Bahasa Pemrograman Singkong

Antara Passion dan Kolaborasi Riset

Dr. Noprianto

singkong.dev (PT. Stabil Standar Sinergi)

Agenda

- Antara Bahasa dan Pengembang Program
- Kenapa Singkong
- Passion dan Kolaborasi
- Lampiran: contoh langkah demi langkah: GUI
- Lampiran: contoh langkah demi langkah: Database dan GUI

Antara Bahasa dan Pengembang Program
Bahasa Pemrograman

```
reset()
var e = component("text", "")
var b = component("button", "Halo")
add(e)
add_s(b)

event(b, fn() {
  message(get(e, "contents"))
})

show()
```


| Kode sumber program | Melewati proses translasi | Menjadi program yang dapat dijalankan |

Antara Bahasa dan Pengembang Program
Hello, World

```c
#include<stdio.h>

int main(void) {
        printf("Hello, World\n");
        return 0;
}
```
C

```java
class Hello {
        public static void main(String[] args) {
                System.out.println("Hello, World");
        }
}
```
Java

```python
print('Hello, World')
```
Python

```
println("Hello, World")
```
Singkong

Antara Bahasa dan Pengembang Program
Alat Bantu: Tidak Semua Kode Program Harus Diketik

```
class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

Java

- **Menggunakan NetBeans**
- New -> Java Class -> Hello
- Di dalam kode class Hello yang dihasilkan:
  - Ketik psvm
  - Tekan <control> <space>
  - Pilih: *public static void main*
  - Tekan Enter
- Dalam kode yang dihasilkan, ketik:
  - System.out.println("Hello, World");
  - **Hanya mengetik 1 baris kode** (dibantu auto complete)

- **Menggunakan NetBeans**
- New -> JFrame Form -> HelloWorld
- Pada Frame yang tampil, drag Button dari Palette
- Ubah text Button di Properties
- Run File
- **<u>Tanpa</u> mengetik 1 baris kode pun**

Antara Bahasa dan Pengembang Program
Perkembangan: Bahasa-Bahasa Baru Lahir Setiap Dekade

Periode	Jumlah Bahasa Baru	Contoh
Sebelum 1950	>10	
1950-an	~50	FORTRAN, COBOL, LISP
1960-an	~50	BASIC
1970-an	~60	Pascal, C, SQL
1980-an	~60	C++, Perl
1990-an	~70	Python, Java, PHP, JavaScript
2000-an	~50	C#, Go
2010-an	~30	Dart, Swift

'Timeline of programming languages' (2021). Wikipedia. Available at: https://en.wikipedia.org/wiki/Timeline_of_programming_languages (Accessed: 05 March 2021)

Antara Bahasa dan Pengembang Program
Indeks TIOBE: Bahasa-Bahasa Pemrograman Terpopuler (1-10)

Feb 2021	Feb 2020	Bahasa
1	2	C
2	1	Java
3	3	Python
4	4	C++
5	5	C#
6	6	Visual Basic
7	7	JavaScript
8	8	PHP
9	9	SQL
10	12	Assembly

Antara Bahasa dan Pengembang Program
Indeks TIOBE: Bahasa-Bahasa Pemrograman Terpopuler (11-20)

Feb 2021	Feb 2020	Bahasa
11	13	R
12	26	Groovy
13	11	Go
14	15	Ruby
15	10	Swift
16	16	MATLAB
17	18	Delphi / Object Pascal
18	22	Classic Visual Basic
19	19	Perl
20	20	Objective-C

Antara Bahasa dan Pengembang Program
Ruang untuk Bahasa Pemrograman Baru

- Menawarkan kelebihan dari bahasa-bahasa yang ada

- **Untuk mendukung end-user programming: kebutuhan personal/hobi atau menyelesaikan pekerjaan**

- Berbagai bahasa yang spesifik untuk domain tertentu (DSL)

- **Mengkombinasikan kelebihan beberapa bahasa ke dalam bahasa yang lebih sederhana**

Kenapa Singkong
Belajar dan Menggunakan Beberapa Bahasa Pemrograman

- Pascal, C, PHP, Python, Java

- Belajar merancang dan mengimplementasikan bahasa domain-spesifik:

  - Perkedel

  - Pangsit

- **Singkong: akhir 2019-sekarang**

> Singkong terinspirasi dari tanaman singkong: tersedia meluas, dapat diolah menjadi berbagai jenis makanan atau dimakan apa adanya, dan terjangkau oleh hampir siapa pun.

Kenapa Singkong
Kebutuhan: Jalan di Sebanyak Mungkin Sistem Operasi (1)

- Harus dapat jalan di sebanyak mungkin sistem operasi

  - macOS: dari Mac OS X 10.4 sampai terbaru

  - Windows: dari Windows 98 sampai terbaru

  - Linux: yang dirilis awal tahun 2000-an sampai terbaru

  - Chrome OS: sejak tersedia Linux development environment

  - Solaris

  - FreeBSD

  - OpenBSD

  - NetBSD

  - Android (lewat Termux)

Sekali ditulis, program yang ditulis dengan Singkong dapat jalan di sebanyak mungkin sistem operasi

Kenapa Singkong
Kebutuhan: Jalan di Sebanyak Mungkin Sistem Operasi (2)

macOS	14, 11, 10.15, 10.14, Mac OS X 10.4
Windows	11, 10, 7, XP, 2000, 98
Linux	Ubuntu (22.04, 20.04, 18.04, 16.04, 4.10), Raspberry Pi OS, Red Hat Linux 7.3, Debian 10 di Android, Termux / Android
Chrome OS	101 (Linux development environment)
Solaris	11.4
FreeBSD	13.0, 12.1
OpenBSD	7.0, 6.6
NetBSD	9.2, 9.0

Noprianto. (2025). *Mengenal dan Menggunakan Bahasa Pemrograman Singkong.* Jakarta: PT. Stabil Standar Sinergi, pp.4

Agar dapat dijalankan pada sebanyak mungkin sistem operasi, Singkong ditulis dengan **Java** (dan Singkong)

Singkong hanya membutuhkan Java 5.0 (dan telah diuji pada Java versi 21 pada saat presentasi ini disesuaikan)

Java 5.0 dirilis **2004**, **15 tahun sebelum Singkong** mulai dikembangkan (akhir 2019)

Kenapa Singkong
Kebutuhan: Sintaks Sesederhana Mungkin

- Prosedural, _tidak_ berorientasi objek

- Tidak membedakan huruf besar dan huruf kecil

