Singkong versi 7.1

Mengenal dan Menggunakan Bahasa Pemrograman **Singkong**

Dr. Noprianto

Penerbit:
PT. Stabil Standar Sinergi
Download gratis buku ini:
https://nopri.github.io

ISBN 978-602-52770-1-6

Mengenal dan Menggunakan Bahasa Pemrograman Singkong

Penulis: Dr. Noprianto

ISBN: 978-602-52770-1-6

Penerbit:
PT. Stabil Standar Sinergi
Website: https://singkong.dev
Email: info@singkong.dev

Tahun	Bulan	Menyesuaikan Versi Singkong
2020	Januari	
2020	April	3.1 3.2 3.3
2020	Mei	3.4 3.5 3.6 3.7 3.8
2020	Juni	3.9 4.0
2020	Desember	4.1 4.2 4.3 4.4
2021	Januari	4.5 4.6
2021	Februari	4.7 5.0
2021	Maret	5.1
2021	Mei	5.2 5.3
2021	Juni	5.4 5.5 5.6
2021	Juli	5.7 5.9 6.0 6.1 6.2
2021	Agustus	6.3 6.4 6.5 6.6
2022	April	6.7 6.8
2022	Mei	6.9 7.0 7.1

Tentang Penulis

Dr. Noprianto mengembangkan bahasa pemrograman Singkong dan interpreternya sejak akhir 2019. Beliau mendirikan/mengelola Singkong.dev (PT. Stabil Standar Sinergi), menyukai pemrograman, dan telah menulis beberapa buku cetak, termasuk Python, Java, dan Singkong. Buku dan softwarenya dapat didownload dari https://nopri.github.io

Download

Interpreter Singkong akan selalu didistribusikan sebagai sebuah file jar (Singkong.jar), yang kompatibel dengan Java 5.0 atau lebih baru.

Download: https://nopri.github.io/Singkong.jar

Singkong.jar berukuran **4,3** MB (versi 7.1) dan berisikan semua yang diperlukan untuk menjalankan interpreter Singkong, termasuk:

- Interactive evaluator, editor, database tool, dokumentasi
- Driver JDBC / database untuk Apache Derby (Embedded Driver dan Client Driver) dan PostgreSQL
- Network Server Apache Derby

Singkong dikembangkan dengan Java versi 8 dan dikompilasi dengan opsi -source 1.5 -target 1.5 (dan hanya menggunakan API yang kompatibel dengan Java 5.0).

Singkong.jar telah diuji berjalan pada berbagai sistem berikut. Mulai dari sistem operasi yang dirilis pada tahun 1998 sampai yang terbaru. Penulis berharap Singkong tersedia untuk siapa saja, selama dapat menjalankan Java Runtime Environment 5.0 atau lebih baru (Java 5.0 dirilis 2004, namun dapat dijalankan di berbagai sistem operasi yang dirilis sebelumnya).

macOS	11, 10.15, 10.14, Mac OS X 10.4
Windows	10, 7, XP, 2000, 98
Linux	Ubuntu (20.04, 18.04, 16.04, 4.10), Raspberry Pi OS, Red Hat Linux 7.3, Debian 10 di 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

Kata Pengantar

Bahasa pemrograman Singkong merupakan bahasa pemrograman yang case-insensitive (tidak membedakan huruf besar/kecil), dynamically typed (tipe data ditentukan secara dinamis pada saat program berjalan), prosedural, dan interpreted, yang berjalan pada Java Virtual Machine (Java 5.0 atau lebih baru).

Singkong mendukung tipe data number, boolean, string, array, hash, date, function (first-class function), component (GUI), dan database. Untuk memudahkan pemrograman, Singkong juga datang dengan 339 built-in function (fungsi bawaan) dan 6 built-in module (modul bawaan). Singkong dapat digunakan untuk mengembangkan berbagai jenis aplikasi yang dapat dilengkapi dengan Graphical User Interface dan terhubung ke berbagai sistem database relasional. Aplikasi yang dikembangkan tersebut dapat dijalankan pada berbagai sistem operasi dimana Java tersedia.

Singkong juga dapat di-embed ke dalam aplikasi lain (misal untuk kebutuhan scripting sederhana) dan dapat memanggil method Java (yang menyediakan fungsionalitas tambahan).

Dengan dirancang hanya membutuhkan Java 5.0 (yang dirilis pada tahun 2004, 15 tahun sebelum Singkong dikembangkan), namun dapat berjalan pada Java versi terbaru (pada saat buku ini ditulis), Singkong diharapkan dapat digunakan oleh siapa pun, dengan komputer apapun, termasuk untuk mempelajari pemrograman.

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

Terima kasih telah menggunakan Singkong dan/atau membaca buku ini.
Noprianto

Halaman ini sengaja dikosongkan

Daftar Isi

Halaman ini sengaja dikosongkan

1. Mengenal Singkong

Singkong adalah sebuah bahasa pemrograman. Sebagaimana bahasa pada umumnya, Singkong memiliki sejumlah aturan. Akan tetapi, karena Singkong merupakan bahasa pemrograman yang relatif sederhana, aturan yang perlu ditaati juga relatif sedikit, yang akan dibahas secara bertahap di dalam buku ini. Mari kita memulai perkenalan dengan beberapa karakteristik Singkong berikut.

Pertama: Singkong tidak membedakan huruf besar dan huruf kecil untuk nama variabel, nama fungsi, ataupun kata kunci. Bagi Singkong, variabel nama dan NAMA adalah variabel yang sama: kita bisa menuliskannya sebagai NAMA, Nama, nama, ataupun kombinasi huruf besar/kecil lainnya. Demikian juga nama fungsi bawaan seperti random, yang dapat dituliskan sebagai RANDOM, Random, random, ataupun kombinasi lain. Kata kunci TRUE dan true sama-sama menyatakan benar. If, if, dan IF juga merupakan kata kunci yang sama.

Tentu saja, ini tidak berlaku untuk nilai sebuah STRING. Apabila Anda meminta input kepada user, kemudian user memberikan nilai “Singkong”, dan nilai tersebut disimpan dalam sebuah variabel nama, maka variabel nama akan bernilai “Singkong”, bukan “SINGKONG” atau kombinasi huruf besar/kecil lainnya.

Kedua: Singkong tidak mengharuskan sebuah variabel dideklarasikan dengan tipe tertentu. Kita cukup memberikan nilai untuk sebuah variabel dan tipenya akan ditentukan pada saat program berjalan. Untuk memberikan nilai pada sebuah variabel, Singkong menggunakan kata kunci var seperti contoh berikut:

```
var nama = “Singkong"
```

Dalam hal ini, nama tidak perlu dideklarasikan terlebih dahulu dengan tipe tertentu. Ketika program dijalankan, nama bertipe STRING. Namun, apabila dibaris berikutnya kita memberikan statement berikut:

```
var nama = true
```

maka nama setelahnya akan bertipe BOOLEAN.

Ketiga: Singkong adalah bahasa pemrograman prosedural. Dalam hal ini, kode Singkong akan banyak menggunakan fungsi, baik yang telah disediakan ataupun dibuat sendiri oleh programmer. Kode program dijalankan dari atas ke bawah, sesuai yang dituliskan, mengikuti alur kontrol seperti seleksi/ kondisi dan perulangan.

Sejumlah statement dapat dikelompokkan dalam blok, yang diawali dengan { dan diakhiri dengan }. Blok digunakan dalam fungsi, seleksi/kondisi, dan perulangan.

Fungsi di Singkong dapat memanggil fungsi lain ataupun dirinya sendiri (rekursif), baik yang dideklarasikan dalam file program yang sama ataupun file program lainnya.

Keempat: Kode program Singkong akan dijalankan lewat interpreter Singkong. Dengan demikian, agar kode program Singkong yang Anda buat dapat dijalankan di komputer lain, komputer tersebut perlu terinstal Java Runtime Environment dan Singkong terlebih dahulu.

Akan tetapi, Anda dapat menggunakan instalasi runtime Java portable atau membundelnya bersama dengan interpreter Singkong. Tidak diperlukan instalasi secara system-wide di komputer tujuan (dapat diinstall di direktori manapun). Kita bahkan dapat sepenuhnya menjalankan kode program Singkong lewat media penyimpanan portable seperti USB Flash Disk.

Interpreter Singkong sendiri akan selalu didistribusikan sebagai sebuah file jar tunggal Singkong.jar, yang pada saat buku ini ditulis, kompatibel dengan Java 5.0 atau yang lebih baru.

File Singkong.jar berisi interpreter Singkong, editor sederhana, interactive evaluator, database tool, dan dokumentasi.

Interpreter Singkong dapat berjalan pada lingkungan kerja GUI ataupun berbasis teks. Pada sistem operasi atau desktop yang mendukung, klik ganda pada Singkong.jar akan menjalankan interactive evaluator GUI.

Interactive Evaluator

Tujuan dari disediakannya interactive evaluator adalah programmer dapat mengetikkan kode Singkong baris demi baris dan melihat hasilnya pada waktu itu juga. Sebagai pelengkap, sebuah tabel berisikan nama variabel, tipe, serta representasi STRING dari nilainya akan ditampilkan.

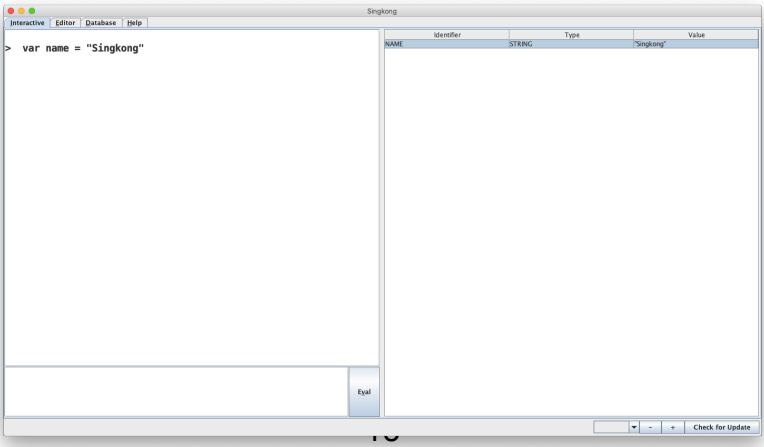
Interactive evaluator dapat berjalan pada lingkungan kerja GUI (seperti sistem operasi desktop pada umumnya) ataupun yang berbasis teks (sebagai contoh, pada sistem operasi server). Walaupun GUI tersedia, kita juga dapat meminta agar evaluator ini dijalankan hanya pada mode teks.

Pada interactive evaluator, nilai dari suatu ekspresi akan langsung ditampilkan, sehingga kita tidak perlu menampilkannya secara eksplisit. Sebagai contoh, ketika Anda mengetikkan 1+2 pada prompt, maka secara otomatis, 3 akan ditampilkan ketika Anda menekan tombol Enter atau klik pada tombol Eval.

```
> 1+2
3
```

Ketika evaluator dijalankan pada mode GUI, tabel berisikan nama, tipe, dan representasi STRING dari nilai akan ditampilkan. Dengan demikian, kita dapat mengetahui variabel apa saja yang telah ada. Secara otomatis, apabila kita membuat variabel baru atau mengubah nilai sebuah variabel, baris untuk nama variabel tersebut akan terpilih.

Interactive evaluator sangat berguna untuk menjalankan beberapa baris kode Singkong dengan cepat, tanpa harus menyimpannya ke dalam file terlebih dahulu. Anda juga dapat memanfaatkan evaluator ini sebagai kalkulator, bahkan untuk bilangan yang sangat besar.



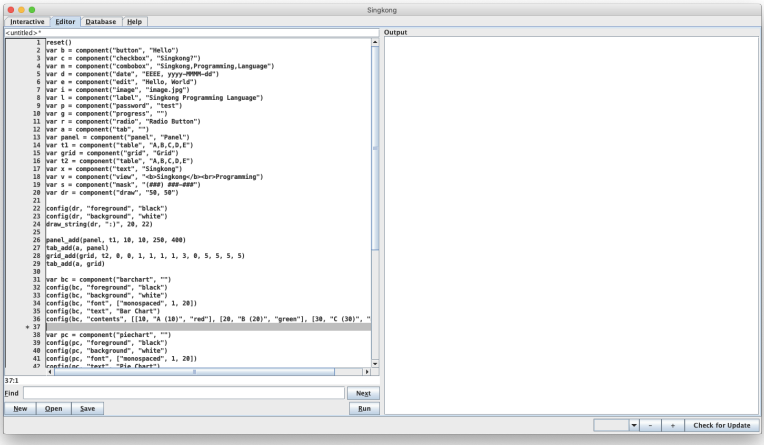
Untuk mengubah ukuran font atau tema editor, gunakanlah combo box serta tombol di sisi bawah frame.

Editor

Tujuan dari disediakannya editor adalah programmer dapat melakukan pengeditan sederhana untuk kode program Singkong. Kita dapat membuat file baru, membuka file yang telah ada, dan menyimpan perubahan pada file tersebut. Kita juga dapat langsung menjalankan kode program Singkong, baik tanpa disimpan ke file atau setelah menyimpannya.

Editor juga dapat digunakan untuk mencoba menjalankan beberapa baris kode Singkong sekaligus, yang mana tidak dapat dilakukan pada interactive evaluator.

Akan tetapi, berbeda halnya dengan interactive evaluator yang dapat berjalan pada mode GUI ataupun teks, editor hanya tersedia pada mode GUI.



Catatan: klik kanan / control click pada editor akan menampilkan menu popup.

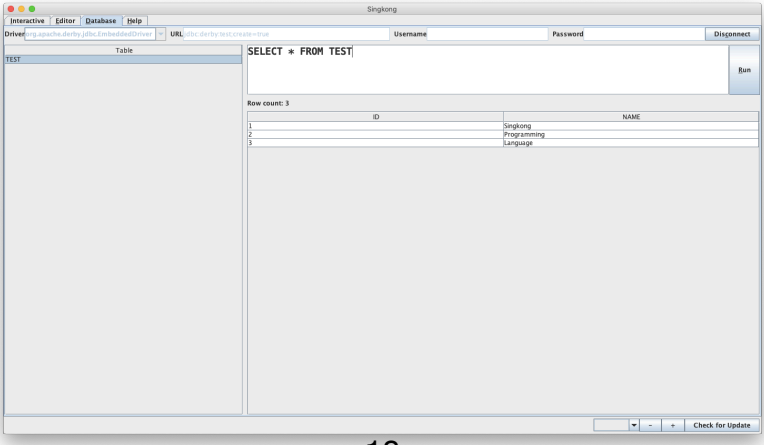
Untuk kepentingan penuliskan kode Singkong yang panjang dengan nyaman, Anda mungkin lebih memilih untuk mengunakan editor favorit Anda. Editor file teks yang datang bersama Singkong.jar sangatlah sederhana.

Database Tool

Untuk terhubung ke sistem database relasional, melihat daftar tabel yang ada dalam database, dan memberikan query (serta mendapatkan hasilnya), database tool yang disertakan dapat digunakan.

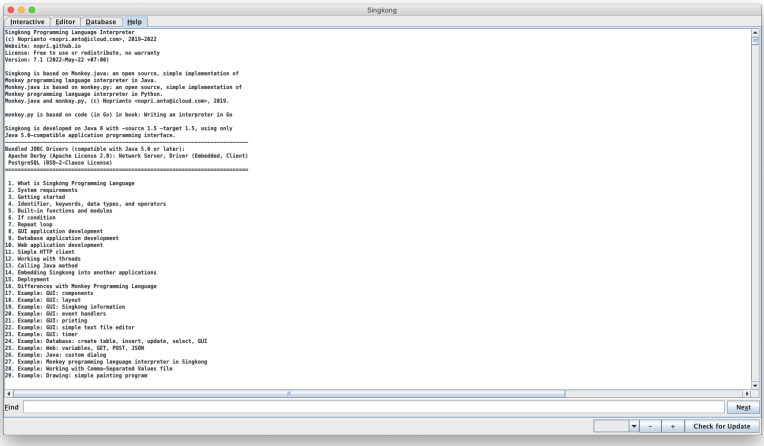
Fitur ini memungkinkan pengelolaan database sederhana (misal pengujian query, pembuatan tabel baru, pengelolaan data, dan lain sebagainya) sambil pengembangan program dilakukan, tanpa harus menggunakan alat bantu tambahan. Cukup dengan Singkong.jar saja.

