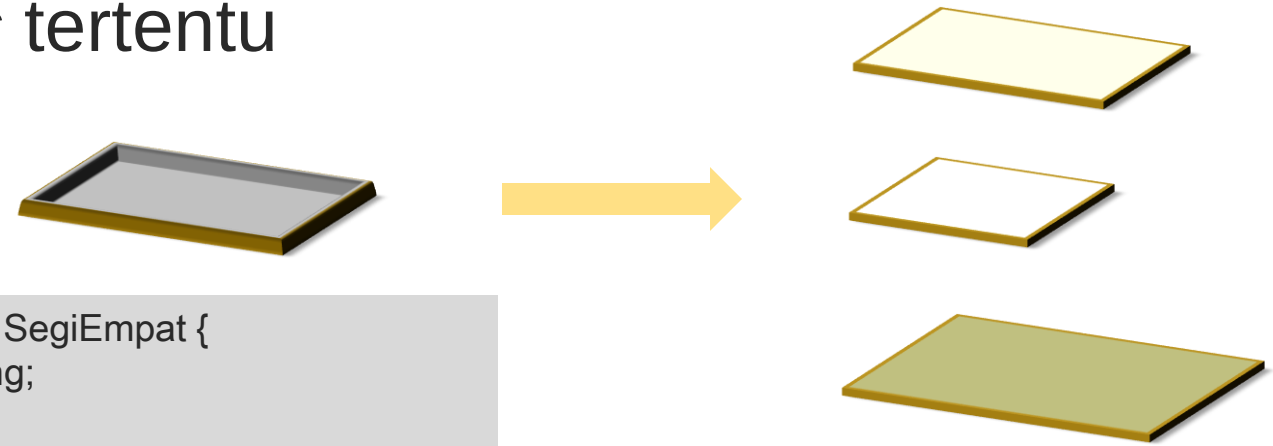
- Class merupakan “cetakan” (*template*) untuk *instance* (wujud nyata) entitas-entitas yang direpresentasikannya
  - Sebuah kelas dapat melahirkan lebih dari satu instance

OOP CONCEPTS [1]: Object (a)

- Objek merupakan representasi keberadaan sesuatu dari dunia nyata (baik secara abstrak maupun real) yang memiliki name, *property*, dan *events* (dapat juga ditambah mengandung *methods*)
- Objek - membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

OOP CONCEPTS [1]: Object (b)

Sebuah *instance* (perwujudan nyata) dari suatu *class* tertentu



```
public class SegiEmpat {
    int panjang;
    int lebar;
    string warna;

    public SegiEmpat(string w, int p, int l) {
        panjang = p;   lebar = l;   warna = w;
    }
    public setWarna (string w) {
        warna = w;  }
}
```

```
SegiEmpat sPink = new SegiEmpat("pink", 20,10);

SegiEmpat sPutih = new SegiEmpat("putih", 15,10);

SegiEmpat sOrange = new SegiEmpat("orange", 30,15);
```

OOP CONCEPTS [1]: Definisi Class & Object

```
public class SegiEmpat {
    int panjang;
    int lebar;
    string warna;

    public SegiEmpat(string w, int p, int l) {
        panjang = p;   lebar = l;   warna = w;
    }

    public setWarna (string w) {
        warna = w;
    }
}
```

field, menunjukkan atribut/property

constructor, untuk menciptakan object (instance) baru dengan property tertentu

methods atau member functions, mendeskripsikan behaviour atau aktivitas yang bisa dijalankan

```
SegiEmpat sPink = new SegiEmpat("pink", 20,10);
```

memanggil constructor untuk membentuk object baru

OOP CONCEPTS [1]: Abstraction

- Abstraksi - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

OOP CONCEPTS [1]: Encapsulation (a)

- Enkapsulasi - Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

OOP CONCEPTS [1]: Encapsulation (b)