```
var name = "Singkong"
println(NAME)
```

- Mendukung fungsi rekursif, first class function, fungsi dalam fungsi

```
var f = fn(x) {
   if (x == 1) {
      1
   } else {
      x * f(x-1)
   }
}
```

```
var f = fn(x) {
   return x
}

var g = fn(x) {
   return x * x
}

var h = f(g)(10)
println(h)
```
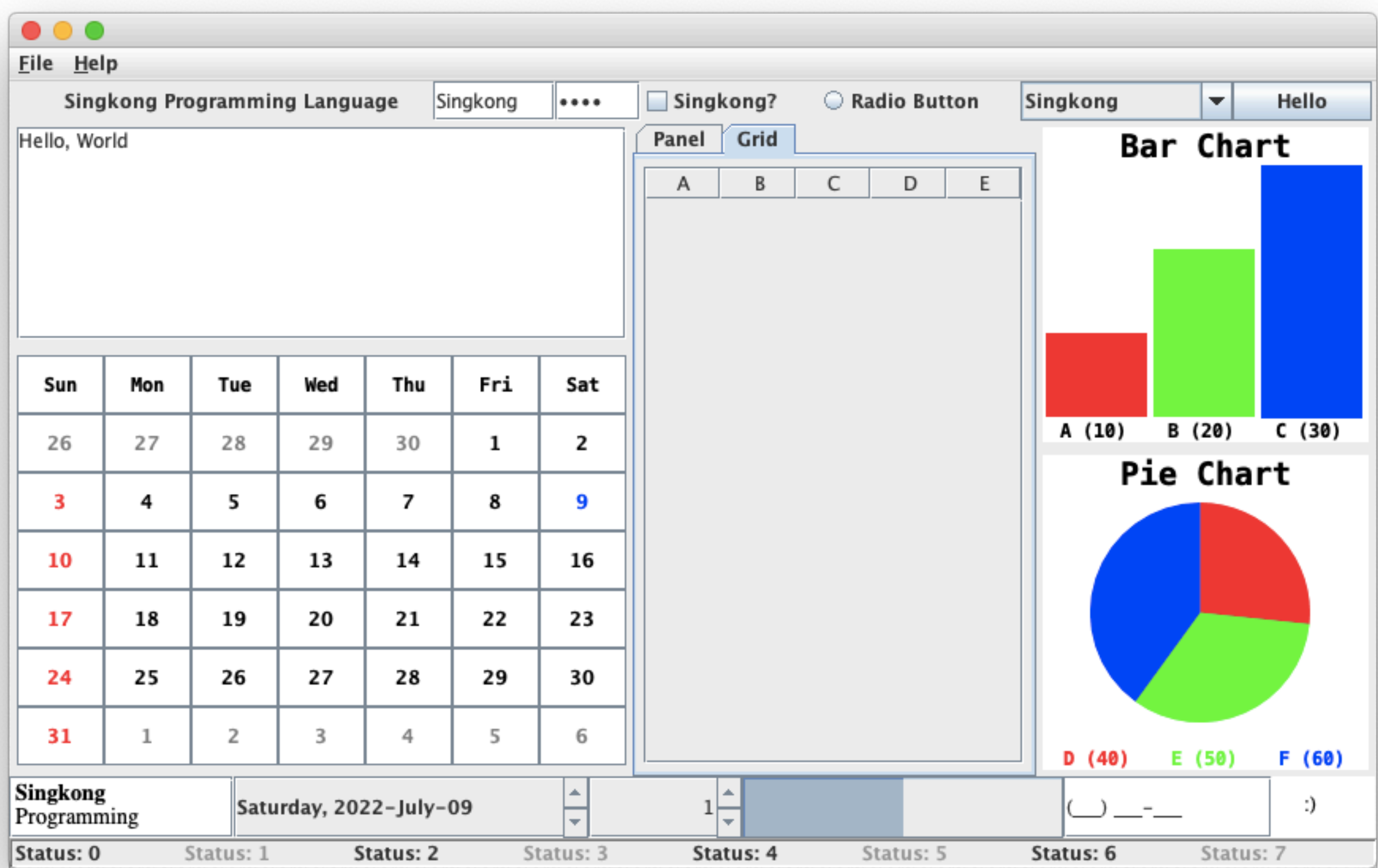
```
var a = fn() {
   println("a")
   var b = fn() {
      println("b")
      var c = fn() {
         println("c")
         var d = fn() {
            println("d")
         }
         d()
      }
      c()
   }
   b()
}
a()
```

Kenapa Singkong
Kebutuhan: GUI Harus Dapat Dibuat Semudah Mungkin (1)

- Menu bar
- Status bar
- Konfirmasi keluar aplikasi
- Bar Chart
- Button
- Checkbox
- Combobox
- Date
- Draw
- Edit
- Image
- Label
- Mask
- Password
- Pie Chart
- Progress
- Radio
- Tab
- Panel
- Grid
- Spin
- Table
- Text
- View
- Kalender

Semua dalam 73 baris

Kenapa Singkong
Kebutuhan: GUI Harus Dapat Dibuat Semudah Mungkin (2)

```
reset()
var b = component("button", "Hello")
var c = component("checkbox", "Singkong?")
var m = component("combobox", "Singkong,Programming,Language")
var d = component("date", "EEEE, yyyy-MMMM-dd")
var e = component("edit", "Hello, World")
var i = component("image", "image.jpg")
var l = component("label", "Singkong Programming Language")
var p = component("password", "test")
var sp = component("spin", "1,0,10,2")
var g = component("progress", "")
config(g, "contents", 50)
var r = component("radio", "Radio Button")
var a = component("tab", "")
var panel = component("panel", "Panel")
var t1 = component("table", "A,B,C,D,E")
var grid = component("grid", "Grid")
var t2 = component("table", "A,B,C,D,E")
var x = component("text", "Singkong")
var v = component("view", "<b>Singkong</b><br>Programming")
var s = component("mask", "(###) ###-###")
var dr = component("draw", "50, 50")

config(dr, "foreground", "black")
config(dr, "background", "white")
draw_string(dr, ":)", 20, 22)

panel_add(panel, t1, 10, 10, 250, 400)
tab_add(a, panel)
grid_add(grid, t2, 0, 0, 1, 1, 1, 1, 3, 0, 5, 5, 5, 5)
tab_add(a, grid)

var bc = component("barchart", "")
config(bc, "foreground", "black")
config(bc, "background", "white")
```