Database tool dapat bekerja dengan perintah SQL apapun, selama valid untuk sistem database yang digunakan. Apabila terdapat kesalahan, maka pesan kesalahan akan ditampilkan. Apabila perintah mengembalikan result set, maka akan ditampilkan di tabel hasil. Untuk perintah yang mengembalikan update count, maka jumlah baris yang ter-update akan ditampilkan.



Membaca Dokumentasi

Di dalam file Singkong.jar, dokumentasi atau referensi bahasa pemrograman Singkong telah ikut disertakan. Apabila Singkong.jar dijalankan pada mode GUI, kita dapat mengakses dokumentasi ini.



Pada mode teks, kita tetap dapat mengakses dokumentasi setiap fungsi bawaan dengan mengetikkan nama fungsi saja. Fasilitas ini tersedia untuk mode GUI dan teks.

```
> random
built-in function: random: returns random
number (between 0 inclusive and 1 exclusive),
random number between min and max (both
inclusive), random element in ARRAY, random
key in HASH
arguments: 0, 1 (ARRAY or HASH), 2: (NUMBER
and NUMBER)
return value: <any type>
```

Menjalankan Interpreter Singkong

Sebelum memulai, download-lah terlebih dahulu Singkong.jar. Apabila diperlukan, bacalah juga petunjuknya di halaman 4.

Sebelum memulai, pastikanlah Java Runtime Environment telah terinstal. Anda mungkin ingin merujuk pada cara instalasi Java Runtime Environment di sistem operasi yang Anda gunakan, apabila belum terinstal.

Interpreter Singkong dapat dijalankan dengan tugas-tugas berikut:

1. Standalone tanpa command-line argument: dalam hal ini, kita sepenuhnya ingin menggunakan interactive evaluator, editor, database tool, ataupun membaca dokumentasi. Apabila dijalankan pada mode teks, hanya evaluator saja yang tersedia.
2. Standalone dengan command line argument: interpreter akan mencoba menjalankan argument tersebut sebagai file program Singkong. Apabila ini gagal, misal karena bukan merupakan file atau file tidak dapat diakses, maka argument tersebut akan dijalankan sebagai kode program Singkong.
3. Pustaka: dalam hal ini, interpreter Singkong di-embed ke dalam aplikasi lain. Kita akan membahas ini pada bagian lain dalam buku ini.

Properti-properti berikut dapat di-set ketika menjalankan interpreter Singkong:

- SINGKONG=0 (-DSINGKONG=0): menjalankan Singkong pada mode teks walaupun GUI tersedia. Apabila GUI memang tidak tersedia, setting ini tidak berefek apapun.
- DISABLE=<daftar_dipisahkan_koma> (-DDISABLE=<daftar_dipisahkan_koma>): menjalankan Singkong dengan sejumlah (atau semua) fungsi bawaan sengaja dinonaktifkan. Apabila item pertama dalam daftar adalah -, maka hanya fungsi-fungsi bawaan dalam daftar

tersebut yang akan diaktifkan. Untuk menonaktifkan semua fungsi bawaan, gunakanlah -DDISABLE=-. Properti ini dapat di-set untuk setiap tugas yang dibahas sebelumnya, namun untuk tugas (3), caranya berbeda.

Pada tugas (1), apabila sistem operasi atau desktop Anda mendukung, Anda bisa klik ganda pada file Singkong.jar. Apabila Anda perlu menjalankan dari command line, terdapat metode berikut:

A. Dengan menjalankan file jar (java -jar)
B. Dengan menjalankan main class dalam Singkong.jar dan memberikan daftar classpath (java -cp). Metode ini harus digunakan apabila kita ingin membuat program yang bekerja dengan sistem database relasional dengan driver selain yang disertakan dalam Singkong.jar (Apache Derby dan PostgreSQL), atau kita perlu memanggil method Java yang tersimpan dalam file class atau jar lainnya. Umumnya, metode ini tidak dimungkinkan dengan klik ganda pada file Singkong.jar.

Untuk (1A), berikut adalah contoh menjalankan Singkong.jar apabila klik ganda pada file tersebut tidak didukung oleh sistem operasi atau desktop yang digunakan, atau pengaturan tambahan ingin dilakukan (setiap contoh diketikkan sebagai satu baris perintah):

```
java –jar Singkong.jar
```

```
java –DSINGKONG=0 –jar Singkong.jar
```

```
java –DDISABLE=system,info –jar Singkong.jar
```

```
java -DSINGKONG=0 -DDISABLE=system -jar
Singkong.jar

java -DDISABLE=-,len,print -jar Singkong.jar
```

Untuk (1B), berikut adalah contoh menjalankan main class dalam Singkong.jar (setiap contoh diketikkan sebagai satu baris perintah). Pembahasan lebih lanjut akan dilakukan ketika membahas tentang bekerja dengan method Java. Perhatikanlah bahwa nama main class dalam Singkong.jar adalah `com.noprianto.singkong.Singkong`.

```
java -cp Singkong.jar
com.noprianto.singkong.Singkong


java -DSINGKONG=0 -cp Singkong.jar
com.noprianto.singkong.Singkong


java -DSINGKONG=0 -DDISABLE=info -cp
Singkong.jar com.noprianto.singkong.Singkong
```

Untuk tugas (2), berikut adalah contoh menjalankan Singkong secara standalone dengan command line argument:

```
java -jar Singkong.jar sort.singkong

java -cp Singkong.jar:modules
com.noprianto.singkong.Singkong “singkong()"
```

2. Tipe Data

Setiap ekspresi di Singkong tetap memiliki tipe walaupun nilainya ditentukan ketika program berjalan. Setiap tipe memiliki operator yang didukung, termasuk operator yang didukung ketika operand lain merupakan tipe tertentu. Apabila memungkinkan, Singkong akan cukup longgar dalam aturan tipe data. Dan, sebagaimana bahasa pemrograman pada umumnya, kita bisa melakukan assignment nilai ke suatu variabel, dimana nama variabel memiliki aturan penamaan tertentu.

Assignment

Untuk memberikan nilai ke suatu variabel, atau identifier, kita menggunakan statement var. Penamaan variabel memiliki aturan sederhana berikut: (1) dimulai dengan huruf atau underscore dan dapat diikuti oleh huruf, underscore, atau digit.

Berikut adalah contoh assignment yang berhasil:

```
var nama = “Singkong"
var _nama = “Singkong"
var nama_bahasa = “Singkong"
var x1 = 12345
```

Berikut adalah contoh assignment yang gagal, disebabkan oleh nama variabel yang tidak sesuai dengan aturan penamaan:

```
>  var 1x = 12345
PARSER ERROR: [line: 1]  expected next token
to be IDENT, got NUMBER instead
PARSER ERROR: [line: 1]  no prefix parse
function for = found
ERROR: [line: 1] identifier not found: x
```

Pesan kesalahan tersebut dapat diartikan bahwa setelah kata kunci var, nama identifier (variabel) yang valid diharapkan. Namun, justru sebuah NUMBER (bilangan) yang ditemukan.

Aturan lain pemberian nama variabel adalah bahwa: (2) nama variabel harus belum digunakan sebagai nama fungsi bawaan.

Berikut adalah contoh assignment yang gagal, disebabkan oleh nama variabel yang merupakan salah satu nama fungsi bawaan:

```
>  var info = ""
ERROR: [line: 1] info is a built-in function
```

Informasi lebih lanjut tentang daftar fungsi bawaan akan dibahas pada bagian tersendiri, di dalam buku ini.

Akhir Statement dan Penanda Blok

Di Singkong, titik koma tidak diperlukan pada akhir baris statement. Contoh-contoh berikut tidak menyebabkan terjadinya kesalahan sintaks.

```
var a = "Singkong"
var b = "Nama: " + a
println(b)
```

atau (perintah diketikkan dalam satu baris):

```
var a = "Singkong" var b = "Nama: " + a
println(b)
```

atau (perintah diketikkan dalam satu baris):

```
var a = "Singkong"; var b = "Nama: " + a;
println(b)
```

Penanda blok diawali dengan sebuah { dan diakhiri dengan sebuah }. Blok digunakan pada seleksi/kondisi, perulangan, dan pembuatan fungsi.

Fungsi Built-in Terkait

Fungsi built-in (bawaan) akan kita bahas tersendiri. Namun, beberapa fungsi berikut terkait dengan tipe data. Sebagai contoh:

- builtins: mendapatkan daftar fungsi bawaan
- eval: mengevaluasi sebuah STRING sebagai kode program Singkong
- is: memeriksa apakah sebuah variabel merupakan tipe tertentu
- is_array_and_of: memeriksa apakah merupakan ARRAY dan setiap elemennya merupakan tipe tertentu
- is_array_of: memeriksa apakah setiap elemen dalam sebuah ARRAY merupakan tipe tertentu
- is_array_of_component_of: memeriksa apakah semua elemen dalam sebuah ARRAY dari COMPONENT merupakan tipe komponen GUI tertentu

- hash: mendapatkan hash code
- print, println, puts: mencetak representasi STRING dari nilai sebuah variabel. Fungsi println akan menambahkan newline.
- string: mendapatkan representasi STRING dari nilai sebuah variabel
- type: mengetahui tipe sebuah variabel
- types: mengetahui tipe-tipe data apa saja yang didukung
- variables: mendapatkan daftar variabel global

Komentar

Komentar pada source code diawali dengan # dan diakhiri dengan ; (titik koma). Setiap komentar dapat dituliskan lebih dari satu baris.

```
#
      ini
      adalah
      komentar
      yang terdiri dari
      beberapa baris
;
println("Singkong")

# ini juga merupakan komentar;
println("Programming")

# dan ini adalah komentar lainnya
;
println("Language")
```

Keyword

Berikut adalah daftar keyword dalam bahasa pemrograman Singkong (bisa didapatkan dengan fungsi built-in keywords):

#
ELSE
FALSE
FN
IF
LET
NULL
REPEAT
RETURN
TRUE
VAR

NULL

Di Singkong, NULL merupakan suatu tipe, dengan nilai adalah kata kunci null (tidak dibedakan huruf besar/kecil). Berbeda dengan sejumlah bahasa pemrograman lain, tipe NULL di sini tidak memiliki hubungan apapun dengan pointer ataupun konsep serupa.

Selain itu, NULL juga bukan merupakan nilai default sebuah variabel. Di Singkong, variabel tidak perlu dideklarasikan terlebih dahulu.

Di Singkong, NULL umum digunakan ketika suatu fungsi atau ekspresi mengembalikan nilai yang tidak seharusnya, tapi belum merupakan kesalahan yang menyebabkan program diterminasi. Atau, NULL dikembalikan ketika fungsi tidak mengembalikan nilai secara eksplisit.

Sebagai contoh, ekspresi pembagian antar bilangan berikut harusnya menghasilkan bilangan lain:

```
>  var x = 1/2
>  x
```

```
0.5000
```

Namun, ekspresi berikut akan bernilai null, walaupun pada sejumlah bahasa pemrograman lain merupakan suatu kesalahan yang berpotensi terminasi program:

```
>  var x = 1/0
```

Kegunaan lain dari NULL adalah ketika sebuah fungsi tidak perlu mengembalikan nilai tertentu, NULL bisa digunakan. Tidak berarti nilai yang tidak seharusnya seperti contoh sebelumnya, hanya nilai kembalian yang tidak diperlukan. Sebagai contoh, fungsi bawaan show akan menampilkan Frame (dibahas lebih lanjut pada pembahasan tentang pemrograman GUI). Fungsi ini tidak perlu mengembalikan nilai tertentu, dan oleh karenanya, mengembalikan NULL.

Operator untuk NULL adalah == dan !=.

Nilai null tidak dicetak atau ditampilkan. Untuk mencetaknya, gunakanlah fungsi print, println, puts, atau message.

NUMBER

Di Singkong, semua bilangan adalah NUMBER. Baik bilangan bulat seperti 1, 2, 3, atau bilangan desimal seperti 0.5. Pemisah desimal selalu adalah karakter titik, bukan koma.

Setiap bilangan secara default memiliki 4 digit setelah koma (ditampilkan atau tidak), dan memiliki total digit penting sejumlah 10240 digit (yang merupakan bilangan yang sangat

besar, untuk program yang mungkin akan dikembangkan dengan bahasa Singkong).

Bilangan di Singkong juga dapat digunakan untuk aplikasi finansial. Nilai sebuah bilangan adalah seperti apa yang direpresentasikan/ditampilkan.

Sejak Singkong versi 4.7, jumlah digit setelah koma dapat diatur, dalam rentang nilai 1 sampai 16. Untuk mengubah jumlah digit setelah koma tersebut, gunakanlah fungsi bawaan number_scale. Contoh:

```
>  _pi()
3.1416
>  number_scale(8)
true
>  _pi()
3.14159265
>  number_scale(16)
true
>  _pi()
3.1415926535897930
```

Beberapa nilai konstan diberikan lewat fungsi bawaan, sebagai contoh:

```
>  number_scale(15)
true
>  _e()
2.718281828459045
>  _pi()
3.141592653589793
```

Pengurutan akan dilakukan sebagaimana pengurutan bilangan yang kita ketahui, yaitu berdasarkan nilainya. Sebagai contoh, 0 lebih kecil dari 1, dan 1 sama dengan 1.00.

Operator yang didukung adalah:

Operator	Deskripsi
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
==	Sama dengan
!=	Tidak sama dengan
%	Sisa bagi
^	Pemangkatan
<	Lebih kecil dari
<=	Lebih kecil dari atau sama dengan
>	Lebih besar dari
>=	Lebih besar dari atau sama dengan

Fungsi bawaan terkait:

Kategori	Daftar fungsi
NUMBER	abs, array_number, chr, integer, integer_gcd, number, number_boolean, number_group, number_to_byte, random, round, sort_number, words_en, words_id
Sistem	delay

Kategori	Daftar fungsi
Singkong	require
Thread	thread, thread_alive, thread_join
Trigonometri (dalam derajat)	sin, cos, tan, asin, acos, atan
Konversi derajat dan radian	to_degrees, to_radians
Hyperbolic	sinh, cosh, tanh
Natural exponential, logarithm, square/cube root	exp, log, log10, sqrt, cbrt
Konversi basis bilangan	from_bin, from_hex, from_oct, to_bin, to_hex, to_oct
Operasi bitwise	number_xor, number_or, number_and, number_not
Operasi bit shift	left_shift, right_shift, unsigned_right_shift
Nilai konstan	_e, _pi

Untuk bekerja dengan mean, median, mode, dan range, gunakanlah modul bawaan util, seperti contoh berikut:

```
>  load_module("util")
>  var a = [0, 1, 2, 3, 4, 3, 2, 1, 0, 0]
>  mean(a)
1.6000
>  median(a)
1.5000
>  mode(a)
[0, 3]
>  range_(a)
4
```

Untuk pengelompokan bilangan (pemisah ribuan), gunakanlah fungsi bawaan number_group. Beberapa fungsi tambahan dari modul util juga dapat digunakan:

number_group_c_p
number_group_p_c
number_group_s_c
number_group_s_p

Fungsi-fungsi lain yang terkait NUMBER juga disediakan oleh modul util:

binomial_coefficient
binomial_distribution
factorial
geometric_distribution_failure
geometric_distribution_success
poisson_distribution

BOOLEAN

Di Singkong, nilai BOOLEAN adalah salah satu dari kata kunci true atau false (tidak dibedakan huruf besar atau kecil). Pengurutan akan dilakukan berdasarkan nilai, dimana false dianggap lebih kecil dari true.

Operator yang didukung adalah:

Operator	Deskripsi
==	Sama dengan
!=	Tidak sama dengan
&	and
\| | or |

Fungsi bawaan terkait:

Kategori	Daftar fungsi
BOOLEAN	number_boolean, boolean_xor, sort_boolean

STRING

Di Singkong, nilai STRING diapit oleh dua karakter kutip ganda (“), bukan kutip tunggal (‘). Panjang maksimum STRING tidak ditentukan.

