Pemrograman Berorientasi Objek

Pertemuan 1: Pendahuluan

Tentang Saya

- Nama lengkap: Intan Yuniar Purbasari, S.Kom, M.Sc.
- Kontak:
  - Email: intan.yuniar@gmail.com
  - WA: 083857716113
- MK :
  - Algoritma & Pemrograman
  - Pemrograman Berorientasi Objek
  - Struktur Data
  - Kecerdasan Buatan
  - Data Mining

Tentang Mata Kuliah ini

- IF141105 – Pemrograman Berorientasi Objek
  - Dahulu: Bahasa Pemrograman 2-OOP
- 4 SKS:
  - 3 x 50 menit = 2 jam 30 menit (tatap muka)
  - 1 x 100 menit sesi praktikum
  - 1 sks = (50’ Tatap Muka + 50’ Penugasan Terstruktur + 50’ Belajar Mandiri)/Minggu
- Prasyarat: tidak ada
- Nilai kelulusan: C-
- Materi kuliah: http://elearning.upnjatim.ac.id → Pemrograman Berorientasi Objek dosen Intan Yuniar Purbasari

Background Knowledge

- Algoritma & Pemrograman (IF141102)
  - Alur kontrol (loops; if-then)
  - Array
  - Method/fungsi dan passing parameter dalam fungsi
- Software Raptor (http://raptor.martincarlisle.com/)
- Java programming knowledge (http://www.java.com) dan dokumentasinya
- Text Editor Software (apa saja: notepad, wordpad, vi (under linux), Jcreator, NetBeans, …)
  - Recommended: Eclipse (http://www.eclipse.org)

Tujuan Mata Kuliah

- Mahasiswa dapat:
  - menjelaskan konsep perancangan Pemrograman Berorientasi Obyek
  - menjelaskan dan mendeskripsikan fungsi konstruktor dan destruktor dalam konsep pemrograman berorientasi obyek
  - mengimplementasikan konsep *class* dan *obyek*, serta *constructor* dan *destructor* dalam pemrograman
  - menjelaskan konsep *overloading* dan mendeskripsikannya dalam Pemrograman Berorientasi Obyek
  - menjelaskan konsep *inheritance* dan mendeskripsikannya dalam Pemrograman Berorientasi Obyek
  - menjelaskan dan mendeskripsikan konsep *Pointer* dan *Virtual Function* dalam konsep pemrograman berorientasi obyek
  - merancang pemrograman dengan menggunakan paradigma Berorientasi Obyek

Referensi

- Bruce Eckell (2006) , “Thinking in Java, 4th Edition”, Prentice-Hall, 2006
  - Download 3rd Edition: http://www.cs.ust.hk/~dekai/library/ECKEL_Bruce/
- Balagurusamy, “Object Oriented Programming with C++”, Mc Graw-Hill, 1995
- Rudolv Pecinovsky – Eva (2013), “OOP: Learn Object Oriented Thinking and Programming”, Eva & Thomas Brucker Publising, 2013, ISBN-13 97888090466197, http://pub.Brucker.cz/titles/oop.
- Matthias Felleisen, et al (2011), “How to Design Classes”, http://www.ccs.edu/home/matthias/htdc.html
- S. Demeyer, S. DUcasse, O. Nierstrasz (2008), “Object Oriented Reengineering Pattern”, Elsevier Science 2008, ISBN-13 9783952334126, http://scg.unbe.ch/download/oorp/. .
- Stefen F. Lott (2009), “Building Skills in Object-Oriented Design”, http://www.itmaubehack.com/homepage/books/oodesign.html
- Thomas Kuhne, “A Functional Pattern System for Object-Oriented Design”, Verlag Dr. Kovac, ISBN/ASIN 3860647709, http://homepages.mcs.vuw.ac.nz/~tk/fps/

KONTRAK KULIAH

- Toleransi keterlambatan 30 menit berlaku bagi dosen dan mahasiswa
- Pakaian SOPAN, berkerah, bebas rapi dan bersepatu.
- Tugas dikumpulkan sesuai deadline masing-masing tugas. Keterlambatan pengumpulan : minus 5% dari nilai (per hari keterlambatan), tugas dikumpulkan sebelum kuliah dimulai
- NILAI :
  - UTS++ terdiri dari:

Kehadiran	: 10%
Tugas	: 50%
UTS	: 40%

  - UAS++ terdiri dari:

Kehadiran	: 10%
Tugas	: 10%
Final Project	: 40%
UAS	: 40%

$$\text{Nilai Akhir} = \frac{UTS+UAS}{2}$$

- Untuk dapat lulus mata kuliah ini, semua komponen nilai di atas tidak boleh kosong

Materi Kuliah

- Review materi Algoritma & Pemrograman
- Konsep dasar pemrograman berorientasi objek
- Konsep objek dan class
- Pemrograman berorientasi objek dalam Raptor
- Fungsi konstruktor (dan destruktor)
- Fungsi overloading
- Konsep pewarisan *(inheritance)*
- Polymorphisme dan virtual function
- Tambahan materi:
  - Debugging
  - File handling
  - Error handling dan exception
  - Abstract class dan interface
  - Desain pustaka

Satuan Acara Perkuliahan

Minggu ke-	Topik Pokok Bahasan
1	Pendahuluan : • Kontrak Perkuliahan • Review konsep variable, Tipe data, dan pengendali program, fungsi & prosedur, array dalam C • Latar belakang OOP

2-3	• Pengenalan bahasa pemrograman Java
	• Konsep pemrograman struktural
	• Contoh studi kasus
	• Konsep dasar pemrograman berorientasi objek (abstraksi data dan enkapsulasi, pewarisan, polymorphism)
	• Konsep class dan objek:
	  • Definisi Class
	  • Definisi Object
	  • Definisi Method
	  • Desain class dan objek dalam Raptor