```
config(bc, "font", ["monospaced", 1, 20])
config(bc, "text", "Bar Chart")
config(bc, "contents", [[10, "A (10)", "red"], [20, "B (20)", "green"],
[30, "C (30)", "blue"]])

var pc = component("piechart", "")
config(pc, "foreground", "black")
config(pc, "background", "white")
config(pc, "font", ["monospaced", 1, 20])
config(pc, "text", "Pie Chart")
config(pc, "contents", [[40, "D (40)", "red"], [50, "E (50)", "green"],
[60, "F (60)", "blue"]])

var grid_chart = component("grid", "Grid")
grid_add(grid_chart, bc, 0, 0, 1, 1, 1, 1, 3, 0)
grid_add(grid_chart, pc, 0, 1, 1, 1, 1, 1, 3, 0)

var ge = component("grid", "")
load_module("ui_calendar")
var dd = part(@)
var ca = create_calendar_simple_compact(dd[0], dd[1])
grid_add(ge, e, 0, 0, 1, 1, 1, 1, 3, 0)
grid_add(ge, ca, 0, 1, 1, 1, 1, 1, 3, 0)

add([ge, a, grid_chart])
add_n([i, l, x, p, c, r, m, b])
add_s([v, d, sp, g, s, dr])

each(range(0,8), fn(e, i) {
    statusbar(e, "Status: " + e, i%2 == 0)
})

menubar([
    ["File", 0, [ ["Quit", 0, true, fn() {frame_close()}] ]],
    ["Help", 0, [ ["About", 0, true, fn() {message("Singkong")}] ]]
])

closing("Are you sure you want to quit this application?",
  "Please confirm")
show()
```

Kenapa Singkong
Kebutuhan: GUI Harus Dapat Dibuat Semudah Mungkin (3)

```
Test
```

```
reset()
var e = component("edit", "")
var o = component("button", "open")
var s = component("button", "save")
var l = component("label", "")

var oo = fn() {
    var f = open()
    if (!empty(f)) {
        config(e, "contents", read(f))
        config(l, "text", f)
    }
}
event(o, oo)

var ss = fn() {
    var f = save()
    if (!empty(f)) {
        var t = get(e, "contents")
        write(f, t)
        config(l, "text", f)
    }
}
event(s, ss)

add_n(l)
add(e)
add_s([o, s])
show()
```

- Editor teks sederhana
- Buka/Simpan file
  Semua dalam < 30 baris

Kenapa Singkong
Kebutuhan: GUI Harus Dapat Dibuat Semudah Mungkin (4)

- Program paint sederhana
- Buka/Simpan file gambar
- Menggambar sederhana dengan mouse
- Dapat menambahkan teks
- Mengubah nilai RGB pixel
  Semua dalam < 200 baris

Kenapa Singkong
Kebutuhan: Dukungan Database Relasional (1)

- Koneksi dan query (dengan transaksi) semudah mungkin, dalam masing-masing 1 baris kode

- Database relasional tanpa menggunakan SQL secara langsung

- Bundel JDBC Driver:

  - Apache Derby: Network Server, Driver (Embedded, Client)

  - PostgreSQL

- Dapat menggunakan berbagai JDBC Driver lain

Kenapa Singkong
Kebutuhan: Dukungan Database Relasional (2)

A	B
28	Hello World
11	Hello World
65	Hello World
37	Hello World
62	Hello World
64	Hello World
88	Hello World
45	Hello World

• GUI
• Koneksi database
• Query: pembuatan tabel
• Query: insert
• Query: update
• Query: select
**Semua dalam < 25 baris
(atau hanya 15 baris
dengan modul db_util)**

```
reset()
var t = component("table", "A,B", true)
add(t)

var d = database("org.apache.derby.jdbc.EmbeddedDriver", "jdbc:derby:test;create=true", "", "")
if (d != null) {
    var q = [ ["create table test(a integer, b varchar(64))", []] ]
    var r = query(d, q)

    var q = [
        ["insert into test(a,b) values(?, ?)", [random(0,100), "hello"]],
        ["update test set b=? where b=?", ["Hello World", "hello"]]
    ]
    var r = query(d, q)

    var q = [ ["select a,b from test", []] ]
    var r = query(d, q)
    if (!empty(r)) {
        config(t, "contents", r[0])
    }
}

show()
```

```
load_module("db_util")

reset()
var t = component("table", "A,B", true)
add(t)

var d = db_connect_embed("test")
if (d != null) {
    db_create_table_embed(d, "test", [["a", "integer."], ["b", "varchar."]])

    db_insert(d, "test", {"a": random(0,100), "b": "hello"})
    db_update(d, "test", [["b = ", "hello", ""]], {"b": "Hello World"})

    var r = db_select_all(d, "test")
    if (!empty(r)) {
        config(t, "contents", r[0])
    }
}