Singkong tidak menyediakan cara untuk melakukan escape karakter spesial tertentu. Sebagai gantinya, fungsi built-in digunakan, sebagaimana dapat dilihat pada tabel berikut.

Fungsi	Deskripsi
cr	Carriage return
crlf	Carriage return diikuti dengan line feed
lf	Line feed
newline	Karakter penanda akhir baris sesuai sistem operasi yang digunakan
quote	Karakter kutip ganda “
tab	Karakter tab

Dengan demikian, sebagai contoh, untuk menambahkan karakter kutip ganda “ di dalam sebuah STRING, kita bisa menggunakan cara berikut:

```
>  var s = "Nama bahasa pemrograman ini
adalah: " + quote() + "Singkong" + quote()
>  s
"Nama bahasa pemrograman ini adalah:
“Singkong""
```

Perbandingan nilai sebuah STRING, apakah sama atau tidak, bisa menggunakan operator ==. Namun, perbandingan akan memperhatikan huruf besar dan huruf kecil. Untuk perbandingan tanpa membedakan huruf besar/kecil, gunakanlah fungsi built-in equals.

Pengurutan STRING dilakukan secara lexicographically, sebagaimana contoh berikut:

```
> sort_string(["Singkong", "Programming",
"Language"])
["Language", "Programming", “Singkong"]
```

Operator yang didukung adalah:

Operator	Deskripsi
+	Penggabungan
-	Menghapus STRING lain di dalam suatu STRING. Misal "Singkong" - “Sing" akan menghasilkan “kong”.
==	Sama dengan, dilakukan secara case-sensitive
!=	Tidak sama dengan
*	Pengulangan. Misal "Singkong " * 3 akan menghasilkan "Singkong Singkong Singkong “

Fungsi bawaan terkait:

Kategori	Daftar fungsi
STRING	array_string, base64_decode, base64_encode, call, center, count, cr, crlf, dir, empty, endswith, equals, eval, in, index, isalnum, isalpha, isdigit, islower, isupper, join, left, len, lf, lower, matches, md5, md5_file, newline, ord, quote, random_string, replace, right, set, sha1, sha1_file, sha256, sha256_file, sha384, sha384_file, sha512, sha512_file, slice, sort_string, split, startswith, stat, tab, trim, upper
Sistem	cwd, inet_address_local, separator, user, userhome
File	abs, append, copy, copy_resource, delete, mkdir, read, rename, temp_file, write
HTTP client	http_delete, http_get, http_get_file, http_head, http_post, http_post_override, http_put, url_decode, url_encode
Singkong	load, load_file_or_resource, load_module, load_resource
Extended	x_base64_decode_file, x_base64_encode_file

Untuk mengakses elemen tertentu dari STRING, kita bisa menggunakan indeks, yang dimulai dari 0. Apabila indeks lebih kecil dari 0 atau lebih besar dari jumlah karakter - 1, maka NULL akan dikembalikan.

Untuk mengubah indeks tertentu dari STRING, gunakanlah fungsi bawaan set. Sebagai contoh:

```
>  var s = "singkong"
>  s
"singkong"

>  set(s, 0, "S")
"Singkong"

>  s
"Singkong"
```

Untuk mendapatkan panjang sebuah STRING, fungsi built-in len dapat digunakan. Untuk mendapatkan bagian dari sebuah STRING (substring), gunakanlah fungsi built-in slice.

Untuk contoh enkripsi/dekripsi STRING atau file sederhana, gunakanlah fungsi-fungsi dari modul util, seperti contoh berikut:

String:

```
load_module("util")
var a = "Hello World"
var b = "Singkong"
var c = simple_string_encrypt(a, b)
var d = simple_string_decrypt(c, b)
println(a, b, a == d)
```

File:

```
load_module("util")
var a = "input.txt"
var b = "Singkong"
var c = "input.data"
var d = "output.txt"
write(a, "Hello World")
var e = simple_file_encrypt(a, b, c)
var f = simple_file_decrypt(c, b, d)
println(e, f, md5_file(a) == md5_file(d))
```

ARRAY

Di Singkong, sebuah ARRAY merupakan kumpulan elemen dengan urutan, dari berbagai nilai dengan tipe yang berbeda-beda, termasuk NULL dan ARRAY. Panjang sebuah ARRAY tidak dibatasi.

Sebuah nilai ARRAY diketikkan dengan diawali oleh sebuah [dan diakhiri dengan sebuah]. Elemen di dalam ARRAY dipisahkan dengan sebuah koma.

Perbandingan ARRAY dilakukan dengan operator ==, yang mana akan membandingkan semua elemen dalam ARRAY, termasuk ARRAY di dalam ARRAY, dan seterusnya. Dengan demikian, dua ARRAY adalah sama jika dan hanya jika semua elemen di dalamnya benar-benar sama. Sebagai contoh:

```
>  var a = [ [], [[]], [[], 1, [2,3]], null,
true, {}, 123]
>  var b = [ [], [[]], [[], 1, [2,3]], null,
true, {}, 123]
>  var c = [ [], [[]], [[[]], 1, [2,3]], null,
true, {}, 123]
>  a == b
true

>  a == c
false
```

Perhatikanlah bahwa operator == memperhatikan urutan elemen di dalam ARRAY. Kedua ARRAY dianggap sama (dengan operator ==) apabila elemen dan urutan elemennya benar-benar sama.

```
>  [] == []
true
>  [1,2] == [1,2]
true
>  [1,2] == [2,1]
false
>  [[1,2]] == [[2,1]]
false
>  [1,2] == [1,2,3]
false
```

Apabila urutan elemen tidak ingin ikut dibandingkan, gunakanlah fungsi array_equals:

```
>  [1,2] == [2,1]
false
>  array_equals([1,2], [2,1])
true
>  array_equals([1,2], [1,2])
true
```

Pengurutan ARRAY dilakukan dengan membandingkan jumlah elemen dalam ARRAY. Dengan demikian, ARRAY dengan elemen lebih sedikit dianggap lebih kecil. Sebagai contoh:

```
>  sort_array([ [1,2,3], [], [1,2] ])
[[], [1, 2], [1, 2, 3]]
```

Operator yang didukung adalah:

Operator	Deskripsi
+	Menambahkan elemen ke dalam ARRAY. Sebagai contoh, [] + [] akan menghasilkan [[]].
-	Menghapus elemen dari sebuah ARRAY. Sebagai contoh [true, true, false, false] - false akan menghasilkan [true, true].
==	Sama dengan
!=	Tidak sama dengan

Untuk mengakses elemen tertentu dari ARRAY, kita bisa menggunakan indeks, yang dimulai dari 0. Apabila indeks lebih kecil dari 0 atau lebih besar dari jumlah elemen - 1, maka NULL akan dikembalikan. Sebagai contoh:

```
>  [1,2,3][0]
1


>  [1,2,3][3]
```

Namun, kita tidak bisa mengubah elemen tertentu dalam ARRAY dengan operator index. Sebagai gantinya, mirip dengan STRING, kita gunakan fungsi built-in set. Sebagai contoh:

```
>  var a = [0, 1, 2]
>  a
[0, 1, 2]

>  set(a, 0, true)
[true, 1, 2]
```

```
>  a
[true, 1, 2]
```

Untuk mendapatkan jumlah elemen dalam ARRAY, fungsi built-in len dapat digunakan. Untuk mendapatkan irisan dari sebuah ARRAY, gunakanlah fungsi built-in slice.

Untuk mengopi ARRAY, perhatikanlah contoh kode berikut:

```
var a = ["hello", "world"]
var b = []
each(a, fn(e,i) {
  var b = b + e
 })
set(a, 1, "hello world")
println(a, b)
```

Apabila menggunakan Singkong versi 4.6 atau lebih baru, pengopian ARRAY juga dapat dilakukan dengan fungsi array_copy dari modul util:

```
load_module("util")
var a = ["hello", "world"]
var b = array_copy(a)
set(a, 1, "hello world")
println(a, b)
```

Sebagaimana dicontohkan dalam tabel operator ARRAY, operator + akan menambahkan ke dalam sebuah ARRAY. Apabila yang ditambahkan adalah sebuah ARRAY, maka hasilnya adalah ARRAY dalam ARRAY:

```
>  [1,2] + 3
[1, 2, 3]
>  [1,2] + [3,4]
[1, 2, [3, 4]]
```

Apabila yang diinginkan adalah menggabungkan ARRAY, gunakanlah fungsi array_extend:

```
>  array_extend([1,2],[3,4])
[1, 2, 3, 4]
>  array_extend_all([[1,2], [3,4], 5])
[1, 2, 3, 4, 5]
```

Perbedaan (diff) ARRAY bisa didapatkan dengan fungsi array_diff dari modul util:

```
>  array_diff([1,2,3,4,5], [1,3,5])
[2, 4]
```

Rectangular array

Rectangular array adalah array dimana setiap elemen di dalamnya juga merupakan array, dan semua elemen memiliki panjang (jumlah elemen) yang sama.

Beberapa fungsi bawaan berikut dapat digunakan ketika bekerja dengan rectangular array: is_rect_array, is_rect_array_of, rect_array_size, rect_array_size_of.

Perhatikanlah contoh berikut:

```
        >  var a = []
        >  var b = [[], []]
        >  var c = [[1, 2, 3], ["Singkong",
"Programming", "Language"]]
        >  var d = [[1, 2, 3], [4, 5, 6]]
        >  is_rect_array(a)
        false
        >  is_rect_array(b)
        true
        >  is_rect_array(c)
```

```
        true
        >  is_rect_array_of(c, "NUMBER")
        false
        >  is_rect_array_of(d, "NUMBER")
        true
        >  rect_array_size(c)
        [2, 3]
        >  rect_array_size_of(c, "NUMBER")
         >  rect_array_size_of(c, "NUMBER") ==
null
        true
        >  rect_array_size_of(d, "NUMBER")
        [2, 3]
```

Untuk mengurutkan rectangular array NUMBER berdasarkan indeksnya, gunakanlah fungsi (dari modul util) sort_rect_array_of_number_by_index. Sebagai contoh:

```
       >  load_module("util")
       >  sort_rect_array_of_number_by_index([[1,3],
[3,2], [2,1]], 1)
       [[2, 1], [3, 2], [1, 3]]
       >  sort_rect_array_of_number_by_index([[1,3],
[3,2], [2,1]], 0)
       [[1, 3], [2, 1], [3, 2]]
```

Fungsi bawaan terkait:

Kategori	Daftar fungsi
ARRAY	array, array_equals, array_extend, array_extend_all, array_number, array_string, average, call, count, each, empty, first, in, index, join, last, len, max, min, pop, push, random, range, rest, reverse, shuffle, slice, sort_array, sort_boolean, sort_date, sort_hash, sort_number, sort_string, sum

Kategori	Daftar fungsi
Cek	is_array_and_of, is_array_of, is_array_of_component_of
Sistem	arguments, inet_address, runtime_version, system
File	dir, read_byte, write_byte
Singkong	builtins, modules, singkong_interpreter
CGI	cgi_contents
HTTP client	http_response_ok
byte array	number_to_byte_array, string_from_byte_array, string_to_byte_array
Rectangular array	is_rect_array, is_rect_array_of, rect_array_size, rect_array_size_of

Kategori	Daftar fungsi
Fungsi tambahan untuk Rectangular array dan matriks (modul: rect_array_util)	add_rect_array_of, add_rect_array_of_number, cross_product_3d_rect_array_column_of, cross_product_3d_rect_array_column_of_number, cross_product_3d_rect_array_row_of, cross_product_3d_rect_array_row_of_number, determinant_rect_array_of, determinant_rect_array_of_number, inverse_rect_array_of, inverse_rect_array_of_number, is_constant_rect_array_of, is_constant_rect_array_of_number, is_diagonal_rect_array_of, is_diagonal_rect_array_of_number, is_identity_rect_array_of_number, is_lower_triangular_rect_array_of, is_lower_triangular_rect_array_of_number, is_rect_array_column_of, is_rect_array_column_of_number, is_rect_array_row_of, is_rect_array_row_of_number, is_square_rect_array_of, is_square_rect_array_of_number, is_symmetric_rect_array_of, is_symmetric_rect_array_of_number, is_upper_triangular_rect_array_of, is_upper_triangular_rect_array_of_number, is_zero_rect_array_of, is_zero_rect_array_of_number, minor_rect_array_of, minor_rect_array_of_number, mul_rect_array_of, mul_rect_array_of_number, mul_scalar_rect_array_of, mul_scalar_rect_array_of_number, rect_array_not_row_col, sub_rect_array_of, sub_rect_array_of_number, trace_rect_array_of, trace_rect_array_of_number, transpose_rect_array_of, transpose_rect_array_of_number

Byte array

Untuk bekerja dengan byte array (misal dalam bekerja dengan file binary), selain fungsi read_byte dan write_byte, Anda mungkin ingin melakukan konversi dari NUMBER dan dari/ke STRING. Untuk itu, fungsi-fungsi number_to_byte_array, string_from_byte_array, dan string_to_byte_array dapat dipergunakan, seperti contoh berikut:

```
> number_to_byte_array(123)
[123]
> number_to_byte_array(1234)
[4, -46]
> string_to_byte_array("Singkong")
[83, 105, 110, 103, 107, 111, 110,
103]
> string_from_byte_array([83, 105,
110, 103, 107, 111, 110, 103])
"Singkong"
```

Set

Untuk bekerja dengan dasar-dasar set (diemulasikan dengan ARRAY), modul set_util datang dengan fungsi-fungsi berikut:

- create_relation_from_array
- create_set_from_array
- get_relation_part
- inverse_bijective_function_set
- is_antisymmetric_relation_set
- is_bijective_function_set
- is_function_set
- is_injective_function_set
- is_reflexive_relation_set
- is_relation
- is_relation_set
- is_same_set
- is_sub_set
- is_surjective_function_set
- is_symmetric_relation_set
- is_transitive_relation_set

set_diff
set_intersection
set_power
set_product
set_union

Fungsi-fungsi lain yang terkait dengan ARRAY, yang disediakan oleh modul util:

standard_deviation
standard_deviation_
standard_deviation_sample
variance
variance_
variance_sample

HASH

Di Singkong, HASH merupakan pemetaan dari key ke value, di mana key dan value dapat berupa nilai apapun, termasuk HASH dan NULL. Jumlah pemetaan di dalam HASH tidak dibatasi.

Sebuah nilai HASH diketikkan dengan diawali oleh sebuah { dan diakhiri dengan sebuah }. Pemetaan di dalam HASH dipisahkan dengan sebuah koma, sementara key dan value dipisahkan dengan sebuah tanda titik dua.