4	Implementasi class dan objek dalam Java • Implementasi data dan fungsi dalam class • Contoh studi kasus

Satuan Acara Perkuliahan

Minggu ke-	Topik Pokok Bahasan
5	Konsep dan fungsi Constructor dan Destructor dalam class Implementasi fungsi Constructor pada class dalam Java
6	Konsep dan fungsi overloading dalam class Implementasi fungsi overloading dalam Java
7	Pengenalan software tool Greenfoot Release tugas final project
8	UTS
9	Bahas UTS Debugging program dengan Eclipse

Satuan Acara Perkuliahan

Minggu ke-	Topik Pokok Bahasan
10	Konsep dan fungsi inheritance/pewarisan dalam class Implementasi fungsi inheritance dalam Java
11	Virtual function dan Polymorphism
12	Presentasi progress report final project
13	Tambahan materi: • File handling di dalam Java • Error handling dan exception dalam Java
14	Tambahan materi: • *Abstract class* dan Interface • Pengantar desain pustaka (library)
15	Demo final project
16	UAS

Review Materi Algoritma & Pemrograman

- Variabel dan tipe data
- Kontrol Alur
- Fungsi/Method
- Array

Variabel dan tipe data C

- Bilangan bulat → integer

Tipe	Rentang Nilai	Format
byte (unsigned char)	0 … 255	Unsigned 1 byte
shortint (signed char)	-128 … 127	Signed 1 byte
word (unsigned int)	0 … 65535	Unsigned 2 bytes
integer (int)	-32768 … 32767	Signed 2 bytes
long int	-2177483648 … 2177483647	Signed 4 bytes

- Operasi aritmetika dan perbandingan pada integer:

Aritmetika	Perbandingan
x=3+10; //Hasil=13	3<8; //Hasil=true (1)
x=87-31; //Hasil=56	74>101; //Hasil=false (0)
x=5*10; //Hasil=50	9<=9; //Hasil=true (1)
x=5/2; //Hasil=2	9<9; //Hasil=false (0)
x=5%2; //Hasil=1	17==17; //Hasil=true (1)

Variabel dan tipe data C

- Bilangan pecahan/floating-point → float

Tipe	Rentang Nilai	Format	Ketepatan (di belakang koma)
float	$1.2x10^{-38}$ … $3.4x10^{38}$	4 bytes	6 angka
double	$2.3x10^{-308}$ … $1.7x10^{308}$	8 bytes	15 angka
long double	$3.4x10^{-4932}$ … $1.1x10^{4932}$	10 bytes	19 angka

- Operasi aritmetika dan perbandingan pada float:

Aritmetika		Perbandingan	
x=3.2+10.1;	//Hasil=13.300000	3.2<10.1;	//Hasil=true (1)
x=8.0-3.12;	//Hasil=13.120000	7.4>10.1;	//Hasil=false (0)
x=5.0*10.0;	//Hasil=50.000000	9.0<=9.0;	//Hasil=true (1)
x=10.0/3;	//Hasil=3.333333	9.1<9.0;	//Hasil=false (0)

Variabel dan tipe data C

- Karakter → char
  - Semua huruf abjad kecil dan besar, semua tanda baca, angka ‘0’, ’1’,…, ’9’, karakter khusus (‘&’, ’^’, ‘%’, ‘#’, dll)
  - Penulisannya diapit dengan petik tunggal atau *single quote* (‘’)
  - Karakter kosong *(null)*: karakter dengan panjang nol dan dilambangkan dengan ‘’
  - Operasi perbandingan pada karakter:

Perbandingan	
‘a’ == ‘a’	//Hasil=true (1)
‘T’ == ‘t’;	//Hasil=false (0)
‘y’ != ‘Y’;	//Hasil=true (1)
‘m’ < ‘z’;	//Hasil=true (1)

Variabel dan tipe data C

- Sekumpulan karakter akan membentuk sebuah string
- Bahasa C tidak memiliki tipe khusus untuk menampung sebuah string
- String dapat disimpan dalam *array of characters*
- Penulisan sebuah string diapit oleh petik ganda atau *double quotes* (“”)
- Contoh:
  - `char kota[] = “Surabaya”;`
- Operasi pada string meliputi operasi penyambungan *(concatenation)* dan perbandingan
  - Contoh *concatenation*: “kota” + “Surabaya” = “kotaSurabaya”
  - Contoh perbandingan:
    - “abcd” > “acbd” → Hasil: false (0)
    - “aku” < “AKU” → Hasil: true (1)