show()
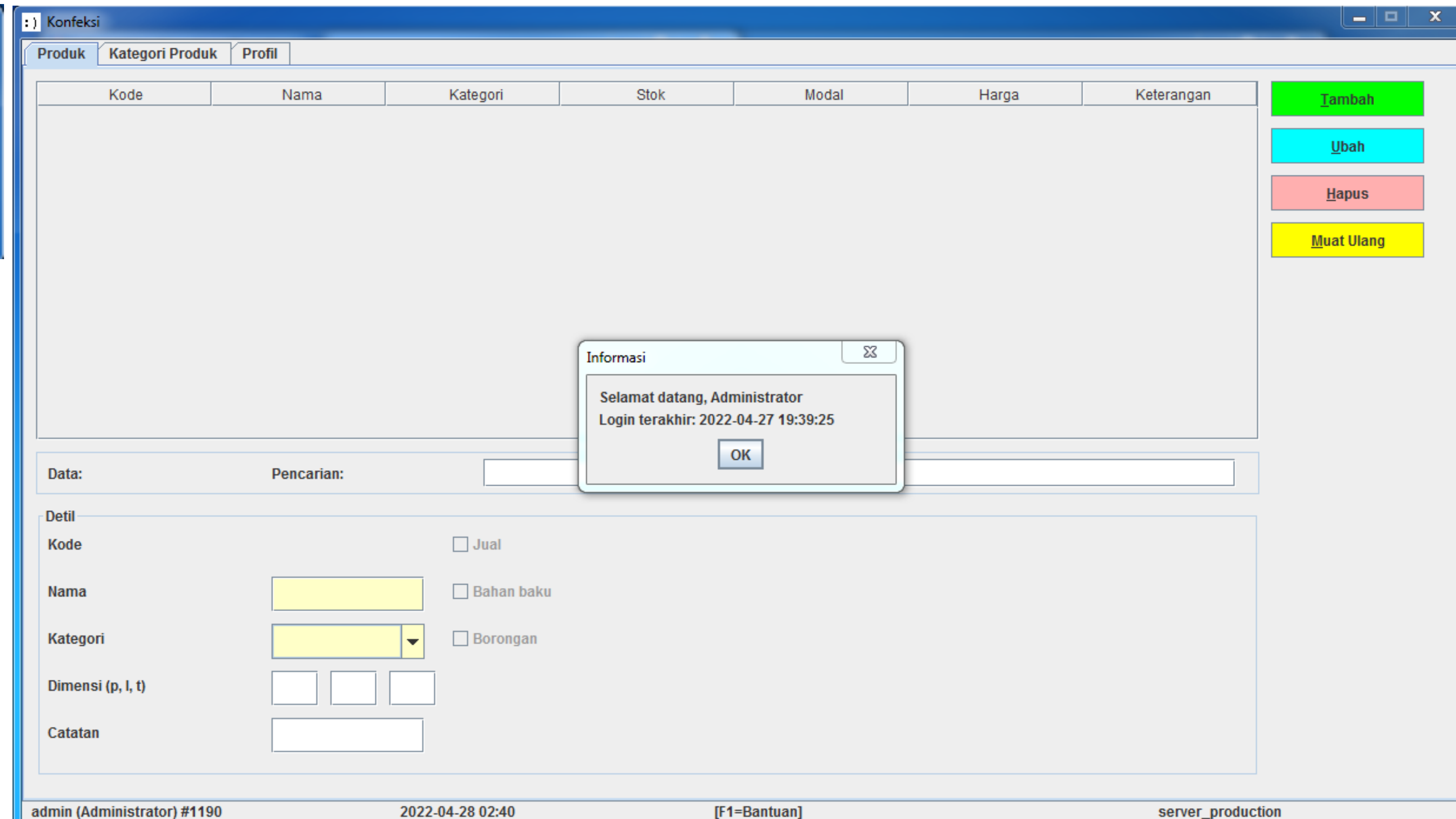```

# Kenapa Singkong
## Kebutuhan: HTTP dan Multithreading



- Frontend modul aplikasi konfeksi
- Bekerja dengan HTTP API
- Backend ditulis dengan Singkong
- Komunikasi dengan backend oleh thread tersendiri

# Kenapa Singkong
## Kebutuhan: Tipe Data Praktis

| Tipe Data | Deskripsi | Catatan |
|---|---|---|
| NUMBER | Bilangan bulat dan desimal | Batas maksimum tidak ditentukan. Scale 1-16 (default 4). Dapat diterapkan langsung pada aplikasi keuangan dan saintifik. Operator: + - * / == != %(remainder) ^(power) < <= > >= |
| BOOLEAN | true atau false | |
| STRING | Data string atau teks | Panjang tidak dibatasi. Diapit dengan kutip ganda. Operator: +(concatenation) -(remove) ==(equals,case-sensitive) != *(repeat) |
| ARRAY | Array (heterogen, Campur berbagai tipe), array dalam array. | Panjang tidak dibatasi. Termasuk rectangular array. Operator: +(add), -(remove) == != |
| HASH | Hash table / dictionary | Jumlah pemetaan tidak dibatasi. Memperhatikan insertion-order. Operator: +(add dictionary), -(remove) == != |
| DATE | Tanggal dan Waktu | @ @Y @YY @YYY @YYYY @YYYYM @YYYYMM @YYYYMMD @YYYYMMDD @YYYYMMDDh @YYYYMMDDhh @YYYYMMDDhhm @YYYYMMDDhhmm @YYYYMMDDhhmms @YYYYMMDDhhmmss |
| FUNCTION | Fungsi | First class. Mendukung documentation string. Mendukung rekursif. Mendukung fungsi dalam fungsi. |
| BUILTIN | Fungsi bawaan | Menyediakan berbagai fungsionalitas |
| COMPONENT | Komponen GUI | "barchart", "button", "checkbox", "combobox", "date", "draw", "edit", "grid", "image", "label", "mask", "panel", "password", "piechart", "progress", "radio", "spin", "tab", "table", "text", "view" |
| DATABASE | Koneksi Database | |
| UDP | User Datagram Protocol | Dapat digunakan untuk membuat server ataupun client dalam User Datagram Protocol |
| NULL | null | |

# Kenapa Singkong
## Kebutuhan: Fungsi dan Modul Bawaan

- Menyediakan berbagai fungsionalitas siap pakai

- Modul bawaan ditulis dengan Singkong

- Fungsi dan modul bawaan akan ditambahkan secara berkala

| Singkong v11.3 | 372 | 10 |
|:---:|:---:|:---:|
| | Fungsi bawaan | Modul bawaan |

# Kenapa Singkong
## Dapat Memanggil Method Java, Dapat Di-Embed ke Aplikasi Java (1)

Singkong dapat memanggil method yang ditulis dengan Bahasa Pemrograman Java dan mendapatkan nilai kembalian dari pemanggilan method tersebut.

Dengan demikian, fungsionalitas yang tidak disediakan oleh fungsi built-in dan tidak dapat dibuat dengan kode Singkong saja, dapat ditulis dalam Java.