Perbandingan HASH dilakukan dengan operator ==, yang mana akan membandingkan semua pemetaan dalam HASH, termasuk HASH di dalam HASH, dan seterusnya. Dengan demikian, dua HASH adalah sama jika dan hanya jika semua pemetaan di dalamnya benar-benar sama. Sebagai contoh:

```
>  var a = {"name": "Singkong", "age": 1, !
true: false, []: [1,2,3]}
```

```
>  var b = {"name": "Singkong", "age": 1, !
true: false, []: [1,2,3]}
>  var c = {"name": "Singkong", "age": 1, !
true: false, [[]]: [1,2,3]}
>  a == b
true

>  a == c
false
```

Pengurutan HASH dilakukan dengan membandingkan jumlah pemetaan dalam HASH. Dengan demikian, HASH dengan pemetaan lebih sedikit dianggap lebih kecil. Sebagai contoh:

```
>  sort_hash([{1:2, 3:4}, {}, {1:2}])
[{}, {1: 2}, {1: 2, 3: 4}]
```

Operator yang didukung adalah:

Operator	Deskripsi
+	Menambahkan pemetaan ke dalam HASH. Sebagai contoh, {} + {"name": “Singkong"} akan menghasilkan {"name": “Singkong"}.
-	Menghapus pemetaan dari sebuah HASH berdasarkan key. Sebagai contoh {0: "nol", 1: "satu", 2: "dua"} - 0 akan menghasilkan {1: "satu", 2: “dua"}.
==	Sama dengan
!=	Tidak sama dengan

Mirip dengan ARRAY, HASH di Singkong menjaga urutan pemetaan ditambahkan dan sama-sama bekerja dengan operator indeks. Bedanya, HASH menggunakan indeks

berupa key. Apabila bilangan diberikan sebagai indeks (sebagaimana halnya pada ARRAY), maka interpreter Singkong akan menganggap bahwa kita ingin mendapatkan value dari key berupa bilangan tersebut. Apabila key tidak ditemukan, NULL akan dikembalikan.

```
>  {"name": "Singkong"}["name"]
"Singkong"

>  {"name": "Singkong"}[0]
```

Kita tidak bisa mengubah pemetaan tertentu dalam HASH dengan operator index. Sebagai gantinya, mirip dengan STRING dan ARRAY, kita gunakan fungsi built-in set. Sebagai contoh:

```
>  var h = {"name": ""}
>  h
{"name": ""}

>  set(h, "name", "Singkong")
{"name": "Singkong"}

>  h
{"name": "Singkong"}
```

Untuk mendapatkan jumlah pemetaan dalam HASH, fungsi built-in len dapat digunakan.

Untuk mengopi HASH, perhatikanlah contoh kode berikut:

```
var a = {"hello": "world"}
var b = {} + a
set(a, "hello", "hello world")
println(a, b)
```

Apabila menggunakan Singkong versi 4.6 atau lebih baru, pengopian HASH juga dapat dilakukan dengan fungsi hash_copy dari modul util:

```
load_module("util")
var a = {"hello": "world"}
var b = hash_copy(a)
set(a, "hello", "hello world")
println(a, b)
```

Fungsi bawaan terkait:

Kategori	Daftar fungsi
HASH	empty, keys, len, parse_hash, random, sort_hash, values
Sistem	env, info
File	properties_read, properties_write, stat
Singkong	singkong
CGI	cgi_header, cgi_get, cgi_post, cgi_post_hash

DATE

Di Singkong, DATE digunakan untuk merepresentasikan tanggal ataupun waktu. Apabila komponen waktu tidak diberikan, maka secara otomatis akan dianggap jam 00:00:00.

Sebuah nilai DATE diberikan sebagaimana perincian dalam tabel berikut:

DATE	Deskripsi
@	Tanggal dan waktu aktif

DATE	Deskripsi
@Y	Hanya tahun saja, 1 digit (bulan 1, tanggal 1, jam 00:00:00)
@YY	Hanya tahun saja, 2 digit (bulan 1, tanggal 1, jam 00:00:00)
@YYY	Hanya tahun saja, 3 digit (bulan 1, tanggal 1, jam 00:00:00)
@YYYY	Hanya tahun saja, 4 digit (bulan 1, tanggal 1, jam 00:00:00)
@YYYYM	Hanya tahun dan bulan 1 digit (tanggal 1, jam 00:00:00)
@YYYYMM	Hanya tahun dan bulan 2 digit (tanggal 1, jam 00:00:00)
@YYYYMMD	Hanya tahun, bulan, dan tanggal 1 digit (jam 00:00:00)
@YYYYMMDD	Hanya tahun, bulan, dan tanggal 2 digit (jam 00:00:00)
@YYYYMMDDh	Tahun, bulan, tanggal, dan jam 1 digit (menit 00:00)
@YYYYMMDDhh	Tahun, bulan, tanggal, dan jam 2 digit (menit 00:00)
@YYYYMMDDhhm	Tahun, bulan, tanggal, jam, dan menit 1 digit (detik 00)
@YYYYMMDDhhmm	Tahun, bulan, tanggal, jam, dan menit 2 digit (detik 00)
@YYYYMMDDhhmms	Tahun, bulan, tanggal, jam, menit, detik 1 digit
@YYYYMMDDhhmmss	Tahun, bulan, tanggal, jam, menit, detik 2 digit

Perbandingan DATE dilakukan dengan operator ==, yang akan membandingkan komponen waktu dalam DATE, sesuai kelengkapan komponen waktu yang diberikan, dengan mempertimbangkan nilai default apabila komponen waktu tertentu tidak diberikan. Sebagai contoh:

```
>  @2020 == @2020
true

>  @20201 == @2020
true

>  @20203 == @2020
false

>  @ == @2020
false
```

Untuk melihat bagaimana Singkong memberikan nilai default pada komponen waktu apabila tidak diberikan, lihatlah pada tabel daftar nama variabel, atau ketikkanlah pada evaluator. Contoh:

```
>  @2020
2020-01-01 00:00:00
```

Pengurutan DATE dilakukan dengan mempertimbangkan sebelum atau sesudah. Dengan demikian, 2019 akan dianggap lebih kecil dari 2020. Sebagai contoh:

```
>  sort_date([@2020, @2019, @2021])
[2019-01-01 00:00:00, 2020-01-01 00:00:00,
2021-01-01 00:00:00]
```

Selain == dan !=, tidak ada operator lain yang bekerja dengan DATE. Untuk mengurangi atau menambahkan komponen waktu dalam DATE, gunakanlah fungsi-fungsi built-in berikut:

Fungsi	Deskripsi
second	Menambahkan/mengurangi detik. Akan menambahkan/mengurangi menit, jam, dan seterusnya apabila diperlukan.
minute	Menambahkan/mengurangi menit. Akan menambahkan/mengurangi jam, hari, dan seterusnya apabila diperlukan.
hour	Menambahkan/mengurangi jam. Akan menambahkan/mengurangi hari, bulan, dan seterusnya apabila diperlukan.
day	Menambahkan/mengurangi hari. Akan menambahkan/mengurangi bulan dan tahun apabila diperlukan.
month	Menambahkan/mengurangi bulan. Akan menambahkan/mengurangi tahun apabila diperlukan.
year	Menambahkan/mengurangi tahun

Fungsi bawaan terkait:

Kategori	Daftar fungsi
DATE	date, datetime, day, days_of_month, diff, format_date, format_datetime, format_diff, format_time, hour, is_leap_year, minute, month, part, second, sort_date, year
File	stat

Untuk mendapatkan komponen dalam sebuah DATE, gunakanlah fungsi part. Sebagai contoh:

```
>  var now = @
>  now
2021-05-07 10:24:49

>  part(now)
[2021, 5, 7, 10, 24, 49]

>  is_leap_year(part(now)[0])
false
```

FUNCTION

Di Singkong, FUNCTION merupakan tipe untuk fungsi yang dibuat oleh programmer. Singkong mendukung first-class function, sehingga fungsi yang dibuat bisa dilewatkan sebagai argumen pada saat pemanggilan fungsi lain, digunakan sebagai key dalam HASH sebagaimana nilai lainnya, dan sebagainya.

Untuk membuat fungsi, kata kunci yang dipergunakan adalah fn, dan fungsi yang dibuat kemudian di-assign ke sebuah variabel dengan var, menjadi nama fungsi. Berikut adalah contoh pembuatan fungsi dengan nama f, yang tidak menerima parameter dan tidak mengembalikan nilai secara eksplisit.

```
var f = fn(){println(“Singkong")}
```

Perhatikanlah (dan) yang mengikuti fn pada contoh sebelumnya. Apabila fungsi menerima parameter, maka deretkanlah diantara (dan), dipisahkan koma. Cukup nama

parameter saja yang disebut. Karena contoh fungsi tersebut tidak menerima parameter apapun, maka daftar ini kosong.

Sebagaimana dibahas sebelumnya, tubuh fungsi adalah blok, yang dituliskan di dalam { dan }.

Documentation String

Sejak Singkong versi 4.6, sebuah string dokumentasi dapat ditambahkan ketika fungsi dibuat. Dokumentasi tersebut, selain berguna sebagai dokumentasi untuk melengkapi komentar pada source code, juga dapat diakses lewat fungsi built-in help ketika program berjalan.

Contoh tanpa documentation string:

```
>  var f = fn(x) {x}
>  f
fn(x)
{
x;
}


>  help(f)
“"
```

Contoh dengan documentation string:

```
>  var f = fn(x) "test" {x}
>  f
[documentation string] test

>  help(f)
"[documentation string] test"
```

Untuk memanggil fungsi yang telah dibuat, tuliskanlah nama fungsi, diikuti oleh (dan argumen apabila ada, kemudian

ditutup dengan). Nilai kembalian dari pemanggilan fungsi bisa di-assign ke suatu variabel dengan var. Berikut adalah contoh pemanggilan fungsi yang dibuat sebelumnya:

```
>  f()
Singkong
```

Perhatikanlah bahwa fungsi tersebut tidak secara eksplisit mengembalikan nilai. Apabila kita ingin assign ke sebuah variabel, maka variabel tersebut akan bernilai null.

```
>  var r = f()
Singkong
```

```
>  type(r)
"NULL"
```

Mari kita buat fungsi lain yang menerima sebuah parameter dan mengembalikan nilai:

```
>  var test = fn(x) {x}
```

Pengembalian nilai bisa dengan cara tersebut ataupun dengan kata kunci return. Dengan demikian, kedua fungsi test dan testing berikut adalah sama.

```
>  var testing = fn(x) {return x}
```

Ketika kedua fungsi yang dibuat sebelumnya dipanggil, kita perlu memberikan sebuah argumen. Apabila tidak maka kesalahan akan terjadi.

```
>  test(10)
10

>  testing(20)
20

>  test()
ERROR: [line: 1] wrong number of arguments,
got=0, want=1
```

Pesan kesalahan pada contoh pemanggilan fungsi terakhir menginformasikan bahwa fungsi tersebut membutuhkan sebuah argumen.

Untuk mengetahui berapa parameter yang dibutuhkan oleh sebuah fungsi, gunakanlah fungsi built-in param, seperti contoh berikut:

```
>  param(test)
1

>  param(f)
0
```

Karena kedua fungsi terakhir mengembalikan nilai secara eksplisit, kita dapat pula meng-assign ke sebuah variabel, seperti contoh berikut:

```
>  var r = test(10)
>  r
10
```

Contoh berikut adalah fungsi yang menerima dua parameter dengan mengembalikan sebuah ARRAY:

```
>  var a = fn(x, y) {[x, y]}
>  a(1,2)
[1, 2]
```

Tentu saja, sebagaimana halnya assignment nilai ke variabel, nama fungsi juga tidak boleh menggunakan nama fungsi built-in dan harus mengikuti aturan lain penamaan variabel. Berikut adalah contoh yang tidak benar:

```
>  var info = fn(){}
ERROR: [line: 1] info is a built-in function

>  var 10 = fn(){}
PARSER ERROR: [line: 1]  expected next token
to be IDENT, got NUMBER instead
```

KIta dapat melewatkan sebuah fungsi sebagai argumen dalam pemanggilan fungsi lain. Fungsi di Singkong dapat diperlakukan sebagaimana nilai atau tipe lainnya. Sebagai contoh:

```
>  var p = fn(f) {param(f)}
>  p(f)
0
```

```
>  p(test)
1

>  p(testing)
1
```

Perhatikanlah bahwa fungsi dapat memiliki variabel lokal yang hanya tersedia dalam fungsi, namun juga dapat mengakses variabel global. Perhatikanlah contoh berikut, dimana a adalah variabel global dan x adalah lokal terhadap f:

```
>  var a = 1
>  var f = fn(x) {println(x); println(a)}
>  a
1

>  f(10)
10
1

>  x
ERROR: [line: 1] identifier not found: x
```

Sebuah fungsi bisa dibandingkan dengan fungsi lainnya dengan operator == atau !=, dimana perbandingan jumlah parameter, nama parameter, dan isi fungsi akan dilakukan. Perhatikanlah contoh-contoh berikut:

```
>  var func1 = fn(x) {x+1}
>  var func2 = fn(y) {y+1}
>  var func3 = fn(x, y) {[x, y]}
>  var func4 = fn(a, b) {[a, b]}
>  var func5 = fn(x, y) {[y, x]}
>  func1 == func2
false
```

```
>  func3 == func4
false

>  func3 == func5
false

>  func1 == fn(x) {x+1}
true

>  func1 == fn(x) {1+x}
false
```

Dari contoh tersebut, func1 dan func2 tidaklah menawarkan fungsionalitas yang berbeda. Akan tetapi, mereka dianggap tidak sama karena walaupun sama-sama membutuhkan satu parameter, namanya beda.

Bahkan, pada dua contoh perbandingan terakhir, fn(x) {x+1} dan fn(x) {1+x} juga dianggap berbeda.

Fungsi di Singkong tidak harus memiliki nama. Contoh berikut sepenuhnya valid dan cara demikian dapat ditemukan pada pengembangan aplikasi GUI dengan Singkong.

```
>  fn(x) {x}(1000)
1000

>  [fn(x){x}][0](2000)
2000
```

Fungsi bawaan terkait:

Kategori	Daftar fungsi
FUNCTION	do, each, help, param
Singkong	load, load_file_or_resource, load_module, load_resource
Thread	thread

Singkong mendukung fungsi di dalam fungsi. Walau demikian, perhatikanlah hal-hal berikut, terutama ketika pemrograman GUI dilakukan dan fungsi di dalam fungsi kerap digunakan (misal sebagai event handler).

Contoh kode:

```
var a = fn() {
        println("a")
        var b = fn() {
                println("b")
                var c = fn() {
                        println("c")
                        var d = fn() {
                                println("d")
                        }
                        d()
                }
                c()
        }
        b()
}
a()
```

Keluaran:

```
a
b
c
d
```

Untuk scope variable dalam fungsi di dalam fungsi, perhatikanlah contoh kode berikut (baru, sejak Singkong versi 6.0):

```
        var x = "Hello"
        println("X: " + x)
        var a = fn() {
            println("Function A (X): " + x)
            var b = fn() {
                println("Function B in A (X): " + x)
                var c = fn() {
                    # x defined in c;
                    var x = "World"
                    println("Function C in B (local
variable X): " + x)
                    var d = fn() {
                        println("Function D in C (using
variable X in C): " + x)
                    }
                    d()
                }
                c()
                # original x;
                println("B (original X): " + x)
            }
            b()
            # not found;
            println(y)
        }
        a()
```

Keluaran:

```
        X: Hello
        Function A (X): Hello
        Function B in A (X): Hello
        Function C in B (local variable X): World
        Function D in C (using variable X in C): World
        B (original X): Hello
        ERROR: [line: 22] identifier not found: y
        ERROR: [line: 22] identifier not found: y
```

BUILT-IN

Pada saat buku ini ditulis, Singkong datang dengan 339 fungsi built-in (bawaan) yang menyediakan berbagai fungsionalitas.

Sejak Singkong versi 4.2, disertakan juga fungsi-fungsi extended, yang mana membutuhkan Java versi > 5.0. Nama fungsi-fungsi tersebut berawalan x_. Apabila kebutuhan versi minimum Java tidak terpenuhi, fungsi-fungsi tersebut tetap akan tersedia, hanya saja akan mengembalikan NULL.

Masing-masing dari fungsi bawaan bertipe BUILTIN, dan sebagaimana halnya FUNCTION di Singkong, built-in juga dapat diperlakukan sebagaimana nilai atau tipe lainnya.

```
>  var panjang = len
>  panjang([1,2,3])
3

>  var f = fn(x){x("Singkong")}
>  f(len)
8
```

Untuk mengetahui berapa parameter yang dibutuhkan oleh sebuah fungsi built-in, gunakanlah fungsi built-in param, seperti contoh berikut:

```
>  param(len)
1
```

Ketika berada pada interactive evaluator, kita dapat mengetikkan nama fungsi built-in tanpa (dan), dan informasi mendasar fungsi tersebut akan ditampilkan. Apabila Anda membutuhkan informasi ini sebagai STRING, gunakanlah fungsi built-in help.

Untuk mendapatkan ARRAY berisikan nama semua fungsi built-in yang tersedia, gunakanlah fungsi built-in builtins.