CONTROL STRUCTURES

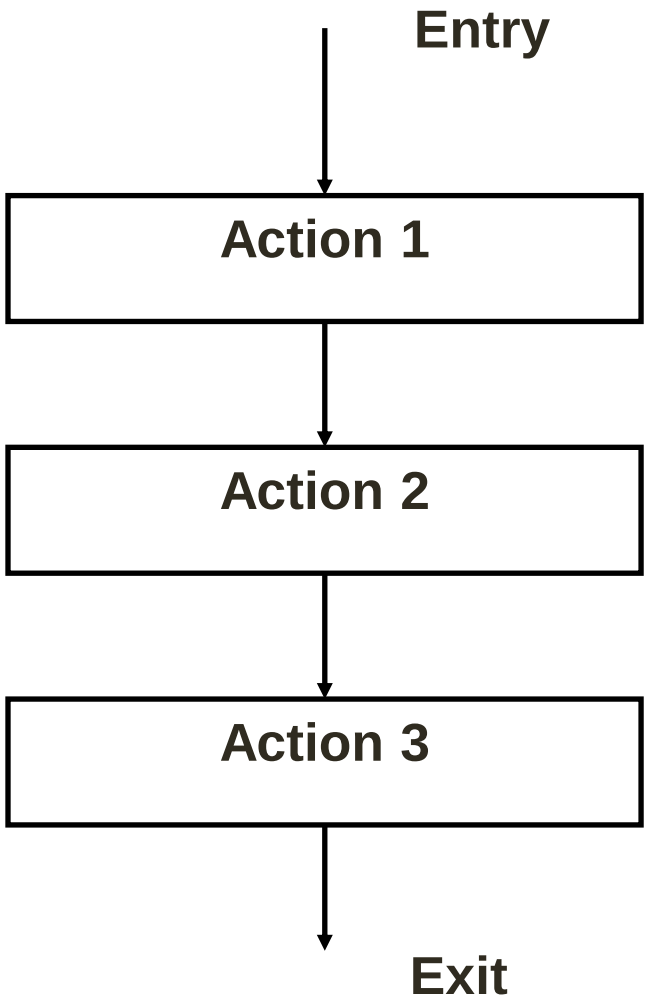
- **SEQUENTIAL/ URUTAN**

  Program dijalankan mulai dari perintah paling atas/ awal sampai paling akhir secara berurutan/ sekuensial.
- **BRANCHING/ PERCABANGAN**

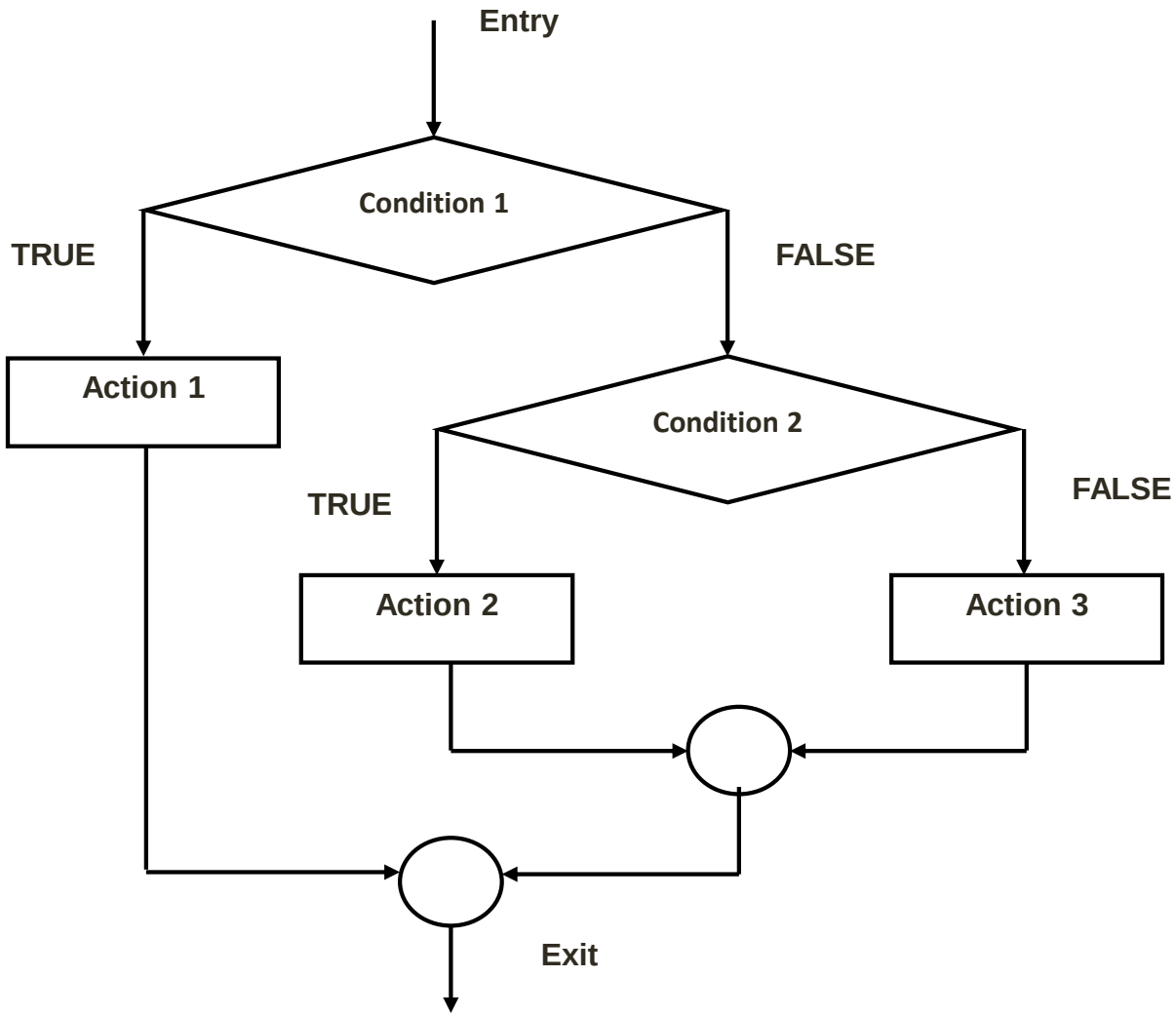
  Penyeleksian kondisi (TRUE/ FALSE) untuk menentukan statemen selanjutnya
- **LOOPING/ PERULANGAN**

  Mengulangi proses selama syarat/ kondisi tertentu masih terpenuhi

Sequential/Urutan



Branching


Entry
Condition 1
TRUE
FALSE
Action 1
Condition 2
TRUE
FALSE
Action 2
Action 3
Exit