Noprianto. (2021). *Mengenal dan Menggunakan Bahasa Pemrograman Singkong.* Jakarta: PT. Stabil Standar Sinergi

# Kenapa Singkong
## Dapat Memanggil Method Java, Dapat Di-Embed ke Aplikasi Java (2)

Apabila diinginkan, programmer Java bisa menambahkan Singkong.jar ke class path dan menggunakan interpreter Singkong untuk menginterpretasikan kode program Singkong, yang mungkin didapatkan dari input user. Singkong dapat berfungsi sebagai scripting engine sederhana dalam hal ini.

Noprianto. (2021). *Mengenal dan Menggunakan Bahasa Pemrograman Singkong.* Jakarta: PT. Stabil Standar Sinergi

# Kenapa Singkong
## Singkong.jar (4,4 MB): Evaluator, Editor, Database Tool, Dokumentasi (1)

# Kenapa Singkong
## Singkong.jar (4,4 MB): Evaluator, Editor, Database Tool, Dokumentasi (2)

# Kenapa Singkong
## Singkong.jar (4,4 MB): Evaluator, Editor, Database Tool, Dokumentasi (3)

# Kenapa Singkong
## Singkong.jar (4,4 MB): Evaluator, Editor, Database Tool, Dokumentasi (4)

- Bundel bersama JDBC Driver:

  - Apache Derby: Network Server, Driver (Embedded, Client)

  - PostgreSQL

- Modul Singkong (ditulis dengan Singkong)

# Kenapa Singkong
## Buku Singkong: singkong.pdf

- Buku tersedia gratis, ditulis dalam Bahasa Indonesia

- Mencakup semua yang dibutuhkan untuk mempelajari Singkong, termasuk berbagai contoh kode

- Selalu diperbaharui sesuai dengan versi terbaru Singkong (dirilis pada waktu yang sama)

- Buku dalam format siap cetak

# Kenapa Singkong
## Distribusi Aplikasi Anda Dalam File Jar Tunggal

- Aplikasi yang Anda kembangkan, bersama semua file pendukung (termasuk modul, gambar, suara, class Java), dapat dibundel bersama interpreter Singkong.

  - Menjadi file jar tunggal

  - Selama nama file jar aplikasi Anda tidak mengandung kata "Singkong"

- File jar tunggal tersebut dapat dijalankan di semua sistem operasi yang telah terinstall Java 5.0 atau lebih baru

  - Secara teknis, Anda dapat pula membundel Java runtime bersama file jar tunggal aplikasi Anda

# Passion dan Kolaborasi
## Bahasa Pemrograman Monkey

- Singkong berbasiskan pada Monkey.java (sekitar 3.000 baris kode Java)

    - Monkey.java berbasiskan pada monkey.py (sekitar 2.000 baris kode Python)

    - monkey.py berbasiskan pada kode dalam Bahasa Go, dalam buku: WRITING AN INTERPRETER IN GO

- Tersedia pula implementasi Bahasa Monkey dengan Bahasa Singkong: monkey.singkong (sekitar 2.100 baris kode Singkong)

- monkey.singkong, Monkey.java, dan monkey.py adalah free/open source dan dapat didownload dari situs web Singkong (https://nopri.github.io)

- Saat ini, source code Singkong (v11.3) berukuran lebih dari 13 kali Monkey.java, dalam sekitar 38.000 baris kode Java dan Singkong

# Passion dan Kolaborasi
## Sedikit Demi Sedikit

- Dimulai dari impian dan proyek hobi, Singkong kini telah menjadi passion, didukung secara komersial, dan memungkinkan:

  - Singkong digunakan di production (backend ataupun frontend)

    - Bug diperbaiki

    - Fungsionalitas baru/tambahan disediakan

  - Beberapa aplikasi berjalan telah/sedang ditulis ulang dengan Singkong

  - Rilis dilakukan berkala

  - Porting beberapa pustaka ke Singkong sedang/akan dilakukan

# Passion dan Kolaborasi
## Riset dan Pengembangan Bersama

| Kategori | Pekerjaan | Deskripsi |
|---|---|---|
| Editor | Editor, plugin untuk editor | Editor yang datang bersama Singkong.jar masih sederhana. Selama masih diperlukan, editor ataupun plugin-plugin untuk berbagai editor akan dikerjakan secara kolaboratif. |
| Buku | Berbagai buku ilmu komputer | Berbagai topik dengan implementasi dalam Bahasa Singkong. Saat ini, terdapat beberapa buku cetak yang telah beredar, dan beberapa lagi sedang ditulis atau dipersiapkan. |
| Pustaka | Pembuatan atau porting pustaka | Pembuatan pustaka untuk Singkong ataupun porting pustaka yang ditulis dengan bahasa pemrograman lain ke Singkong |
| Aplikasi | Pengembangan aplikasi dengan bahasa Singkong | Contoh: Bakwan, sebuah aplikasi open source akunting sederhana, sedang dikembangkan secara kolaboratif. |

# Terima Kasih

**atas perhatian dan partisipasi Anda :)**

# Contoh: GUI

## GUI: 4 baris

```
reset()
title("Matrix")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI

GUI: 10 baris

```
load_module("ui_util")

reset()
var t = component("table", "A,B,C,D,E,F,G,H,I,J")
table_add_fill(t, "")
add(t)

title("Matrix")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI
GUI: 11 baris

```
load_module("ui_util")

reset()
var t = component("table", "A,B,C,D,E,F,G,H,I,J")
var r = component("table", "Property, Value")
table_add_fill(t, "")
add([t, r])

title("Matrix")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI

GUI: 30 baris

```
load_module("ui_util")
load_module("rect_array_util")

var functions = [["Square", is_square_rect_array_of_number],
    ["Diagonal", is_diagonal_rect_array_of_number],
    ["Identity", is_identity_rect_array_of_number],
    ["Upper Triangular", is_upper_triangular_rect_array_of_number],
    ["Lower Triangular", is_lower_triangular_rect_array_of_number],
    ["Zero", is_zero_rect_array_of_number],
    ["Symmetric", is_symmetric_rect_array_of_number],
    ["Add (itself)", add_rect_array_of_number, null],
    ["Subtract (itself)", sub_rect_array_of_number, null],
    ["Multiply (itself)", mul_rect_array_of_number, null],
    ["Trace", trace_rect_array_of_number],
    ["Transpose", transpose_rect_array_of_number],
    ["Determinant", determinant_rect_array_of_number],
    ["Inverse", inverse_rect_array_of_number]]