Fungsi-fungsi built-in tertentu mungkin dinonaktifkan ketika interpreter Singkong dijalankan. Untuk mengetahui fungsi-fungsi built-in apa saja yang dinonaktifkan, gunakanlah fungsi built-in disabled (kecuali, fungsi disabled juga dinonaktifkan).

Pada contoh berikut, interpreter Singkong dijalankan dengan fungsi built-in info dan system dinonaktifkan:

```
java -DDISABLE=info,system -jar Singkong.jar
```

Walaupun dinonaktifkan, kita tetap tidak bisa menggunakan nama fungsi built-in tersebut sebagai variabel:

```
>  var info = 123
ERROR: [line: 1] info is a built-in function
```

```
>  info()
ERROR: [line: 1] built-in function "info" is
disabled
```

```
>  disabled()
["INFO", "SYSTEM"]
```

Sebuah fungsi built-in bisa dibandingkan dengan fungsi built-in lainnya dengan operator == ataupun !=.

```
>  len == len
true

>  len == print
false

>  info == info
true

>  info == system
false


>  var panjang = len
>  panjang == len
true
```

Fungsi bawaan terkait:

Kategori	Daftar fungsi
BUILTIN	builtins, disabled, help, param

COMPONENT

Singkong mendukung pengembangan aplikasi GUI (Graphical User Interface) sederhana, dimana setiap komponen user interface (seperti button, combobox, dan lainnya) adalah sebuah COMPONENT.

Terdapat sejumlah fungsi built-in untuk bekerja dengan COMPONENT. Karena GUI bisa saja tidak tersedia (misal pada lingkungan kerja tanpa GUI atau karena interpreter Singkong dijalankan tanpa GUI dengan -DSINGKONG=0), maka fungsi-fungsi tersebut, walaupun tersedia, ketika dipanggil akan menyebabkan terjadinya kesalahan.

Sebagai contoh:

```
java –DSINGKONG=0 –jar Singkong.jar

> component("button","")
ERROR: [line: 1] GUI is not available
```

Oleh karena itu, apabila dirasa perlu, periksalah apakah GUI tersedia atau tidak dengan menggunakan fungsi built-in gui, seperti pada contoh berikut:

```
> gui()
false
```

Sebuah COMPONENT dapat pula dibandingkan dengan COMPONENT lain dengan operator == atau !=, yang mana hanya akan sama apabila merujuk ke COMPONENT itu sendiri.

```
>  var b = component("button","Singkong")
>  var c = component("button","Singkong")
>  b == c
false

>  b == b
true

>  var d = b
>  d == b
true
```

Fungsi bawaan terkait dapat dilihat pada halaman berikut.

Kategori	Daftar fungsi
COMPONENT	add, add_e, add_n, add_s, add_w, button_image, clear, closing, component, component_info, component_type, components, config, event, event_focus, event_frame, event_keyboard_frame, event_mouse, event_mouse_frame, frame, frame_close, frame_location, frame_top, fonts, get, grid_add, grid_clear, grid_remove, gui, hide, menubar, panel_add, panel_clear, panel_remove, popup_component, popup_hide, popup_show, printer, radio_group, remove, remove_e, remove_n, remove_s, remove_w, reset, resizable, screen, show, size, start, start_timer, statusbar, stop, stop_timer, tab_add, tab_clear, tab_remove, table_add, table_bottom, table_center, table_column, table_column_count, table_column_info, table_column_name, table_get_value, table_left, table_middle, table_print, table_remove, table_right, table_row_count, table_scroll, table_set_value, table_top, timer, timer_running, title, wait
Input/Output (versi sederhana apabila GUI tidak tersedia)	confirm, directory, input, open, message, password, save
Extended	x_edit_print
Dialog (GUI)	color_chooser, login_dialog, panel_dialog

Kategori	Daftar fungsi
Drawing	draw_width, draw_rect, fill_rect, draw_arc, fill_arc, draw_oval, fill_oval, draw_round_rect, fill_round_rect, draw_string, draw_line, draw_polygon, fill_polygon, draw_polyline, draw_read, draw_write_png, draw_write_jpg, draw_write_bmp, draw_get_pixel, draw_set_pixel
Clipboard	clipboard_get, clipboard_set
Sound	beep, play_sound
Array Component	is_array_of_component_of

Untuk fungsi tambahan terkait user interface, gunakanlah fungsi-fungsi dari modul ui_util:

table_add_fill
table_add_row_fill
table_fill
table_get_array_
table_get_array_number
table_get_array_string
table_to_html
table_to_text

Untuk informasi selengkapnya, bacalah bagian pengembangan aplikasi GUI.

DATABASE

Singkong mendukung pengembangan aplikasi database (relasional) sederhana, dimana setiap koneksi ke sistem database adalah sebuah DATABASE.

Sebuah DATABASE dapat pula dibandingkan dengan DATABASE lain dengan operator == atau !=.

Fungsi bawaan terkait:

Kategori	Daftar fungsi
DATABASE	database, database_connected, database_metadata, query

Untuk bekerja dengan database relasional tanpa secara langsung menggunakan perintah-perintah SQL, gunakanlah modul bawaan db_util:

create_field_from_array
db_connect
db_connect_embed
db_connect_embed_
db_connect_embed_user
db_create_table
db_create_table_
db_create_table_derby
db_create_table_derby_
db_create_table_embed
db_create_table_embed_
db_create_table_postgresql
db_create_table_postgresql_
db_delete
db_delete_
db_driver
db_insert
db_insert_
db_last
db_last_derby
db_last_embed
db_last_postgresql

db_query_simple
db_query_single
db_run_query
db_select
db_select_
db_select_all
db_select_all_
db_select_derby
db_select_derby_
db_select_embed
db_select_embed_
db_select_postgresql
db_select_postgresql_
db_update
db_update_
query_result

Untuk informasi selengkapnya, bacalah bagian pengembangan aplikasi database.

3. Daftar Fungsi dan Modul Built-in

Fungsi built-in atau fungsi bawaan menyediakan sejumlah fungsionalitas, baik yang mendasar seperti bekerja dengan berbagai tipe data, alat bantu untuk bekerja dengan sistem, dan fungsi lanjutan seperti bekerja dengan GUI atau database.

Beberapa fungsi lanjutan bekerja lintas tipe data, seperti halnya fungsi set atau random. Namun, sejumlah besar fungsi built-in hanya bekerja dengan tipe data tunggal. Di bab sebelumnya, kita telah melihat fungsi-fungsi terkait untuk masing-masing tipe data.

Fungsi bawaan telah dirancang agar tidak terlalu ketat, namun juga diusahakan agar jangan sampai berpeluang menyebabkan kerepotan gara-gara terlalu longgar. Sebagai contoh, beberapa fungsi yang menerima argumen bertipe STRING akan menerima argumen bertipe apa saja, dan kemudian menggunakan representasi STRING dari nilai yang dilewatkan. Sejumlah fungsi menerima argumen wajib dan opsional. Beberapa fungsi cukup ketat seperti sama sekali menolak untuk dipanggil apabila kondisi tertentu tidak terpenuhi (misal GUI tidak tersedia). Lalu, beberapa fungsi akan menyediakan fungsionalitas yang lebih sederhana ketika GUI tidak tersedia.

Pengembangan lanjutan interpreter Singkong tidak terhindarkan akan menambah daftar fungsi built-in yang disediakan. Pada saat buku ini ditulis, terdapat 339 fungsi bawaan dan 6 modul bawaan yang akan kita lihat di dalam bab ini.

Untuk mendapatkan daftar fungsi bawaan, gunakanlah fungsi built-in builtins. Sebuah ARRAY berisikan nama fungsi akan dihasilkan.

Di dalam bab ini, kita hanya akan melihat nama fungsi, deskripsi singkat, dan contoh sederhana. Untuk informasi lebih lanjut seperti jumlah argumen wajib dan opsional, serta cara kerja yang lebih rinci apabila ada, ketikkanlah nama fungsi tanpa memanggilnya (tanpa (dan)) di interactive evaluator. Sebagai alternatif, fungsi built-in help dapat juga digunakan.

Contoh yang lebih rumit untuk penggunaan fungsi akan dibahas pada bagian tersendiri, misal untuk pengembangan aplikasi GUI, aplikasi database, atau ketika bekerja dengan method Java. Contoh penggunaan built-in lainnya dapat juga dilihat pada berbagai contoh kode program Singkong.

Sejak Singkong versi 4.6, disertakan juga modul bawaan (di bundel bersama file interpreter Singkong.jar). Untuk mendapatkan daftar modul bawaan, gunakanlah fungsi bawaan modules. Modul bawaan datang dengan sejumlah fungsionalitas, akan tetapi:

- Perlu di-load dengan fungsi bawaan load_module.
- Nama fungsi yang disertakan dalam modul bawaan bisa digunakan ulang sebagai identifier.

Pembahasan singkat tentang modul bawaan dilakukan setelah daftar fungsi bawaan berikut.

Fungsi	Deskripsi Singkat	Contoh
_E	e (base natural logarithm)	
_PI	pi	

Fungsi	Deskripsi Singkat	Contoh
ABS	Nilai absolut untuk NUMBER atau path absolut untuk file	abs(-1.23)
ACOS	Trigonometri	
ADD	Menambahkan COMPONENT atau ARRAY COMPONENT ke region center dari Frame	
ADD_E	Menambahkan COMPONENT atau ARRAY COMPONENT ke region east dari Frame	
ADD_N	Menambahkan COMPONENT atau ARRAY COMPONENT ke region north dari Frame	
ADD_S	Menambahkan COMPONENT atau ARRAY COMPONENT ke region south dari Frame	

Fungsi	Deskripsi Singkat	Contoh
ADD_W	Menambahkan COMPONENT atau ARRAY COMPONENT ke region west dari Frame	
APPEND	Menambahkan konten ke dalam file teks	
ARGUMENTS	Mendapatkan command line argument sebagai ARRAY STRING	
ARRAY	Konversi HASH atau STRING ke ARRAY	array("Singkong")
ARRAY_EQUALS	Membandingkan sebuah ARRAY dengan ARRAY lain, dari sisi elemen dan urutan elemen. Operator == pada ARRAY tidak membandingkan urutan elemen dan kedua ARRAY dianggap sama apabila mengandung elemen yang sama (walau dalam urutan berbeda).	
ARRAY_EXTEND	Menambahkan elemen ARRAY lain ke sebuah ARRAY	array_extend([1,2], [3,4])

Fungsi	Deskripsi Singkat	Contoh
ARRAY_EXTEND_ALL	Menggabungkan semua elemen dalam ARRAY ke sebuah ARRAY baru	
ARRAY_NUMBER	Mendapatkan ARRAY dengan semua element NUMBER	
ARRAY_STRING	Mendapatkan ARRAY dengan semua elemen adalah representasi STRING dari elemen berbagai tipe	
ASIN	Trigonometri	
ATAN	Trigonometri	
AVERAGE	Rata-rata elemen NUMBER dalam ARRAY	average([1,2,3,4,5])
BASE64_DECODE	Decode Base64	
BASE64_ENCODE	Encode Base64	base64_encode("Singkong")
BEEP	Membunyikan beep	
BOOLEAN_XOR	Boolean XOR	
BUILTINS	Mendapatkan semua fungsi built-in yang tersedia	
BUTTON_IMAGE	Mengatur icon-icon untuk button	

Fungsi	Deskripsi Singkat	Contoh
CALL	Memanggil method Java (sesuai aturan modul Singkong)	call(“Dialog", "Singkong")
CALL_INFO	Mendapatkan informasi method Java yang tersedia (sesuai aturan modul Singkong)	
CBRT	Cube root	
CENTER	Rata tengah STRING	
CGI_CONTENTS	Mencetak representasi STRING setiap item dalam ARRAY ke standard output (fungsi bantu untuk konten dalam CGI)	
CGI_GET	Mendapatkan QUERY_STRING (x-www-form-urlencoded) dari request HTTP GET, sebagai HASH (key/value telah di-decode)	
CGI_HEADER	Mencetak header-header CGI ke standard output. Tanpa argumen: Content-Type: text/html	

Fungsi	Deskripsi Singkat	Contoh
CGI_POST	Mendapatkan body (x-www-form-urlencoded) dari request HTTP POST, sebagai HASH (key/value telah di-decode)	
CGI_POST_HASH	Mendapatkan body (x-www-form-urlencoded, STRING dari HASH) dari request HTTP POST, sebagai HASH (key/value telah di-decode)	
CHR	Mendapatkan STRING (satu karakter) dari NUMBER (ordinal)	
CLEAR	Menghapus semua COMPONENT dari Fram	
CLIPBOARD_GET	Mendapatkan isi clipboard	
CLIPBOARD_SET	Mengeset isi clipboard	
CLOSING	Mengaktifkan atau menonaktifkan konfirmasi menutup Frame	closing("Apakah yakin ingin keluar dari program?", "Konfirmasi")
COLOR_CHOOSER	Menampilkan dialog pemilihan warna, dapat diberikan title	

Fungsi	Deskripsi Singkat	Contoh
COMPONENT	Membuat komponen GUI	
COMPONENT_INFO	Mendapatkan x (screen), y (screen), x, y, lebar, tinggi, nama, read-only COMPONENT	
COMPONENT_TYPE	Mendapatkan tipe sebuah COMPONENT	
COMPONENTS	Mendapatkan daftar semua tipe komponen GUI yang didukung	
CONFIG	Konfigurasi komponen GUI	
CONFIRM	Menampilkan dialog konfirmasi	
COPY	Mengopi file	
COPY_RESOURCE	Membaca /resource/<file> yang berada dalam file interpreter, dan menuliskan isinya ke sebuah file	
COS	Trigonometri	
COSH	Hyperbolic	
COUNT	Mendapatkan jumlah elemen tertentu dalam ARRAY	count([1,2,3], 1)

Fungsi	Deskripsi Singkat	Contoh
CR	Carriage Return	
CRLF	Carriage Return diikuti dengan Line Feed	
CWD	Mendapatkan direktori kerja aktif	
DATABASE	Membuat koneksi DATABASE	
DATABASE_CONNECTED	Mendapatkan informasi apakah sebuah DATABASE terkoneksi atau tidak	
DATABASE_METADATA	Mendapatkan metadata sebuah DATABASE	
DATE	Konversi STRING atau ARRAY komponen waktu ke DATE	date([2020, 1, 1])
DATETIME	Konversi STRING atau ARRAY komponen waktu dengan jam ke DATE	datetime("2020-01-01 00:00:00")
DAY	Menambahkan atau mengurangi hari dari DATE	day(@, 10)
DAYS_OF_MONTH	Mendapatkan ARRAY tanggal dan day of week dari suatu bulan	

Fungsi	Deskripsi Singkat	Contoh
DELAY	Sleep untuk sejumlah milidetik	
DELETE	Menghapus file atau direktori kosong	
DIFF	Mendapatkan perbedaan antara dua DATE	diff(@2020, @2021, 5)
DIR	Mendapatkan semua nama file dalam sebuah direktori	
DIRECTORY	Menampilkan dialog untuk memilih direktori	
DISABLED	Mendapatkan ARRAY fungsi built-in yang dinonaktifkan	
DO	Memanggil sebuah fungsi sejumlah kali	do(5, fn() {println("Singkong")})
DRAW_ARC	Menggambar sebuah arc	
DRAW_GET_PIXEL	Mendapatkan nilai pixel RGB pada posisi tertentu	
DRAW_LINE	Menggambar sebuah garis	
DRAW_OVAL	Menggambar sebuah oval	

Fungsi	Deskripsi Singkat	Contoh
DRAW_POLYGON	Menggambar sebuah polygon	
DRAW_POLYLINE	Menggambar sebuah polyline	
DRAW_READ	Membaca dari file image	
DRAW_RECT	Menggambar sebuah rectangle	
DRAW_ROUND_RECT	Menggambar sebuah round rectangle	
DRAW_SET_PIXEL	Mengeset nilai pixel RGB pada posisi tertentu	
DRAW_STRING	Menggambar sebuah STRING	
DRAW_WIDTH	Menentukan line width pada saat menggambar	
DRAW_WRITE_BMP	Menulis ke file image (bmp)	
DRAW_WRITE_JPG	Menulis ke file image (jpg)	
DRAW_WRITE_PNG	Menulis ke file image (png)	
EACH	Untuk setiap elemen dalam ARRAY, panggil sebuah fungsi sejumlah kali, dengan argumen	each([1,2,3], fn(x, y) {println(y + ": " + x)})