Percabangan: IF dan IF-ELSE

- Perintah yang digunakan adalah `if` dan `if-else`
- Bentuk umum:
  - Bentuk `if`

```
if (kondisi) {
  //lakukan sejumlah action disini jika kondisi bernilai true
}
```

  - Bentuk `if-else`

```
if (kondisi) {
  //lakukan sejumlah action disini jika kondisi bernilai true
}
else {
  //lakukan sejumlah action disini jika kondisi bernilai false
}
```

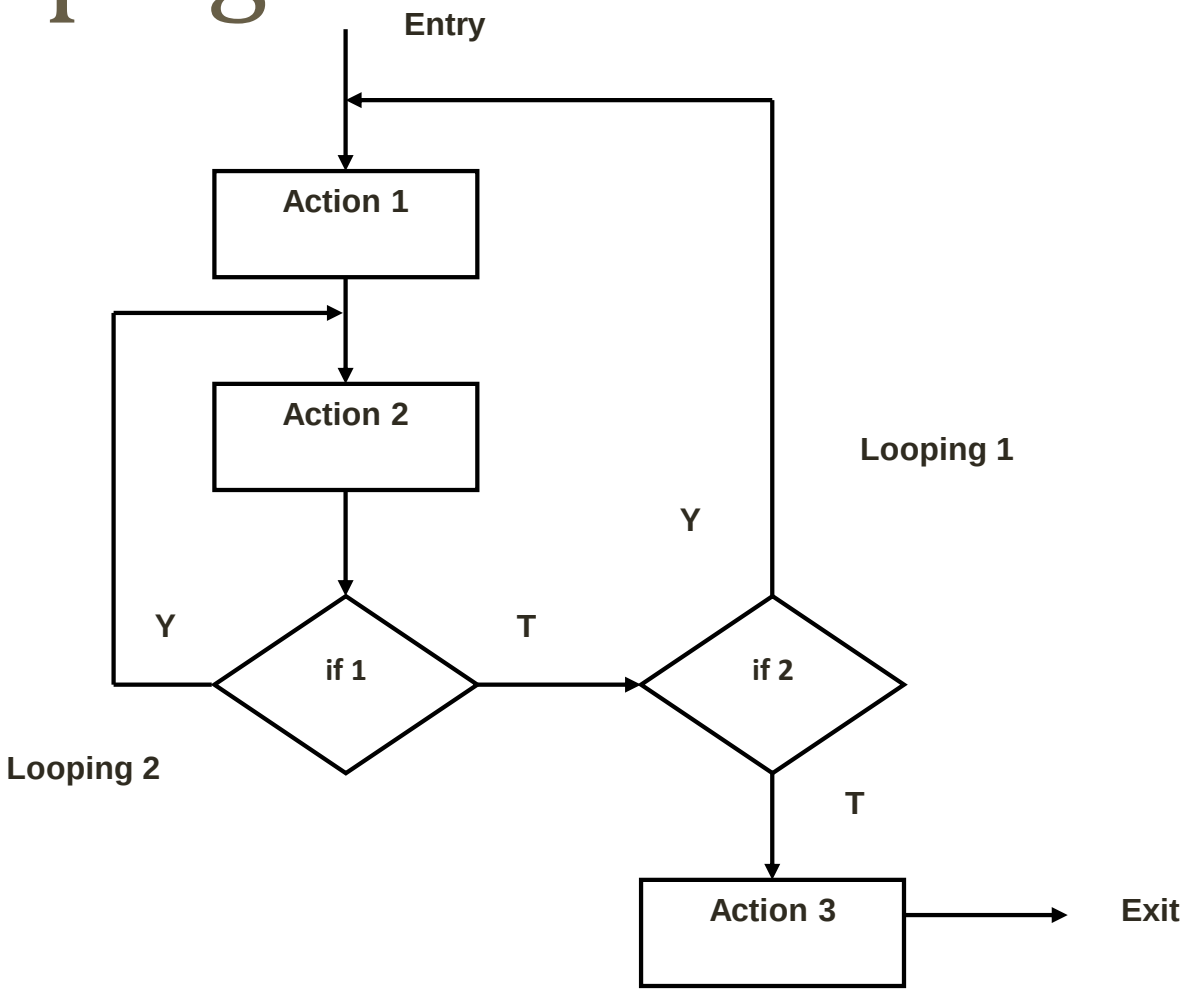
Percabangan: SWITCH-CASE

- Perintah `switch-case` digunakan pada percabangan dengan 2 kasus atau lebih untuk menyederhanakan penulisan `if-else` yang bertingkat
- Bentuk umum:

Catatan: nilai_1, nilai_2, dst adalah nilai yang bertipe int atau char

```
switch (ekspresi) {
    case nilai_1://lakukan sejumlah action di sini
                 break;
    case nilai_2://lakukan sejumlah action di sini
                 break;
    …
    case nilai_n://lakukan sejumlah action di sini
                 break;
    default: //berfungsi sebagai else dalam if
}
```