reset()
var t = component("table", "A,B,C,D,E,F,G,H,I,J")
var r = component("table", "Property, Value")
each(functions, fn(e, i) {
    table_add(r, [[e[0]]])
})
table_add_fill(t, "")
add([t, r])

title("Matrix")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI

GUI: 35 baris

```
load_module("ui_util")
load_module("rect_array_util")

var functions = [["Square", is_square_rect_array_of_number],
    ["Diagonal", is_diagonal_rect_array_of_number],
    ["Identity", is_identity_rect_array_of_number],
    ["Upper Triangular", is_upper_triangular_rect_array_of_number],
    ["Lower Triangular", is_lower_triangular_rect_array_of_number],
    ["Zero", is_zero_rect_array_of_number],
    ["Symmetric", is_symmetric_rect_array_of_number],
    ["Add (itself)", add_rect_array_of_number, null],
    ["Subtract (itself)", sub_rect_array_of_number, null],
    ["Multiply (itself)", mul_rect_array_of_number, null],
    ["Trace", trace_rect_array_of_number],
    ["Transpose", transpose_rect_array_of_number],
    ["Determinant", determinant_rect_array_of_number],
    ["Inverse", inverse_rect_array_of_number]]

reset()
var t = component("table", "A,B,C,D,E,F,G,H,I,J")
var r = component("table", "Property, Value")
each(functions, fn(e, i) {
    table_add(r, [[e[0]]])
})
table_add_fill(t, "")
add([t, r])

var b = component("button", "Run")
event(b, fn() {
})
add_s(b)

title("Matrix")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI

GUI: 36 baris

```
load_module("ui_util")
load_module("rect_array_util")

var functions = [["Square", is_square_rect_array_of_number],
    ["Diagonal", is_diagonal_rect_array_of_number],
    ["Identity", is_identity_rect_array_of_number],
    ["Upper Triangular", is_upper_triangular_rect_array_of_number],
    ["Lower Triangular", is_lower_triangular_rect_array_of_number],
    ["Zero", is_zero_rect_array_of_number],
    ["Symmetric", is_symmetric_rect_array_of_number],
    ["Add (itself)", add_rect_array_of_number, null],
    ["Subtract (itself)", sub_rect_array_of_number, null],
    ["Multiply (itself)", mul_rect_array_of_number, null],
    ["Trace", trace_rect_array_of_number],
    ["Transpose", transpose_rect_array_of_number],
    ["Determinant", determinant_rect_array_of_number],
    ["Inverse", inverse_rect_array_of_number]]

reset()
var t = component("table", "A,B,C,D,E,F,G,H,I,J")
var r = component("table", "Property, Value")
each(functions, fn(e, i) {
    table_add(r, [[e[0]]])
})
table_add_fill(t, "")
add([t, r])

var b = component("button", "Run")
event(b, fn() {
    var m = table_get_array_number(t)
})
add_s(b)

title("Matrix")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI

GUI: 44 baris

```
load_module("ui_util")
load_module("rect_array_util")

var functions = [["Square", is_square_rect_array_of_number],
    ["Diagonal", is_diagonal_rect_array_of_number],
    ["Identity", is_identity_rect_array_of_number],
    ["Upper Triangular", is_upper_triangular_rect_array_of_number],
    ["Lower Triangular", is_lower_triangular_rect_array_of_number],
    ["Zero", is_zero_rect_array_of_number],
    ["Symmetric", is_symmetric_rect_array_of_number],
    ["Add (itself)", add_rect_array_of_number, null],
    ["Subtract (itself)", sub_rect_array_of_number, null],
    ["Multiply (itself)", mul_rect_array_of_number, null],
    ["Trace", trace_rect_array_of_number],
    ["Transpose", transpose_rect_array_of_number],
    ["Determinant", determinant_rect_array_of_number],
    ["Inverse", inverse_rect_array_of_number]]

reset()
var t = component("table", "A,B,C,D,E,F,G,H,I,J")
var r = component("table", "Property, Value")
each(functions, fn(e, i) {
    table_add(r, [[e[0]]])
})
table_add_fill(t, "")
add([t, r])

var b = component("button", "Run")
event(b, fn() {
    var m = table_get_array_number(t)
    config(r, "contents", [])
    each(functions, fn(e, i) {
        if (len(e) == 3) {
            table_add(r, [[e[0], e[1](m, m)]])
        } else {
            table_add(r, [[e[0], e[1](m)]])
        }
    })
})
add_s(b)

title("Matrix")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI

GUI: Matriks

Sekali ditulis, program yang ditulis dengan Singkong dapat jalan di sebanyak mungkin sistem operasi (terbaru ataupun yang dirilis sekitar 25 tahun lalu)

My Computer

:) Matrix

A	B	C	D	E	F	G	H	I	J		Property	Value
0	-3	-2									Square	true
1	-4	-2									Diagonal	false
-3	4	1									Identity	false
											Upper Triangular	false
											Lower Triangular	false
											Zero	false
											Symmetric	false
											Add (itself)	[[0, -6, -4], [2, -8, -4], [-6, 8, 2]]
											Subtract (itself)	[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
											Multiply (itself)	[[3, 4, 4], [2, 5, 4], [1, -3, -1]]
											Trace	-3
											Transpose	[[0, 1, -3], [-3, -4, 4], [-2, -2, 1]]
											Determinant	1
											Inverse	[[4, -5, -2], [5, -6, -2], [-8, 9, 3]]

Run

Start | JAVA | :) Matrix | 8:48 AM

Contoh: GUI dan Database

GUI dan Database: 4 baris

```
reset()
title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI dan Database

GUI dan Database: 10 baris

```
reset()
var t = component("table", "ID, NAME, PRICE")
table_right(t, 0)
table_right(t, 2)

add(t)

title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI dan Database

GUI dan Database: 18 baris

```
load_module("db_util")

var d = db_connect_embed("database_test")
if (d == null) {
    message("Cannot connect to database", "Error")
    exit()
}

reset()
var t = component("table", "ID, NAME, PRICE")
table_right(t, 0)
table_right(t, 2)

add(t)

title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI dan Database

GUI dan Database: 19 baris

```
load_module("db_util")

var d = db_connect_embed("database_test")
if (d == null) {
    message("Cannot connect to database", "Error")
    exit()
}
db_create_table_embed(d, "products", [["id", "id"], ["name", "varchar."], ["price", "decimal."]])

reset()
var t = component("table", "ID, NAME, PRICE")
table_right(t, 0)
table_right(t, 2)

add(t)

title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI dan Database

GUI dan Database: 32 baris