Fungsi	Deskripsi Singkat	Contoh
EMPTY	Apakah sebuah STRING, HASH, atau ARRAY adalah kosong	
ENDSWITH	Apakah STRING diakhiri dengan STRING tertentu	
ENV	Mendapatkan environment variable sistem. Berguna juga dalam CGI.	
EQUALS	Apakah dua STRING sama, tanpa membedakan huruf besar/kecil	
EVAL	Mengevaluasi STRING sebagai kode Singkong	eval(“var x = 1000; println(x)”) var x = eval(“1+2+3")
EVENT	Mendaftarkan event handler untuk komponen GUI	
EVENT_FOCUS	Mendaftarkan event handler ketika COMPONENT menerima atau kehilangan fokus	
EVENT_FRAME	Mendaftarkan event handler untuk frame	

Fungsi	Deskripsi Singkat	Contoh
EVENT_KEYBOARD_FRAME	Mendaftarkan event handler untuk keyboard, untuk frame	
EVENT_MOUSE	Mendaftarkan event handler untuk mouse dan mouse motion, untuk komponen GUI	
EVENT_MOUSE_FRAME	Mendaftarkan event handler untuk mouse dan mouse motion, untuk frame	
EXIT	Terminasi program Singkong	
EXP	natural exponential	
FILL_ARC	Menggambar dan mengisi sebuah arc dengan warna foreground	
FILL_OVAL	Menggambar dan mengisi sebuah oval dengan warna foreground	
FILL_POLYGON	Menggambar dan mengisi sebuah polygon dengan warna foreground	
FILL_RECT	Menggambar dan mengisi sebuah rectangle dengan warna foreground	

Fungsi	Deskripsi Singkat	Contoh
FILL_ROUND_RECT	Menggambar dan mengisi sebuah round rectangle dengan warna foreground	
FIRST	Mendapatkan elemen pertama dalam ARRAY	
FONTS	Mendapatkan daftar nama font yang tersedia	
FORMAT_DATE	Memformat DATE tanpa komponen waktu, dengan format opsional	format_date(@, "MMM DD, YYYY")
FORMAT_DATETIME	Memformat DATE dengan komponen tanggal dan waktu, dengan format opsional	
FORMAT_DIFF	Memformat perbedaan antara dua DATE	format_diff(1.5, " year ", " month ", " day ", 0)
FORMAT_TIME	Memformat DATE tanpa komponen tanggal, dengan format opsional	
FRAME	Mendapatkan properti Frame	
FRAME_CLOSE	Menutup Frame	

Fungsi	Deskripsi Singkat	Contoh
FRAME_LOCATION	Memindahkan Frame ke lokasi tertentu ataupun ke tengah layar	
FRAME_TOP	Mengubah always on top Frame	
FROM_BIN	Konversi dari biner	
FROM_HEX	Konversi dari heksadesimal	
FROM_OCT	Konversi dari oktal	
GET	Mendapatkan konfigurasi komponen GUI	
GRID_ADD	Menambahkan COMPONENT ke sebuah grid, dengan layout berbasis grid	
GRID_CLEAR	Menghapus semua COMPONENT dalam grid	
GRID_REMOVE	Menghapus sebuah COMPONENT dari grid	
GUI	Apakah GUI tersedia	
HASH	Mendapatkan hash code sebuah nilai	
HELP	Mendapatkan informasi sebuah fungsi built-in	

Fungsi	Deskripsi Singkat	Contoh
HIDE	Menyembunyikan Frame	
HOUR	Menambahkan atau mengurangi jam dari DATE	hour(@, 10)
HTTP_DELETE	Mengirimkan request HTTP DELETE	
HTTP_GET	Mengirimkan request HTTP GET	
HTTP_GET_FILE	Mengirimkan request HTTP GET, hasilnya disimpan di file (download)	
HTTP_HEAD	Mengirimkan request HTTP HEAD	
HTTP_POST	Mengirimkan request HTTP POST	
HTTP_POST_OVERRIDE	Mengirimkan request HTTP POST dengan X-HTTP-Method-Override	
HTTP_PUT	Mengirimkan request HTTP PUT	
HTTP_RESPONSE_OK	Mengembalikan data (bagian dari hasil request HTTP) hanya apabila status code adalah 200	

Fungsi	Deskripsi Singkat	Contoh
IN	Apakah ARRAY atau STRING mengandung elemen tertentu	in([1,2,3], 1)
INDEX	Mendapatkan indeks elemen tertentu dalam ARRAY atau STRING	index([1,2,3,1,2,3], 1)
INET_ADDRESS	Mendapatkan semua alamat IP dari mesin yang digunakan	
INET_ADDRESS_LOCAL	Mendapatkan alamat IP local host	
INFO	Mendapatkan informasi sistem	info()["os.name"]
INPUT	Meminta input	
INTEGER	Konversi NUMBER ke nilai bilangan bulat (integer) saja	
INTEGER_GCD	Mendapatkan greatest common divisor	
INTERACTIVE	Apakah kode Singkong dijalankan dalam interactive evaluator/editor	
IS	Apakah tipe ekspresi, variabel, dan nilai merupakan tipe tertentu	is([], "ARRAY")

Fungsi	Deskripsi Singkat	Contoh
IS_ARRAY_AND_OF	Memeriksa apakah merupakan ARRAY dan setiap elemen di dalamnya merupakan tipe tertentu	
IS_ARRAY_OF	Memeriksa apakah setiap elemen dalam sebuah ARRAY merupakan tipe tertentu	
IS_ARRAY_OF_COMPONENT_OF	Memeriksa apakah semua elemen dalam sebuah ARRAY dari COMPONENT merupakan tipe komponen GUI tertentu	
IS_LEAP_YEAR	Apakah suatu tahun merupakan tahun kabisat	
IS_RECT_ARRAY	Memeriksa apakah sebuah ARRAY merupakan rectangular array	
IS_RECT_ARRAY_OF	Memeriksa apakah sebuah ARRAY merupakan rectangular array, dan setiap elemen merupakan tipe tertentu	

Fungsi	Deskripsi Singkat	Contoh
IS_UPDATE_AVAILABLE	Memeriksa apakah terdapat versi baru interpreter Singkong	
ISALNUM	Apakah karakter dalam STRING adalah alphanumeric	
ISALPHA	Apakah karakter dalam STRING adalah alphabetic	
ISDIGIT	Apakah karakter dalam STRING adalah digit	
ISLOWER	Apakah karakter dalam STRING adalah huruf kecil	
ISUPPER	Apakah karakter dalam STRING adalah huruf besar	
JOIN	Menggabungkan semua elemen dalam ARRAY, dipisahkan STRING pemisah tertentu	join(", ", [1,2,3])
KEYS	Mendapatkan semua key dalam pemetaan	keys({1:2, 3:4})
KEYWORDS	Mendapatkan semua keyword dalam bahasa pemrograman Singkong	

Fungsi	Deskripsi Singkat	Contoh
LAST	Mendapatkan elemen terakhir dalam ARRAY	
LEFT	Rata kiri STRING	left("Singkong", 10, "=")
LEFT_SHIFT	Left shift	
LEN	Mendapatkan panjang STRING, HASH, atau ARRAY	
LF	Line Feed	
LOAD	Menjalankan file program Singkong lain	load("test.singkong")
LOAD_FILE_OR_RESOURCE	Menjalankan file program Singkong lain, yang tersimpan di direktori aktif atau (jika gagal dibaca), /resource/<value>, di file interpreter	
LOAD_MODULE	Menjalankan modul bawaan (ditulis dengan Singkong)	load_module("csv")
LOAD_RESOURCE	Menjalankan file program Singkong lain, yang tersimpan sebagai /resource/<value>, di file interpreter	
LOG	natural logarithm	
LOG10	base 10 logarithm	

Fungsi	Deskripsi Singkat	Contoh
LOGIN_DIALOG	Menampilkan dialog login sederhana	
LOWER	Konversi STRING ke huruf kecil	
MATCHES	Apakah suatu STRING sesuai dengan pola regular expression tertentu	
MAX	Mendapatkan nilai maksimum dalam ARRAY NUMBER	
MD5	MD5	
MD5_FILE	MD5 (file)	
MENUBAR	Mengatur menu-menu yang ditampilkan pada menu bar	
MESSAGE	Menampilkan dialog pesan	
MIN	Mendapatkan nilai minimum dalam ARRAY NUMBER	
MINUTE	Menambahkan atau mengurangi menit dari DATE	
MKDIR	Membuat direktori baru	
MODULES	Mendapatkan daftar modul bawaan	

Fungsi	Deskripsi Singkat	Contoh
MONTH	Menambahkan atau mengurangi bulan dari DATE	
NEWLINE	Mendapatkan karakter newline dari sistem berjalan	
NUMBER	Konversi STRING atau DATE ke NUMBER	
NUMBER_AND	Bitwise AND	
NUMBER_BOOLEAN	Mengembalikan 1 jika true, 0 jika false	
NUMBER_GROUP	Mengelompokkan angka NUMBER berdasarkan pemisah ribuan dan desimal tertentu. Pattern opsional dapat diberikan.	number_group(12345.6789, ".", “,") number_group(12345.6789, ".", ",", "#,##0.#")
NUMBER_NOT	Bitwise NOT	
NUMBER_OR	Bitwise OR	
NUMBER_SCALE	Mengatur scale / jumlah digit (1 sampai 16) setelah koma untuk NUMBER	
NUMBER_TO_BYTE	Konversi NUMBER ke NUMBER (byte)	
NUMBER_TO_BYTE_ARRAY	Konversi NUMBER ke ARRAY NUMBER (byte)	

Fungsi	Deskripsi Singkat	Contoh
NUMBER_XOR	Bitwise XOR	
OPEN	Menampilkan dialog untuk membuka file	
ORD	Mendapatkan NUMBER (integer ordinal) dari STRING (karakter pertama)	
PANEL_ADD	Menambahkan COMPONENT ke panel, dengan posisi absolut	
PANEL_CLEAR	Menghapus semua COMPONENT dalam panel	
PANEL_DIALOG	Membuat custom dialog berbasis panel	
PANEL_REMOVE	Menghapus sebuah COMPONENT dari panel	
PARAM	Mendapatkan parameter FUNCTION atau parameter wajib BUILTIN	
PARSE_HASH	Konversi STRING ke HASH	

Fungsi	Deskripsi Singkat	Contoh
PART	Mendapatkan ARRAY komponen waktu dari DATE dalam [year, month, day, hour, minute, second]	part(@)
PARTS	Mendapatkan komponen waktu dari DATE sebagai STRING yyyyMMddHHmmss	parts(@)
PASSWORD	Meminta input berupa password	
PLAY_SOUND	Memainkan suara, dari file ataupun resource	
POP	Mendapatkan semua elemen dalam ARRAY kecuali elemen terakhir	
POPUP_COMPONENT	Menampilkan popup window pada COMPONENT	
POPUP_HIDE	Menutup popup window	
POPUP_SHOW	Menampilkan popup window pada posisi tertentu	
PRINT	Menulis ke standard output, tanpa diikuti newline	

Fungsi	Deskripsi Singkat	Contoh
PRINTER	Menampilkan dialog pencetakan ke printer untuk ARRAY STRING	
PRINTLN	Menulis ke standard output, dengan diikuti newline	
PROPERTIES_READ	Membaca file properties sebagai HASH	
PROPERTIES_WRITE	Menulis HASH ke file properties	
PUSH	Menambahkan elemen ke dalam ARRAY	
PUTS	Menulis ke standard output, dengan diikuti newline	
QUERY	Menjalankan satu atau lebih SQL query dalam transaksi	
QUOTE	Karakter spesial kutip ganda	
RADIO_GROUP	Membuat mutual-exclusion untuk sejumlah radio button	

Fungsi	Deskripsi Singkat	Contoh
RANDOM	Mendapatkan NUMBER acak (antara 0 dan 1 eksklusif), NUMBER acak dalam batasan tertentu (inklusif), elemen acak dari ARRAY, key acak dari HASH	random(1,100) random([1,2,3]) random({1:2, 3:4})
RANDOM_STRING	Mendapatkan STRING acak dengan panjang tertentu	random_string(4, 8)
RANGE	Mendapatkan ARRAY NUMBER mulai dari bilangan tertentu sampai bilangan tertentu lain, dengan step opsional	range(1, 10, 2) range(10, 1, -2)
READ	Mendapatkan isi sebuah file teks	
READ_BYTE	Mendapatkan isi sebuah file sebagai ARRAY NUMBER (byte)	
RECT_ARRAY_SIZE	Mendapatkan ukuran (baris, kolom) sebuah rectangular array	

Fungsi	Deskripsi Singkat	Contoh
RECT_ARRAY_SIZE_OF	Mendapatkan ukuran (baris, kolom) sebuah rectangular array, dimana setiap elemen merupakan tipe tertentu	
REMOVE	Menghapus semua COMPONENT dari region center Frame	
REMOVE_E	Menghapus semua COMPONENT dari region east Frame	
REMOVE_N	Menghapus semua COMPONENT dari region north Frame	
REMOVE_S	Menghapus semua COMPONENT dari region south Frame	
REMOVE_W	Menghapus semua COMPONENT dari region west Frame	
RENAME	Mengubah nama atau memindahkan sebuah file atau direktori	
REPLACE	Mengganti setiap substring dari STRING tertentu dengan STRING lain	

Fungsi	Deskripsi Singkat	Contoh
REQUIRE	Menentukan bahwa program membutuhkan versi minimum tertentu dari interpreter Singkong, yang apabila tidak terpenuhi, akan menyebabkan terminasi program (dengan pesan kesalahan tertentu)	require(3.0) require (3.0, "Program membutuhkan Singkong 3.0 atau yang lebih baru”)
RESET	Menghapus semua COMPONENT dari Frame, mengubah kembali title dan ukuran ke nilai default, menghentikan semua timer yang ada, mendisable konfirmasi penutupan, mereset status bar dan menu bar, mereset always on top, menutup semua popup.	
RESIZABLE	Mengatur agar frame dapat diubah ukurannya atau tidak	
REST	Mendapatkan semua elemen dalam ARRAY dari indeks 1	

Fungsi	Deskripsi Singkat	Contoh
REVERSE	Membalik ARRAY	reverse(sort_number([3, 1, 2]))
RIGHT	Rata kanan STRING	
RIGHT_SHIFT	Right shift	
ROUND	Membulatkan NUMBER	
RUNTIME_VERSION	Mendapatkan versi runtime	
SAVE	Menampilkan dialog untuk menyimpan file	
SCREEN	Mendapatkan ukuran layar	
SECOND	Menambahkan atau mengurangi detik dari DATE	
SEPARATOR	Pemisah direktori dan file di sistem berjalan	
SET	Mengubah STRING, ARRAY, atau HASH	
SHA1	SHA1	
SHA1_FILE	SHA1 (file)	
SHA256	SHA256	
SHA256_FILE	SHA256 (file)	
SHA384	SHA384	
SHA384_FILE	SHA384 (file)	

Fungsi	Deskripsi Singkat	Contoh
SHA512	SHA512	
SHA512_FILE	SHA512 (file)	
SHOW	Menampilkan Frame	
SHUFFLE	Mengacak urutan elemen dalam ARRAY	
SIN	Trigonometri	
SINGKONG	Mendapatkan informasi Singkong	singkong()["version"]
SINGKONG_INTERPRETER	Mendapatkan nama file dan path absolut interpreter Singkong	
SINH	Hyperbolic	
SIZE	Mengatur ukuran Frame	
SLICE	Mendapatkan irisan atau bagian tertentu dari STRING atau ARRAY	slice("Singkong", 0, 4) slice([1,2,3,4,5], 0, 4)
SORT_ARRAY	Mengurutkan ARRAY yang semua elemen di dalamnya adalah ARRAY	sort_array([[1,2], [], [1,2,3]])
SORT_BOOLEAN	Mengurutkan ARRAY yang semua elemen di dalamnya adalah BOOLEAN	