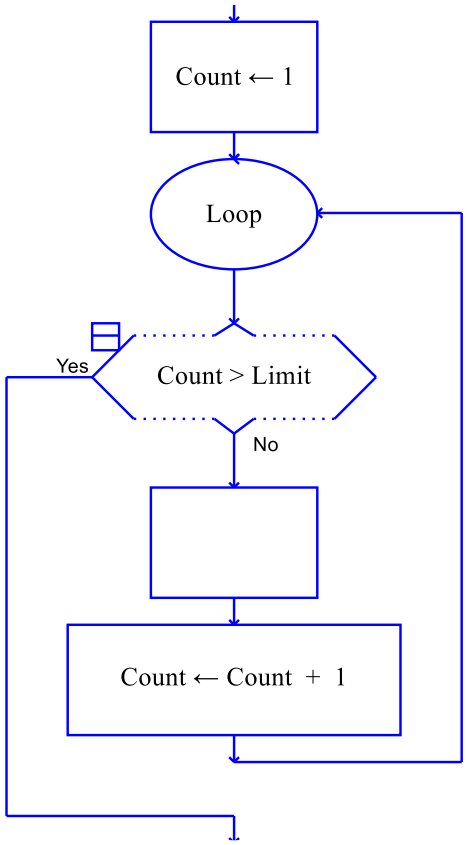
Looping


Entry
Action 1
Action 2
Looping 1
Y
Y
T
if 1
if 2
Looping 2
T
Action 3
Exit

Perulangan Dengan WHILE

- Pernyataan ini akan mengulang satu atau beberapa pernyataan, jika masih memenuhi kondisi. Pengecekan kondisi/syarat perulangan dilakukan di awal iterasi.
- Bentuk umum:

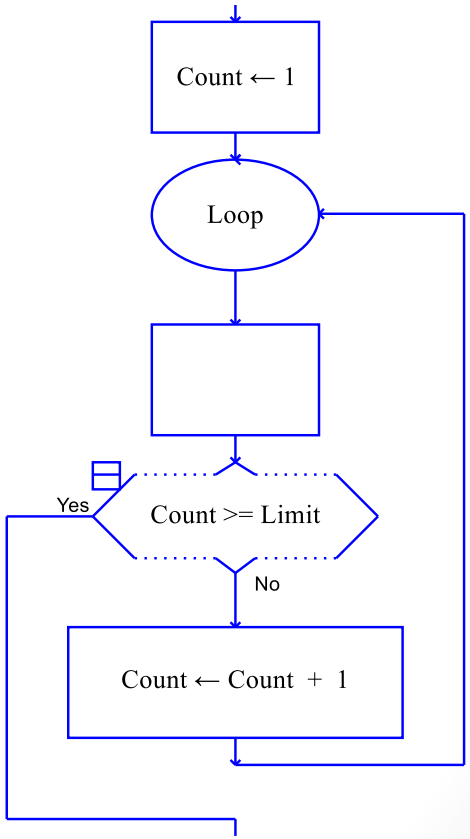
```
while (kondisi){
    //lakukan sejumlah action
    //disini jika kondisi
    //bernilai benar
}
```



Perulangan Dengan DO-WHILE

- Seperti halnya while, perintah ini menyatakan perulangan proses selama kondisi tertentu. Pengecekan kondisi/syarat perulangan dilakukan di akhir iterasi.
- Dengan menggunakan do-while, sebuah statement dieksekusi setidaknya 1 kali
- Bentuk umum:

```
do {
   //lakukan sejumlah action disini jika
   //kondisi bernilai benar
}
while (kondisi)
```



Perulangan Dengan FOR

- Perintah FOR melakukan hal yang sama dengan perintah perulangan yang lain, tetapi di awal perulangan terdapat deklarasi range dari perulangan yang akan dilakukan
- Perulangan dengan FOR dilakukan jika jumlah yang perulangan akan dilakukan telah diketahui sebelumnya
- Bentuk umum:

```
for (inisialisasi_cacah;kondisi;perubahan_nilai_cacah){
        //lakukan sejumlah action disini selama kondisi
        //bernilai benar
}
```

Prosedur/Fungsi/Method

- Merupakan pengelompokan beberapa instruksi/baris program yang melakukan sebuah perhitungan/komputasi tertentu
- Prosedur dapat menerima **parameter input** dan menghasilkan/mengembalikan **parameter output**
  - **Parameter input** : variabel yang diberikan pada prosedur untuk diproses di dalam prosedur
  - **Parameter output** : disebut juga nilai kembalian *(return value),* yakni variabel yang merupakan "output" dari prosedur
- Sebuah prosedur boleh tidak memiliki parameter input dan/atau parameter output

Prosedur/Fungsi/Method

- Prosedur tidak dapat berdiri sendiri dalam sebuah program, ia harus dipanggil oleh program lain (dapat oleh program utama atau prosedur lainnya)
- Prosedur umumnya ditulis sebelum bagian `main()` dari program
- Cara pemanggilan prosedur adalah dengan menuliskan namanya
  - Jika prosedur memiliki parameter input, maka parameter input disertakan dalam pemanggilan prosedur
  - Jika prosedur memiliki parameter output, maka perlu disiapkan **tempat penampung** (dalam bentuk variabel) untuk menerima output dari prosedur
- Tidak ada ketentuan wajib untuk penamaan sebuah prosedur, namun biasanya menggunakan kata kerja.