```
load_module("db_util")

var d = db_connect_embed("database_test")
if (d == null) {
    message("Cannot connect to database", "Error")
    exit()
}
db_create_table_embed(d, "products", [["id", "id"], ["name", "varchar."], ["price", "decimal."]])

var reload = fn() {
    config(t, "contents", [])
    var r = query_result(db_select_all(d, "products"))
    var rr = []
    if (r != null) {
        each(r, fn(e, i) {
            var rr = rr + [e[0], e[1], number_group(e[2], ",", ".")]
        })
    }
    config(t, "contents", rr)
}

reset()
var t = component("table", "ID, NAME, PRICE")
table_right(t, 0)
table_right(t, 2)

add(t)

reload()
title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI dan Database

GUI dan Database: 38 baris

```
load_module("db_util")

var d = db_connect_embed("database_test")
if (d == null) {
    message("Cannot connect to database", "Error")
    exit()
}
db_create_table_embed(d, "products", [["id", "id"], ["name", "varchar."], ["price", "decimal."]])

var reload = fn() {
    config(t, "contents", [])
    var r = query_result(db_select_all(d, "products"))
    var rr = []
    if (r != null) {
        each(r, fn(e, i) {
            var rr = rr + [e[0], e[1], number_group(e[2], ",", ".")]
        })
    }
    config(t, "contents", rr)
}

reset()
var g = component("grid", "")
var t = component("table", "ID, NAME, PRICE")
table_right(t, 0)
table_right(t, 2)
var b_new = component("button", "New")
var b_del = component("button", "Delete")
grid_add(g, t, 0, 0, 3, 1, 1.0, 1.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_new, 0, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_del, 1, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)

add(g)

reload()
title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI dan Database
GUI dan Database: 52 baris

```
load_module("db_util")

var d = db_connect_embed("database_test")
if (d == null) {
    message("Cannot connect to database", "Error")
    exit()
}
db_create_table_embed(d, "products", [["id", "id"], ["name", "varchar."], ["price", "decimal."]])

var reload = fn() {
    config(t, "contents", [])
    var r = query_result(db_select_all(d, "products"))
    var rr = []
    if (r != null) {
        each(r, fn(e, i) {
            var rr = rr + [e[0], e[1], number_group(e[2], ",", ".")]
        })
    }
    config(t, "contents", rr)
}

reset()
var g = component("grid", "")
var t = component("table", "ID, NAME, PRICE")
table_right(t, 0)
table_right(t, 2)
var b_new = component("button", "New")
var b_del = component("button", "Delete")
grid_add(g, t, 0, 0, 3, 1, 1.0, 1.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_new, 0, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_del, 1, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)
```

```
var g_detail = component("grid", "")
config(g_detail, "border", "Detail")
var l_name = component("label", "Name")
var t_name = component("text", "")
var l_price = component("label", "Price")
var t_price = component("text", "")
var b_save = component("button", "Save")
grid_add(g_detail, l_name, 0, 0, 1, 1, 0.0, 0.5, 0, 1, 5, 5, 5, 5)
grid_add(g_detail, t_name, 1, 0, 1, 1, 1.0, 0.5, 1, 1, 5, 5, 5, 5)
grid_add(g_detail, l_price, 2, 0, 1, 1, 0.0, 0.5, 0, 1, 5, 5, 5, 5)
grid_add(g_detail, t_price, 3, 0, 1, 1, 0.1, 0.5, 1, 1, 5, 5, 5, 5)
grid_add(g_detail, b_save, 4, 0, 1, 1, 0.0, 0.5, 1, 1, 5, 5, 5, 5)

grid_add(g, g_detail, 0, 2, 3, 1, 1.0, 0.0, 3, 0, 5, 5, 5, 5)
add(g)

reload()
title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI dan Database

GUI dan Database: 58 baris

```
load_module("db_util")

var d = db_connect_embed("database_test")
if (d == null) {
    message("Cannot connect to database", "Error")
    exit()
}
db_create_table_embed(d, "products", [["id", "id"], ["name", "varchar."], ["price", "decimal."]])

var reload = fn() {
    config(t, "contents", [])
    var r = query_result(db_select_all(d, "products"))
    var rr = []
    if (r != null) {
        each(r, fn(e, i) {
            var rr = rr + [e[0], e[1], number_group(e[2], ",", ".")]
        })
    }
    config(t, "contents", rr)
}

reset()
var g = component("grid", "")
var t = component("table", "ID, NAME, PRICE")
table_right(t, 0)
table_right(t, 2)
var b_new = component("button", "New")
var b_del = component("button", "Delete")
grid_add(g, t, 0, 0, 3, 1, 1.0, 1.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_new, 0, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_del, 1, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)
```

```
var g_detail = component("grid", "")
config(g_detail, "border", "Detail")
var l_name = component("label", "Name")
var t_name = component("text", "")
var l_price = component("label", "Price")
var t_price = component("text", "")
var b_save = component("button", "Save")
grid_add(g_detail, l_name, 0, 0, 1, 1, 0.0, 0.5, 0, 1, 5, 5, 5, 5)
grid_add(g_detail, t_name, 1, 0, 1, 1, 1.0, 0.5, 1, 1, 5, 5, 5, 5)
grid_add(g_detail, l_price, 2, 0, 1, 1, 0.0, 0.5, 0, 1, 5, 5, 5, 5)
grid_add(g_detail, t_price, 3, 0, 1, 1, 0.1, 0.5, 1, 1, 5, 5, 5, 5)
grid_add(g_detail, b_save, 4, 0, 1, 1, 0.0, 0.5, 1, 1, 5, 5, 5, 5)

grid_add(g, g_detail, 0, 2, 3, 1, 1.0, 0.0, 3, 0, 5, 5, 5, 5)
add(g)

event(b_new, fn() {
    config(t_name, "contents", "")
    config(t_price, "contents", "")
    config(t_name, "focus", true)
})

reload()
title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI dan Database
GUI dan Database: 76 baris