Fungsi	Deskripsi Singkat	Contoh
SORT_DATE	Mengurutkan ARRAY yang semua elemen di dalamnya adalah DATE	
SORT_HASH	Mengurutkan ARRAY yang semua elemen di dalamnya adalah HASH	
SORT_NUMBER	Mengurutkan ARRAY yang semua elemen di dalamnya adalah NUMBER	
SORT_STRING	Mengurutkan ARRAY yang semua elemen di dalamnya adalah STRING	
SPLIT	Memecah STRING berdasarkan STRING pemisah tertentu	split("Hello, World", ", ")
SQRT	Square root	
START	Menjalankan semua timer	
START_TIMER	Menjalankan timer tertentu	
STARTSWITH	Apakah STRING diawali dengan STRING tertentu	
STAT	Mendapatkan informasi sebuah file	

Fungsi	Deskripsi Singkat	Contoh
STATUSBAR	Mengatur teks pada bagian tertentu dari status bar, dan apakah teks tersebut enabled atau tidak	
STDIN	Mendapatkan STRING (dari standard input)	
STOP	Menghentikan semua timer	
STOP_TIMER	Menghentikan timer tertentu	
STRING	Mendapatkan representasi STRING dari sebuah nilai	
STRING_FROM_BYTE_ARRAY	Konversi ARRAY NUMBER (byte) ke STRING	
STRING_TO_BYTE_ARRAY	Konversi STRING ke ARRAY NUMBER (byte)	
SUM	Menjumlahkan semua elemen NUMBER di dalam ARRAY	
SYSTEM	Menjalankan perintah sistem dan mendapatkan outputnya	system(["java", "-version"])

Fungsi	Deskripsi Singkat	Contoh
TAB	Karakter spesial tab	
TAB_ADD	Menambahkan sebuah komponen (panel) ke tab	
TAB_CLEAR	Menghapus semua komponen dari tab	
TAB_REMOVE	Menghapus sebuah komponen (panel) dari tab	
TABLE_ADD	Menambahkan baris ke dalam komponen GUI table	
TABLE_BOTTOM	Set alignment kolom tabel (bawah)	
TABLE_CENTER	Set alignment kolom tabel (tengah, horizontal)	
TABLE_COLUMN	Mengatur lebar kolom dalam tabel. Lebar kolom diberikan sebagai ARRAY, dengan jumlah elemen sama dengan jumlah kolom, atau berupa ARRAY kosong (dimana lebar kolom akan disesuaikan secara proporsional).	

Fungsi	Deskripsi Singkat	Contoh
TABLE_COLUMN_COUNT	Mendapatkan jumlah kolom dalam tabel	
TABLE_COLUMN_INFO	Mendapatkan informasi kolom dalam tabel (nama, alignment horizontal, alignment vertikal)	
TABLE_COLUMN_NAME	Mendapatkan nama kolom dalam tabel	
TABLE_GET_VALUE	Mendapatkan nilai pada baris dan kolom tertentu dalam tabel	
TABLE_LEFT	Set alignment kolom tabel (kiri)	
TABLE_MIDDLE	Set alignment kolom tabel (tengah, vertikal)	
TABLE_PRINT	Mencetak tabel	
TABLE_REMOVE	Menghapus sebuah baris dari komponen GUI table	
TABLE_RIGHT	Set alignment kolom tabel (kanan)	
TABLE_ROW_COUNT	Mendapatkan jumlah baris dalam tabel	
TABLE_SCROLL	Scroll table ke baris tertentu	

Fungsi	Deskripsi Singkat	Contoh
TABLE_SET_VALUE	Mengeset nilai pada baris dan kolom tertentu dalam tabel	
TABLE_TOP	Set alignment kolom tabel (atas)	
TAN	Trigonometri	
TANH	Hyperbolic	
TEMP_FILE	Membuat file temporary	
THREAD	Membuat dan menjalankan thread baru, yang secara opsional dapat disertai dengan intrinsic lock	
THREAD_ALIVE	Mengetahui apakah suatu thread masih berjalan	
THREAD_JOIN	Menunggu sebuah thread selesai dijalankan	
TIMER	Memanggil fungsi setiap jeda waktu tertentu (milidetik)	
TIMER_RUNNING	Mengetahui apakah timer tertentu sedang berjalan	
TITLE	Mengubah title Frame	
TO_BIN	Konversi ke biner	

Fungsi	Deskripsi Singkat	Contoh
TO_DEGREES	Konversi radian derajat	
TO_HEX	Konversi ke heksadesimal	
TO_OCT	Konversi ke oktal	
TO_RADIANS	Konversi derajat radian	
TRIM	Mendapatkan STRING dengan whitespace di awal dan akhir STRING dihapus	
TYPE	Mendapatkan tipe ekspresi, variabel, atau nilai	type(null) type([1,2,3])
TYPES	Mendapatkan semua tipe yang didukung dalam Bahasa Pemrograman Singkong	
UNSIGNED_RIGHT_SHIFT	Unsigned right shift	
UPPER	Konversi STRING ke huruf besar	
URL_DECODE	Decode dari STRING application/x-www-form-urlencoded	
URL_ENCODE	Encode ke STRING application/x-www-form-urlencoded	

Fungsi	Deskripsi Singkat	Contoh
USER	Mendapatkan nama user	
USERHOME	Mendapatkan home directory user	
VALUES	Mendapatkan semua value dalam pemetaan	values({1:2, 3:4})
VARIABLES	Mendapatkan daftar variabel global	
WAIT	Menampilkan informasi tunggu (label) dan memanggil FUNCTION atau BUILTIN di thread lain. Sebagai alternatif, kita juga dapat membuat sendiri thread-thread yang diperlukan.	
WORDS_EN	Terbilang dalam Bahasa Inggris	words_en("123.45")
WORDS_ID	Terbilang dalam Bahasa Indonesia	words_id("123.45")
WRITE	Menulis ke file, menghapus isi sebelumnya apabila ada	

Fungsi	Deskripsi Singkat	Contoh
WRITE_BYTE	Menulis ke file, data dari ARRAY NUMBER (byte), menghapus isi sebelumnya apabila ada	
X_BASE64_DECODE_FILE	Fungsi ini membutuhkan versi minimum Java 8. Decode STRING dengan Base64 dan simpan ke file.	
X_BASE64_ENCODE_FILE	Fungsi ini membutuhkan versi minimum Java 8. Encode file dengan Base64.	
X_EDIT_PRINT	Fungsi ini membutuhkan versi minimum Java 6. Mencetak isi COMPONENT edit.	
YEAR	Menambahkan atau mengurangi tahun dari DATE	

Di halaman-halaman berikut adalah daftar modul bawaan, yang dibundel dalam file interpreter Singkong.jar (direktori / singkong). Modul-modul tersebut ditulis sepenuhnya dengan Bahasa Singkong.

Modul	Deskripsi	Daftar Fungsi
csv	Bekerja dengan file Comma-Separated Values	CSV_FROM_STRING CSV_FROM_STRING_DEFAULT CSV_FUNCTIONS CSV_TO_STRING CSV_TO_STRING_DEFAULT
db_util	Bekerja dengan database relasional tanpa perintah SQL secara langsung, Fungsi tambahan lain	CREATE_FIELD_FROM_ARRAY DB_CONNECT DB_CONNECT_EMBED DB_CONNECT_EMBED_ DB_CONNECT_EMBED_USER DB_CREATE_TABLE DB_CREATE_TABLE_ DB_CREATE_TABLE_DERBY DB_CREATE_TABLE_DERBY_ DB_CREATE_TABLE_EMBED DB_CREATE_TABLE_EMBED_ DB_CREATE_TABLE_POSTGRESQL DB_CREATE_TABLE_POSTGRESQL_ DB_DELETE DB_DELETE_ DB_DRIVER DB_INSERT DB_INSERT_ DB_LAST DB_LAST_DERBY DB_LAST_EMBED DB_LAST_POSTGRESQL DB_QUERY_SIMPLE DB_QUERY_SINGLE DB_RUN_QUERY DB_SELECT DB_SELECT_ DB_SELECT_ALL DB_SELECT_ALL_ DB_SELECT_DERBY DB_SELECT_DERBY_ DB_SELECT_EMBED DB_SELECT_EMBED_ DB_SELECT_POSTGRESQL DB_SELECT_POSTGRESQL_ DB_UPDATE DB_UPDATE_ QUERY_RESULT

Modul	Deskripsi	Daftar Fungsi
rect_array_util	Fungsi tambahan, untuk bekerja dengan rectangular ARRAY dan matriks	ADD_RECT_ARRAY_OF ADD_RECT_ARRAY_OF_NUMBER CROSS_PRODUCT_3D_RECT_ARRAY_COLUMN_OF CROSS_PRODUCT_3D_RECT_ARRAY_COLUMN_OF_NUMBER CROSS_PRODUCT_3D_RECT_ARRAY_ROW_OF CROSS_PRODUCT_3D_RECT_ARRAY_ROW_OF_NUMBER DETERMINANT_RECT_ARRAY_OF DETERMINANT_RECT_ARRAY_OF_NUMBER INVERSE_RECT_ARRAY_OF INVERSE_RECT_ARRAY_OF_NUMBER IS_CONSTANT_RECT_ARRAY_OF IS_CONSTANT_RECT_ARRAY_OF_NUMBER IS_DIAGONAL_RECT_ARRAY_OF IS_DIAGONAL_RECT_ARRAY_OF_NUMBER IS_IDENTITY_RECT_ARRAY_OF_NUMBER IS_LOWER_TRIANGULAR_RECT_ARRAY_OF IS_LOWER_TRIANGULAR_RECT_ARRAY_OF_NUMBER IS_RECT_ARRAY_COLUMN_OF IS_RECT_ARRAY_COLUMN_OF_NUMBER IS_RECT_ARRAY_ROW_OF IS_RECT_ARRAY_ROW_OF_NUMBER IS_SQUARE_RECT_ARRAY_OF IS_SQUARE_RECT_ARRAY_OF_NUMBER IS_SYMMETRIC_RECT_ARRAY_OF IS_SYMMETRIC_RECT_ARRAY_OF_NUMBER IS_UPPER_TRIANGULAR_RECT_ARRAY_OF IS_UPPER_TRIANGULAR_RECT_ARRAY_OF_NUMBER IS_ZERO_RECT_ARRAY_OF IS_ZERO_RECT_ARRAY_OF_NUMBER MINOR_RECT_ARRAY_OF MINOR_RECT_ARRAY_OF_NUMBER MUL_RECT_ARRAY_OF MUL_RECT_ARRAY_OF_NUMBER MUL_SCALAR_RECT_ARRAY_OF MUL_SCALAR_RECT_ARRAY_OF_NUMBER RECT_ARRAY_NOT_ROW_COL SUB_RECT_ARRAY_OF SUB_RECT_ARRAY_OF_NUMBER TRACE_RECT_ARRAY_OF TRACE_RECT_ARRAY_OF_NUMBER TRANSPOSE_RECT_ARRAY_OF TRANSPOSE_RECT_ARRAY_OF_NUMBER

Modul	Deskripsi	Daftar Fungsi
set_util	Fungsi tambahan, untuk bekerja dengan set	CREATE_RELATION_FROM_ARRAY CREATE_SET_FROM_ARRAY GET_RELATION_PART INVERSE_BIJECTIVE_FUNCTION_SET IS_ANTISYMMETRIC_RELATION_SET IS_BIJECTIVE_FUNCTION_SET IS_FUNCTION_SET IS_INJECTIVE_FUNCTION_SET IS_REFLEXIVE_RELATION_SET IS_RELATION IS_RELATION_SET IS_SAME_SET IS_SUB_SET IS_SURJECTIVE_FUNCTION_SET IS_SYMMETRIC_RELATION_SET IS_TRANSITIVE_RELATION_SET SET_DIFF SET_INTERSECTION SET_POWER SET_PRODUCT SET_UNION
ui_util	Fungsi tambahan, untuk bekerja dengan user interface	TABLE_ADD_FILL TABLE_ADD_ROW_FILL TABLE_FILL TABLE_GET_ARRAY_ TABLE_GET_ARRAY_NUMBER TABLE_GET_ARRAY_STRING TABLE_TO_HTML TABLE_TO_TEXT

Modul	Deskripsi	Daftar Fungsi
util	Fungsi tambahan	ARRAY_COPY ARRAY_DIFF BINOMIAL_COEFFICIENT BINOMIAL_DISTRIBUTION FACTORIAL GEOMETRIC_DISTRIBUTION_FAILURE GEOMETRIC_DISTRIBUTION_SUCCESS HASH_COPY MEAN MEDIAN MODE NUMBER_GROUP_C_P NUMBER_GROUP_P_C NUMBER_GROUP_S_C NUMBER_GROUP_S_P POISSON_DISTRIBUTION RANGE_ SIMPLE_FILE_DECRYPT SIMPLE_FILE_ENCRYPT SIMPLE_STRING_DECRYPT SIMPLE_STRING_ENCRYPT SORT_RECT_ARRAY_OF_NUMBER_BY_INDEX STANDARD_DEVIATION STANDARD_DEVIATION_ STANDARD_DEVIATION_SAMPLE VARIANCE VARIANCE_ VARIANCE_SAMPLE _NON_EMPTY_ARRAY_OF_NUMBER

Berikut adalah contoh untuk menjalankan/load module:

```
> load_module(“csv”)

> load_module("db_util")

> load_module(“rect_array_util")

> load_module(“set_util")

> load_module("ui_util")
```

```
>  load_module(“util")
```

Contoh penggunaan ulang nama fungsi dalam modul bawaan:

```
>  var len = "hello, world"

ERROR: [line: 1] len is a built-in function

>  load_module("csv")

>  csv_from_string

[documentation string] [csv] function:
csv_from_string: returns ARRAY from CSV STRING
(with separator).

arguments: 2: STRING (separator) and STRING (CSV)

return value: ARRAY or NULL (error)

>  var csv_from_string = "hello, world"

>  csv_from_string

"hello, world”
```

Halaman ini sengaja dikosongkan

4. Percabangan dan Perulangan

Singkong mendukung seleksi/kondisi if dan perulangan repeat, dalam bentuk yang sederhana. Kita dapat memanfaatkan HASH untuk seleksi/kondisi apabila memungkinkan. Untuk perulangan, fungsi built-in seperti do dan each mungkin dapat digunakan apabila perulangan yang sederhana ingin dilakukan.

If

Sintaks dari if adalah sebagai berikut:

```
if (condition) {consequences} else
{alternatives}
```

Dimana:

- condition: ekspresi yang dapat dievaluasi menjadi true atau false. Gunakanlah operator & dan |, serta pengelompokan dnegan (dan) apabila diperlukan.
- consequences: blok ini akan dikerjakan apabila condition bernilai true
- else dan blok alternatives adalah opsional, dan apabila disediakan, maka akan dikerjakan apabila condition bernilai false

Contoh 1: kondisi true/false sederhana

```
var x = 1
if (x > 0) {
  println("x > 0")
}
```

Contoh 2: penggunaan else

```
var x = 0
if (x > 0) {
      println("x > 0")
} else {
      println("x <= 0")
}
```

Contoh 3: penggunaan operator &

```
var a = [1,2,3]
if (is(a, "array") & len(a) > 0) {
      println("array dengan isi")
} else {
      println("bukan array dengan isi")
}


var a = []
if (is(a, "array") & len(a) > 0) {
      println("array dengan isi")
} else {
      println("bukan array dengan isi")
}
```

Contoh 4: penggunaan (dan)

```
var x = 1
if ((x == 1) | (x == 2)) {
      println("x=1 atau x=2")
} else {
      println("x!=1 atau x!=2")
}
```

Contoh 5: if di dalam if

```
var x = 1
if ((x == 1) | (x == 2)) {
      if (x == 1) {
            println("x=1")
      } else {
            println("x=2")
      }
} else {
      println("x!=1 atau x!=2")
}
```

Contoh 6:

```
var t = thread(fn() {delay(500)})
thread_join(t)
if (thread_alive(t)) {
  println("busy")
} else {
  println("idle")
}
```

Penggunaan HASH

Ada kalanya, kita ingin melakukan tindakan tertentu apabila kondisi terpenuhi, namun kondisi yang perlu diperiksa ada banyak. Dengan demikian, penggunaan if di dalam if tidak lagi nyaman. Kita bisa memanfaatkan HASH seperti contoh berikut:

```
var actions = {
      1: fn(){
            println("1")
      },
      2: fn() {
```

```
            println("2")
      },
      3: fn() {
            println("3")
      }
}

var x = 1
if (in(keys(actions), x)) {
      actions[x]()
} else {
      println("tidak ada yang terdaftar")
}
```

Di dalam contoh tersebut, kita hanya mendaftarkan fungsi sederhana untuk setiap kondisi yang ingin terpenuhi. Tentu saja, variasi yang lebih kompleks dimungkinkan, karena key dan value untuk HASH dapat merupakan tipe apapun di Singkong.