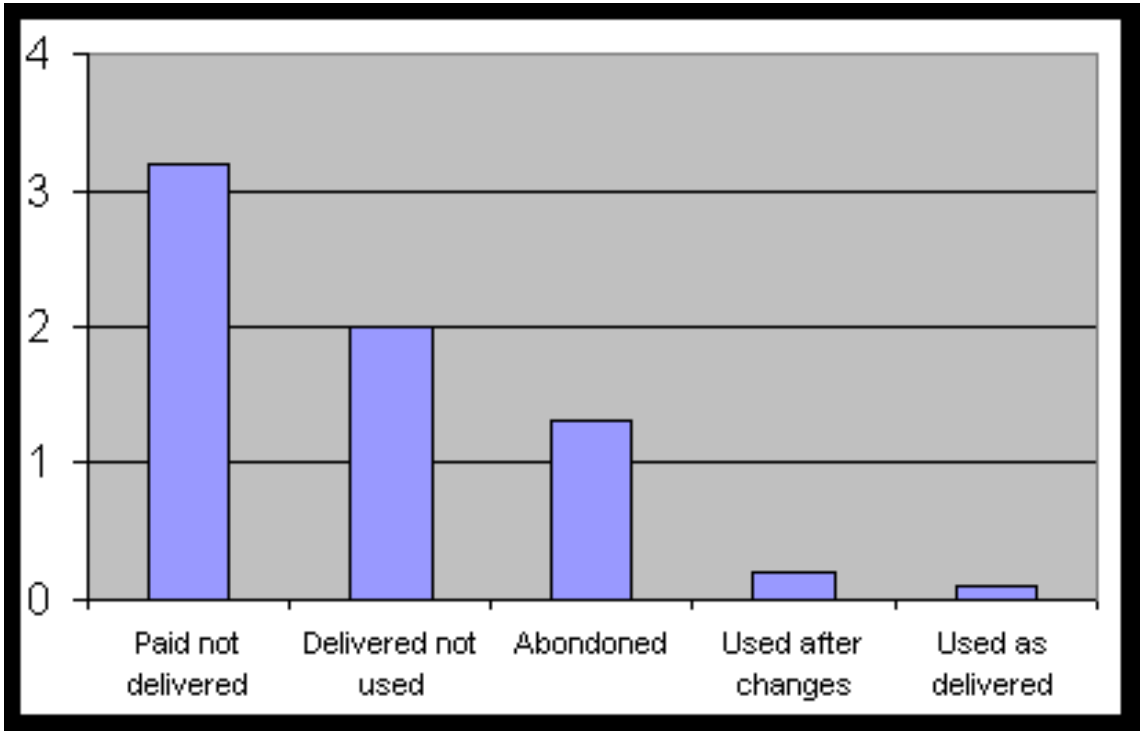
Latar Belakang OOP

- Krisis software
  - Perkembangan software sangat dinamis, tool dan teknik baru terus bermunculan
  - Para *software engineers* dan *software industry* berlomba-lomba mencari pendekatan baru terhadap desain dan pembuatan software
  - Beberapa permasalahan di dalam krisis ini:
    - Bagaimana merepresentasikan permasalahan entitas dunia nyata ke dalam desain sistem?
    - Bagaimana cara mendesain sistem dengan antarmuka yang terbuka *(open interface)?*

Latar Belakang OOP

- Bagaimana memastikan penggunaan modul yang dapat digunakan ulang (aspek ***reusability***) dan diperluas fungsinya (aspek ***extensibility***)?
- Bagaimana membangun sebuah model yang dapat mentoleransi perubahan apapun di masa depan?
- Bagaimana untuk meningkatkan produktivitas software dan menurunkan biaya software?
- Bagaimana untuk meningkatkan kualitas software?
- Bagaimana untuk mengindustrialisasikan proses pembuatan software?

Latar Belakang OOP

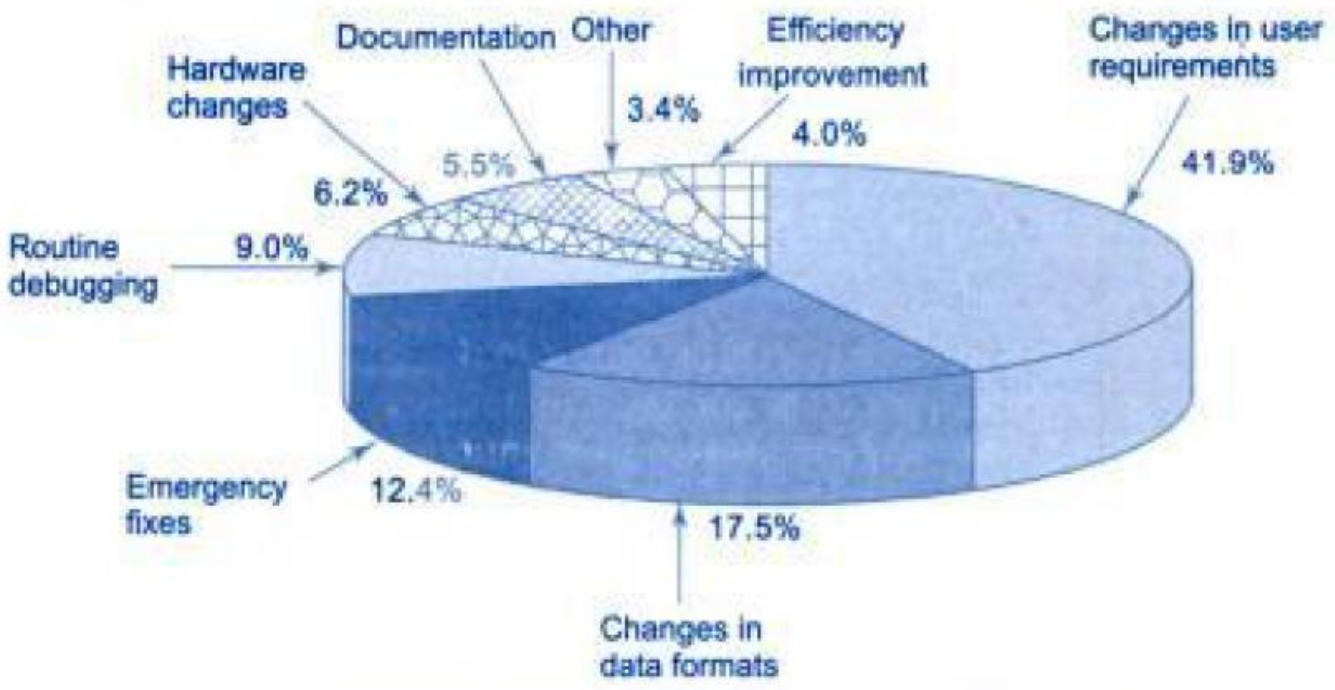


Data sampel nilai proyek software (dalam juta $) di Dephan Amerika Serikat tahun 1970-an