- Enkapsulasi: lokalisasi fitur-fitur sebuah object (fields dan methods) dalam definisi object tersebut
- Enkapsulasi menyembunyikan property dan behaviour object dari pihak luar (object yang lain) → object lain melihat object ini sebagai “black box” saja
- Enkapsulasi memisahkan antara bagian publik (yang bisa dilihat oleh pihak luar (object lain) dan bagian privat (internal object itu sendiri) dengan tegas → fitur ini memberi keleluasaan/independensi untuk bekerja dengan aspek internal tanpa harus bergantung pada aspek publik/eksternal → berguna untuk menangani berbagai persoalan interoperabilitas

OOP CONCEPTS[1]: Properties (e)

- ***Properties***
- *Properties* adalah *characteristics* dari suatu *object* yang meliputi baik *visible behavior* atau *operations* dari *object*. Sebagai contoh, *Visible property* membatasi apakah suatu *object* dapat dilihat pada suatu *application interface*. *Properties* yang dirancang dengan baik menjadikan komponen-komponen Anda lebih mudah bagi yang lainnya untuk digunakan dan lebih mudah bagi Anda untuk me-*maintain*-nya.

OOP CONCEPTS[1]: Meethods (f)

- **Methods**
- *Method* merupakan *procedure* yang selalu diasosiasikan dengan suatu *class*. *Methods* mendefinisikan *behavior* (perilaku) dari suatu *object*. *Class methods* dapat meng-akses semua *properties* (*public*, *protected*, dan *private*) dan *fields* dari *class* dan secara umum direferensikan sebagai anggota *functions*.

OOP CONCEPTS[1]: Events (g)

- **Events**
- *Event* merupakan suatu *action* atau *occurrence* yang dideteksi oleh suatu program. Kebanyakan aplikasi-aplikasi yang modern dikatakan sebagai *event-driven*, karena mereka dirancang untuk merespon ke *events*. Dalam suatu program, *programmer* tidak dapat memprediksi secara pasti *sequence* dari *actions* yang akan dikerjakan oleh *user*. Sebagai contoh, *user* dapat memilih suatu menu item, *click* sebuah *button*, atau menandai sejumlah *text*. Anda dapat menulis kode untuk menangani *events* mana yang Anda minati, dibandingkan dengan menulis *code*.

OOP CONCEPTS [2]: Polymophism

- Polimorfisme melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

OOP CONCEPTS [3]: Inheritance

- Inheritas- Mengatur polimorfisme dan enskapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada - objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas.)

OOP Summary

- Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut(terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas adminiistrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek -objek yang ada karena setiap objek memiliki job descriptionnya sendiri.

Java History

- ‘Oak’ renamed ‘Java’ and ‘Web Runner’ renamed ‘Hot Java’
- Formal Announcement of Java Language and Hot Java browser in Sun World ’95
- Commercial web browsers starting with Netscape begin supporting Java
- Later Java evolved in to other technologies than Java applets (interactive web content)

REFERENCES

Abdillah, L. A. (2009a). *Object Oriented Programming*. Computer Science for Education, from http://blog.binadarma.ac.id/mleonaa/2009/06/26/oop/

Abdillah, L. A. (2009b). *Pemrograman II (Delphi Dasar)* Edisi 4. Palembang: Pusat Penerbitan dan Percetakan Universitas Bina Darma (PPP-UBD) Press.

Abdillah, L. A. (2009c). *Pemrograman III (Delphi Database)* Edisi 4. Palembang: Pusat Penerbitan dan Percetakan Universitas Bina Darma.

Kamin, S. N., Reingold, E., & Mickunas, D. (1998). *An Introduction to Computer Science Using Java. Maidenhead, Berkshire, UK: McGraw-Hill.*

Sierra, K., & Bates, B. (2005). *Head first java* (2 ed.): O'Reilly Media, Inc.

Spring, M. B. (2003). Java Programming, from http://www.sis.pitt.edu/spring/courses/is1090syl042.html

Wu, C. T. (2006). *An introduction to Object-oriented programming with Java* (fourth ed.). New Delhi: Tata McGraw-Hill Publishing Company Limited.

Xu, T. Programming in Java, from http://web.cse.ohio-state.edu/~neelam/courses/45923/