```
load_module("db_util")

var d = db_connect_embed("database_test")
if (d == null) {
    message("Cannot connect to database", "Error")
    exit()
}
db_create_table_embed(d, "products", [["id", "id"], ["name", "varchar."], ["price", "decimal."]])

var reload = fn() {
    config(t, "contents", [])
    var r = query_result(db_select_all(d, "products"))
    var rr = []
    if (r != null) {
        each(r, fn(e, i) {
            var rr = rr + [e[0], e[1], number_group(e[2], ",", ".")]
        })
    }
    config(t, "contents", rr)
}

reset()
var g = component("grid", "")
var t = component("table", "ID, NAME, PRICE")
table_right(t, 0)
table_right(t, 2)
var b_new = component("button", "New")
var b_del = component("button", "Delete")
grid_add(g, t, 0, 0, 3, 1, 1.0, 1.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_new, 0, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_del, 1, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)
```

```
var g_detail = component("grid", "")
config(g_detail, "border", "Detail")
var l_name = component("label", "Name")
var t_name = component("text", "")
var l_price = component("label", "Price")
var t_price = component("text", "")
var b_save = component("button", "Save")
grid_add(g_detail, l_name, 0, 0, 1, 1, 0.0, 0.5, 0, 1, 5, 5, 5, 5)
grid_add(g_detail, t_name, 1, 0, 1, 1, 1.0, 0.5, 1, 1, 5, 5, 5, 5)
grid_add(g_detail, l_price, 2, 0, 1, 1, 0.0, 0.5, 0, 1, 5, 5, 5, 5)
grid_add(g_detail, t_price, 3, 0, 1, 1, 0.1, 0.5, 1, 1, 5, 5, 5, 5)
grid_add(g_detail, b_save, 4, 0, 1, 1, 0.0, 0.5, 1, 1, 5, 5, 5, 5)

grid_add(g, g_detail, 0, 2, 3, 1, 1.0, 0.0, 3, 0, 5, 5, 5, 5)
add(g)

event(b_new, fn() {
    config(t_name, "contents", "")
    config(t_price, "contents", "")
    config(t_name, "focus", true)
})

event(b_save, fn() {
    var n = trim(get(t_name, "contents"))
    var p = get(t_price, "contents")
    var pp = number(p)
    var fp = number_group(pp, ",", ".")
    if (empty(n)) {
        message("Please fill in product name", "Error")
    } else {
        var w = fp + newline() + words_en(p) + newline() + words_id(p)
        var x = n + newline() + w
        var conf = confirm("Are you sure you want to add: " + x , "Please Confirm")
        if (conf == "OK") {
            var res = db_insert(d, "products", {"name": n, "price": pp})
            reload()
        }
    }
})

reload()
title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

51

Contoh: GUI dan Database

GUI dan Database: 90 baris

```
load_module("db_util")

var d = db_connect_embed("database_test")
if (d == null) {
    message("Cannot connect to database", "Error")
    exit()
}
db_create_table_embed(d, "products", [["id", "id"], ["name", "varchar."], ["price", "decimal."]])

var reload = fn() {
    config(t, "contents", [])
    var r = query_result(db_select_all(d, "products"))
    var rr = []
    if (r != null) {
        each(r, fn(e, i) {
            var rr = rr + [e[0], e[1], number_group(e[2], ",", ".")]
        })
    }
    config(t, "contents", rr)
}

reset()
var g = component("grid", "")
var t = component("table", "ID, NAME, PRICE")
table_right(t, 0)
table_right(t, 2)
var b_new = component("button", "New")
var b_del = component("button", "Delete")
grid_add(g, t, 0, 0, 3, 1, 1.0, 1.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_new, 0, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b_del, 1, 1, 1, 1, 0.5, 0.0, 3, 0, 5, 5, 5, 5)

var g_detail = component("grid", "")
config(g_detail, "border", "Detail")
var l_name = component("label", "Name")
var t_name = component("text", "")
var l_price = component("label", "Price")
var t_price = component("text", "")
var b_save = component("button", "Save")
grid_add(g_detail, l_name, 0, 0, 1, 1, 0.0, 0.5, 0, 1, 5, 5, 5, 5)
grid_add(g_detail, t_name, 1, 0, 1, 1, 1.0, 0.5, 1, 1, 5, 5, 5, 5)
grid_add(g_detail, l_price, 2, 0, 1, 1, 0.0, 0.5, 0, 1, 5, 5, 5, 5)
grid_add(g_detail, t_price, 3, 0, 1, 1, 0.1, 0.5, 1, 1, 5, 5, 5, 5)
grid_add(g_detail, b_save, 4, 0, 1, 1, 0.0, 0.5, 1, 1, 5, 5, 5, 5)
```

```
grid_add(g, g_detail, 0, 2, 3, 1, 1.0, 0.0, 3, 0, 5, 5, 5, 5)
add(g)

event(b_new, fn() {
    config(t_name, "contents", "")
    config(t_price, "contents", "")
    config(t_name, "focus", true)
})

event(b_save, fn() {
    var n = trim(get(t_name, "contents"))
    var p = get(t_price, "contents")
    var pp = number(p)
    var fp = number_group(pp, ",", ".")
    if (empty(n)) {
        message("Please fill in product name", "Error")
    } else {
        var w = fp + newline() + words_en(p) + newline() + words_id(p)
        var x = n + newline() + w
        var conf = confirm("Are you sure you want to add: " + x , "Please Confirm")
        if (conf == "OK") {
            var res = db_insert(d, "products", {"name": n, "price": pp})
            reload()
        }
    }
})

event(b_del, fn() {
    var s = get(t, "active")
    if (s > -1) {
        var x = table_get_value(t, s, 0)
        var conf = confirm("Are you sure you want to delete: #" + x , "Please Confirm")
        if (conf == "OK") {
            db_delete(d, "products", {"id=": x})
            reload()
        }
    } else {
        message("Please select a product", "Error")
    }
})

reload()
title("Products")
closing("Are you sure you want to quit this application?", "Please confirm")
show()
```

Contoh: GUI dan Database

GUI dan Database: Select / Insert / Delete

Sekali ditulis, program yang ditulis dengan Singkong dapat jalan di sebanyak mungkin sistem operasi (terbaru ataupun yang dirilis sekitar 25 tahun lalu)