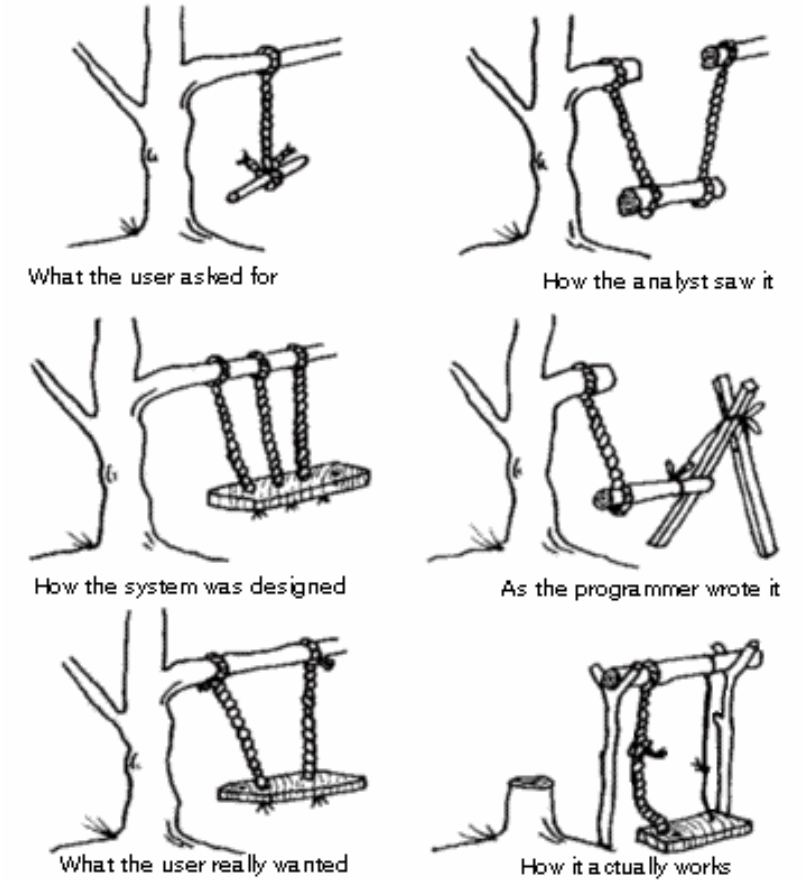
- Sekitar 50% proyek tidak sampai ke klien
- Sepertiga dari yang sampai ke klien, tidak pernah dipakai
- Hanya 2% yang dipakai klien, tanpa ada perubahan apapun

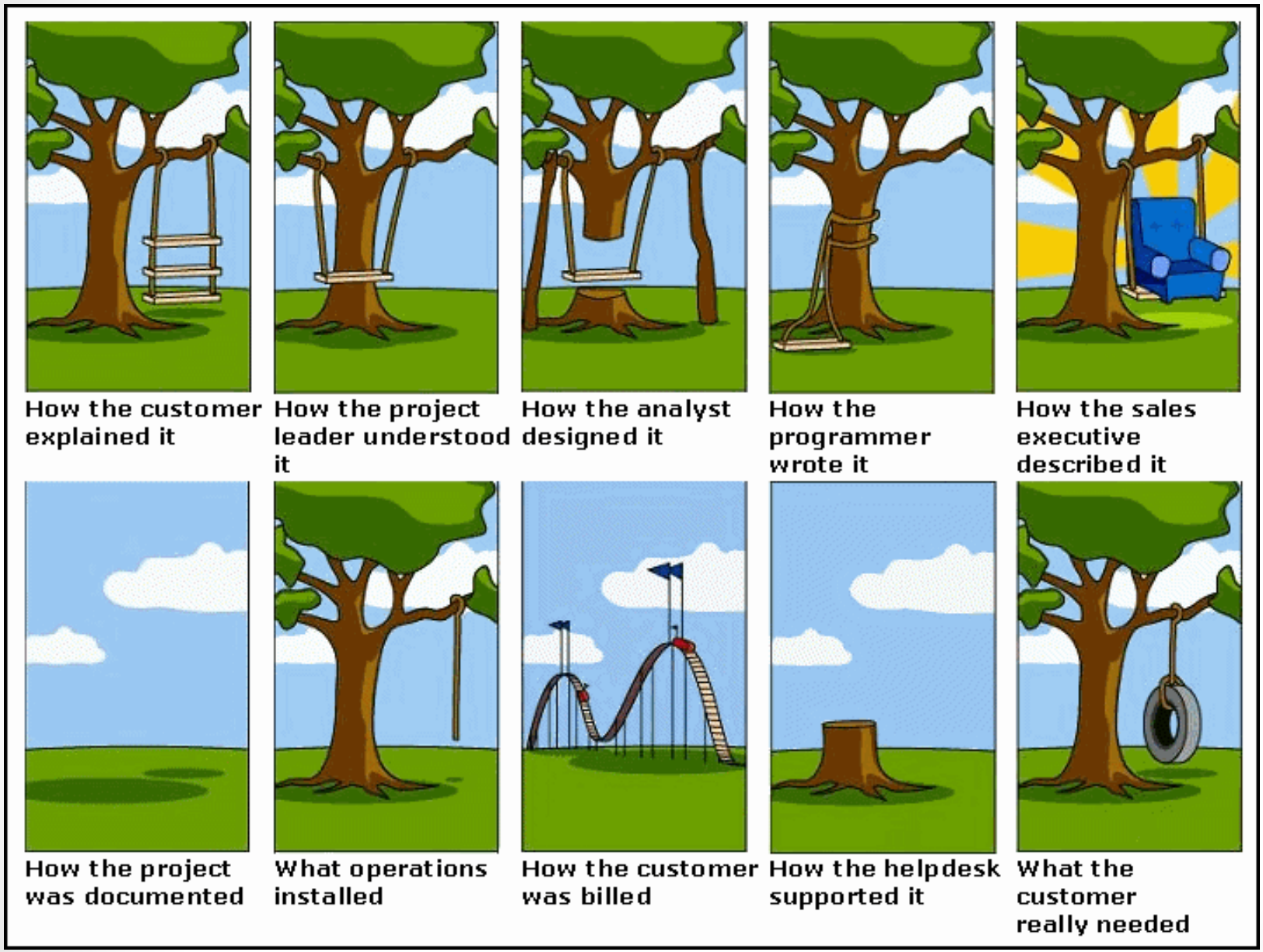
Latar Belakang OOP

- Perubahan pada *user requirement* selalu merupakan sebab utama permasalahan
- Lebih dari 50% sistem memerlukan modifikasi karena perubahan *user requirement* dan format data

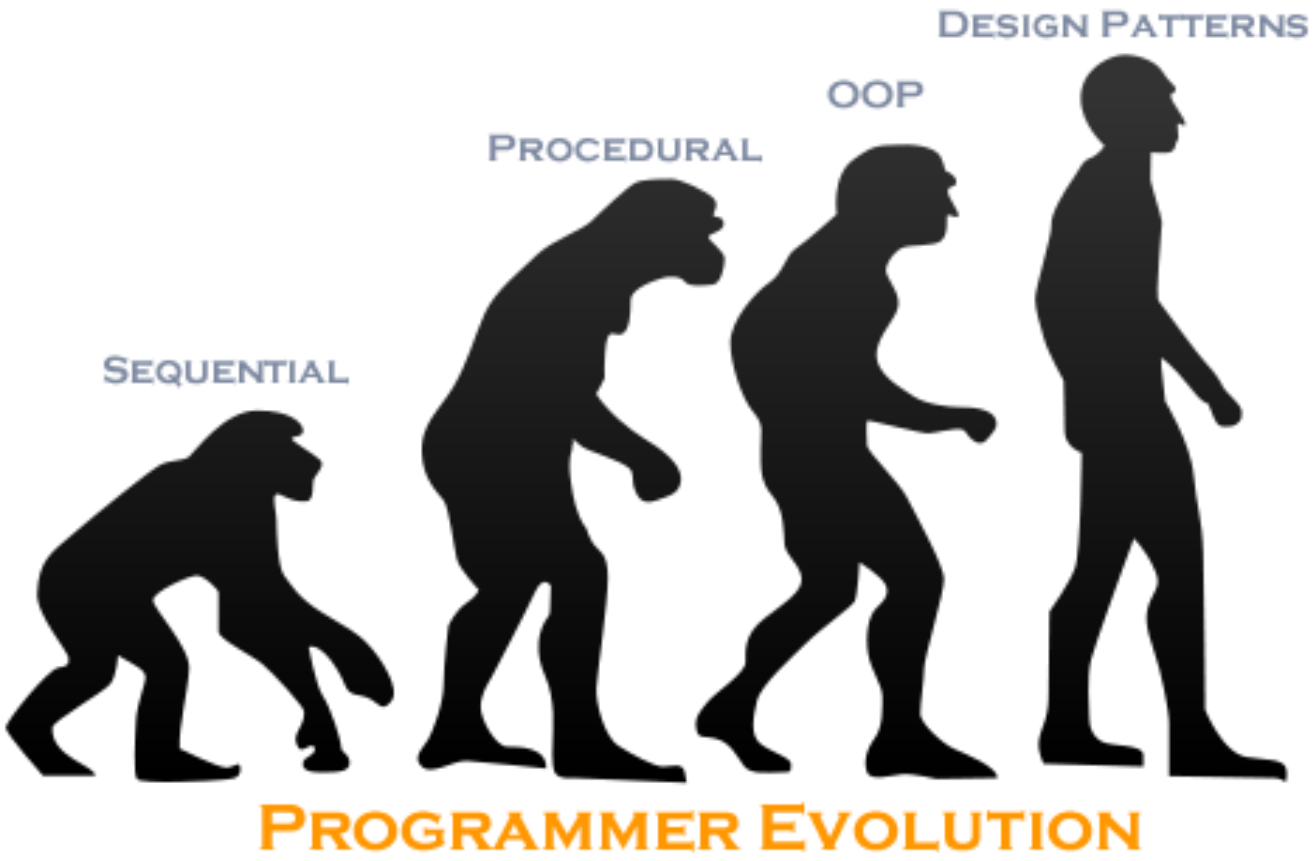


- Permasalahan utama pada pembuatan software yang sukses adalah bagaimana menterjemahkan *user requirement* ke spesifikasi software yang tepat




How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed

Evolusi Software



Design Pattern: a general repeatable solution to a commonly occurring problem in software design

Minggu depan

- Pengenalan bahasa pemrograman Java
- Konsep pemrograman struktural
- Contoh studi kasus
- Konsep dasar pemrograman berorientasi objek (abstraksi data dan enkapsulasi, pewarisan, polymorphism)