Repeat

Sintaks dari repeat adalah sebagai berikut:

```
repeat { statements }
```

Dimana:

- Apabila tidak terdapat statement apapun, seperti repeat{}, maka perulangan tidak akan dikerjakan sama sekali.
- Namun, apabila kita memberikan satu statement saja, seperti repeat {null}, maka perulangan akan dikerjakan tanpa henti. Program dapat dihentikan dengan mekanisme interupsi program di sistem operasi yang Anda gunakan (misal dengan kombinasi control-c untuk shell yang mendukung).

- Untuk keluar dari perulangan, gunakanlah kata kunci return, dengan nilai kembalian eksplit. Perulangan repeat di Singkong dapat mengembalikan nilai, sama halnya dengan fungsi.

Contoh 1: perulangan sederhana sebanyak 5 kali

```
var x = 0
repeat {
      println(x)
      var x = x+1
      if (x > 4) {
            return x
      }
}
```

Contoh 2: nilai kembalian perulangan

```
var x = 0
var r = repeat {
      println(x)
      var x = x+1
      if (x > 4) {
            return "OK"
      }
}

println(r)
```

Fungsi Built-in: do

Ada kalanya, kita hanya ingin melakukan tindakan tertentu beberapa kali. Fungsi built-in do dapat digunakan untuk

kebutuhan tersebut. Anda tidak perlu melakukan perulangan dengan repeat dan return apabila kondisi tertentu terpenuhi.

Contoh 1: memanggil fungsi selama 5 kali

```
do(5, fn() {
      println("Singkong")
})
```

Contoh 2: melewatkan argumen ketika memanggil fungsi

```
var f = fn(x, y, z) {
      println(x + " " + y + " " + z)
}

do(5, f, "Singkong", "Programming",
"Language")
```

Fungsi Built-in: each

Fungsi each sangat berguna apabila kita ingin melakukan perulangan untuk setiap elemen dalam ARRAY. Keunggulan fungsi each adalah bahwa indeks (dimulai dari 0) secara otomatis akan dilewatkan bersama elemen tersebut. Namun, fungsi yang dipanggil harus menerima dua parameter, yaitu elemen dan indeks.

Contoh 1: perulangan sederhana

```
var x = ["Singkong", "Programming",
"Language"]
each(x, fn(e, index) {
      println(e)
})
```

Contoh 2: menggunakan nilai indeks yang dilewatkan

```
var x = [1,2,3,4,5]
each(x, fn(e, index) {
      println(index + ": " + e)
})
```

Contoh 3: memroses hanya elemen tertentu berdasarkan indeks

```
var x = [1,2,3,4,5]
each(x, fn(e, index) {
      if (index % 2 == 0) {
            println(index + ": " + e)
      }
})
```

Halaman ini sengaja dikosongkan

5. Pengembangan Aplikasi GUI

Salah satu tujuan dari bahasa pemrograman Singkong adalah menyediakan cara pengembangan aplikasi Graphical User Interface yang sederhana, semudah dan seringkas mungkin.

Sederhana dalam hal ini adalah Singkong membatasi jumlah komponen user interface yang didukung, dengan aturan tertentu.

Mudah dapat diartikan programmer tidak perlu memahami cara kerja GUI secara mendetil. Dan ringkas dalam hal ini adalah dengan sesedikit mungkin baris kode.

Sebagai gambaran, berikut adalah contoh program editor file teks, yang dapat membuka file, melakukan pengubahan, dan menyimpannya kembali. Total hanya dalam sekitar 30 baris kode program.

```
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()
```

Frame dan Dialog

Program GUI yang ditulis dengan kode Singkong hanya dapat bekerja dengan satu Frame per program. Frame dalam hal ini merupakan top level window.

Untuk bekerja dengan berbagai dialog yang umum digunakan, gunakanlah fungsi-fungsi built-in berikut: confirm, directory, input, open, message, password, save. Untuk menampilkan dialog login sederhana, fungsi built-in login_dialog dapat digunakan.

Untuk membuat dialog sendiri yang berbasiskan pada panel atau grid, gunakanlah fungsi built-in panel_dialog. Dengan demikian, untuk dialog yang tidak disediakan oleh Singkong, kita dapat membuatnya sendiri (cukup dengan kode Singkong). Untuk dialog yang lebih kompleks, kita dapat menggunakan kode Java.

Untuk menampilkan popup, baik di dalam ataupun luar frame, ataupun pada komponen tertentu, fungsi-fungsi berikut dapat digunakan: popup_show, popup_component, popup_hide. Isi dari popup sendiri dapat berupa panel ataupun grid, sebagaimana halnya dialog. Bedanya, popup tidak memiliki dekorasi window. Contoh popup adalah datepicker.

Fungsi built-in terkait Frame. Bacalah juga dokumentasi fungsi terkait apabila diperlukan.

Fungsi	Contoh
title	title("Hello, World")
size	size(400, 300)
frame	frame()
frame_close	frame_close()
reset	reset()
resizable	resizable(false)
closing	closing("Apakah yakin ingin keluar dari program?", "Konfirmasi")
frame_top	frame_top(false)
frame_location	frame_location() frame_location(100, 200)

Gunakanlah fungsi built-in closing sebagaimana dicontohkan dalam tabel sebelumnya untuk mengaktifkan/menonaktifkan (dengan STRING kosong) konfirmasi sebelum menutup Frame. Secara default, Frame akan langsung ditutup ketika user menutup Frame. Tidak ada konfirmasi yang akan dilakukan.

Untuk mengatur agar frame always on top (ataupun tidak, nilai default), gunakanlah fungsi frame_top.

Untuk mendapatkan properti frame (width, height, title, resizable, always on top, x, y), fungsi built-in frame dapat digunakan.

Untuk memindahkan frame ke lokasi tertentu ataupun ke tengah layar (posisi default adalah di tengah layar), gunakanlah fungsi frame_location.

Komponen GUI

Berikut adalah daftar komponen GUI yang didukung pada saat buku ini ditulis, sebagaimana juga bisa didapatkan dengan memanggil fungsi built-in components.

```
>  components()
["barchart", "button", "checkbox", "combobox",
"date",  "draw",  "edit",  "grid",  "image",
"label",    "mask",    "panel",    "password",
"piechart",   "progress",   "radio",   "spin",
"tab", "table", "text", "view"]
```

Komponen	Deskripsi
barchart	Bar chart
button	Tombol
checkbox	Check box
combobox	Combo box, tidak dapat diedit
date	Memilih tanggal/waktu, format dapat ditentukan
draw	Digunakan untuk menggambar dengan berbagai fungsi yang telah disediakan
edit	Editor teks lebih dari satu baris, otomatis dilengkapi dengan scroll bar. Dapat diatur agar tidak dapat diedit.
grid	Grid yang dapat digunakan untuk menampung berbagai komponen, dengan layout berbasis grid
image	Image, dapat digunakan untuk menampilkan gambar dari file

Komponen	Deskripsi
label	Label, dapat digunakan untuk menampilkan teks
mask	Input teks dengan karakter mask. Dapat diatur agar tidak dapat diedit.
panel	Panel yang dapat digunakan untuk menampung berbagai komponen, dengan posisi absolut
password	Input teks berupa password. Dapat diatur agar tidak dapat diedit.
piechart	Pie chart
progress	Progress bar
radio	Radio button, dapat berdiri sendiri ataupun merupakan mutual-exclusion set
spin	Memilih bilangan bulat dari rentang nilai dan langkah tertentu
tab	Tabbed panel, yang memungkinkan panel-panel ditambahkan menjadi tab-tab tersendiri, dengan label setiap tab adalah nama panel.
table	Tabel, dapat digunakan untuk menampilkan data tabular. Dapat diatur agar tidak dapat diedit.
text	Input teks. Dapat diatur agar tidak dapat diedit.
view	Menampilkan HTML

Untuk membuat komponen user interface, gunakanlah fungsi built-in component. Fungsi ini menerima dua argumen wajib, yang keduanya bertipe STRING. Argumen pertama adalah

tipe komponen (salah satu dari komponen di tabel sebelumnya, tidak dibedakan huruf besar/kecil) dan argumen kedua adalah nama komponen. Argumen opsional yang dapat diterima fungsi component adalah bertipe BOOLEAN, yang apabila diberikan nilai true, maka komponen tersebut tidak dapat diedit, sebagai telah dideskripsikan dalam tabel. Apabila tidak terjadi kesalahan, fungsi ini mengembalikan COMPONENT.

Interpretasi nama komponen dapat dilihat pada tabel berikut:

Komponen	Interpretasi nama komponen
barchart	
button	Label
checkbox	Label
combobox	Item dalam combobox, dipisahkan koma
date	Format tanggal/waktu
draw	Lebar (integer; default 100), tinggi (integer; default 100)
edit	Teks
grid	Dapat digunakan sebagai label tab, ketika grid ditambahkan ke tab
image	Nama file
label	Label
mask	Format (#: bilangan, U: huruf besar, L: huruf kecil, A: karakter atau bilangan, ?: karakter, *: apa saja, H: karakter heksa; Informasi lebih lanjut: javax.swing.text.MaskFormatter)

Komponen	Interpretasi nama komponen
panel	Dapat digunakan sebagai label tab, ketika panel ditambahkan ke tab
password	Teks
piechart	
progress	
radio	Label
spin	nilai (integer), minimum (integer), maksimum (integer), langkah (integer) Apabila terdapat nilai yang invalid, maka nilai-nilai default akan digunakan
tab	
table	Kolom tabel, dipisahkan koma
text	Teks
view	Kode HTML

Khusus untuk komponen date:

- Apabila name tidak diberikan, maka format default adalah yyyy-MM-dd.
- Apabila input waktu diperlukan, maka contoh format yang dapat digunakan adalah yyyy-MM-dd HH:mm:ss.
- Jika nama hari dan bulan diinginkan, gunakanlah contoh format EEE, yyyy-MMM-dd atau EEEE, yyyy-MMMM-dd. Untuk informasi selengkapnya, bacalah juga dokumentasi Java tentang SimpleDateFormat.
- Konfigurasi komponen date:
  - DATE atau representasi STRING dari DATE dapat digunakan.
  - Apabila representasi STRING digunakan, gunakanlah fungsi built-in parts seperti contoh berikut:

```
var d = component(“date","")
config(d, "contents", parts(@))
```

- Apabila DATE digunakan, konversi tidak perlu dilakukan.

```
var d = component(“date","")
config(d, "contents", @)
```

- Ketika mendapatkan konfigurasi komponen date, sebuah DATE akan dikembalikan

Tambahan untuk komponen text:
Untuk mengatur jumlah kolom dalam komponen text tersebut, lakukanlah konfigurasi dengan key active. Pengaturan ini berlaku apabila komponen ditempatkan dalam grid. Sebagai contoh:

```
reset()
var g = component("grid", "")
var t1 = component("text", "Text 1")
config(t1, "active", 10)
var t2 = component("text", "Text 2")
grid_add(g, t1, 0, 0, 1, 1, 1, 1, 0, 0)
grid_add(g, t2, 1, 0, 1, 1, 1, 1, 0, 0)
add(g)
show()
```

Untuk komponen edit, text, dan view, menu popup (dan shortcut) beserta beberapa perintah untuk teks, termasuk undo/redo didukung sejak Singkong versi 5.1. Menu popup dapat diaktifkan dengan klik kanan ataupun control click, sesuai yang disediakan oleh sistem operasi.

Berikut adalah contoh pembuatan komponen user interface:

```
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)

add([e, 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()
```

Untuk mendapatkan tipe komponen user interface, gunakanlah fungsi built-in component_type, seperti contoh berikut. Fungsi ini mengembalikan tipe komponen dalam STRING, salah satu dari komponen dalam tabel sebelumnya.

```
var b = component("button", "Hello");
component_type(b)
```

Menambahkan/Menghapus Komponen

Untuk menambahkan komponen ke dalam Frame, gunakanlah salah satu dari fungsi built-in berikut: add, add_e, add_n, add_s, add_w (yang akan menambahkan sebuah COMPONENT atau sebuah ARRAY COMPONENT ke dalam region center, east, north, south, west).

Untuk menghapus komponen dari Frame, gunakanlah salah satu dari fungsi built-in berikut: remove, remove_e, remove_n, remove_s, remove_w (yang akan menghapus semua COMPONENT dari region center, east, north, south, west). Semua fungsi tersebut tidak menerima argumen.

Untuk menghapus semua komponen dari Frame, gunakanlah fungsi built-in clear.

Untuk menghapus semua komponen dari Frame, mengubah kembali title dan ukuran ke nilai default, menghentikan semua timer yang ada, mendisable konfirmasi penutupan, mereset status bar dan menu bar, mereset always on top, serta menutup semua popup, gunakanlah fungsi reset. Fungsi ini

berguna ketika beberapa program dengan GUI dijalankan lewat interactive evaluator/editor (yang mana semua program tersebut menggunakan Frame yang sama).

Pengaturan region pada Frame adalah sebagai berikut:

<table>
<tr><td colspan="3">north</td></tr>
<tr><td>west</td><td>center</td><td>east</td></tr>
<tr><td colspan="3">south</td></tr>
</table>

Setiap region dapat ditambahkan nol atau lebih komponen user interface, dimana region center umumnya berisikan komponen yang diutamakan (dapat berupa panel atau grid). Ukuran komponen akan dihitung agar mengisi semua ruang yang tersedia, sesuai dengan ukuran Frame.

Contoh:

```
reset()
var b1 = component("button", "Button 1")
var b2 = component("button", "Button 2")
var b3 = component("button", "Button 3")
add([b1, b2])
add_s(b3)
show()
```

Penambahan komponen yang disebutkan sebelumnya tidak dilakukan dengan posisi absolut. Kita tidak pernah meminta untuk menempatkan sebuah tombol, sebagai contoh, pada posisi x dan y tertentu, dengan ukuran lebar dan tinggi tertentu. Semua dikalkulasi secara otomatis.

Namun, ada kalanya terdapat kebutuhan untuk menempatkan banyak komponen dengan posisi dan ukuran yang ingin

bebas ditentukan. Hal ini dimungkinkan dengan komponen panel. Kita menambahkan semua komponen tersebut ke dalam sebuah panel, dan barulah panel tersebut yang ditambahkan ke frame. Atau, panel tersebut nantinya dapat ditambahkan ke grid atau panel lain.

Gunakanlah fungsi-fungsi built-in berikut ketika bekerja dengan panel:

- panel_add: untuk menambahkan sebuah komponen ke panel
- panel_remove: untuk menghapus sebuah komponen dalam panel
- panel_clear: untuk menghapus semua komponen dalam panel

Ketika bekerja dengan posisi absolut, dan ukuran frame diubah, ukuran komponen-komponen yang ditambahkan tersebut tidaklah menyesuaikan. Anda mungkin ingin mengatur agar frame tidak dapat diubah ukurannya dengan fungsi resizable. Atau, Anda mungkin ingin menggunakan grid, yang memungkinkan layout komponen yang kompleks, yang ukuran komponen akan menyesuaikan ketika frame diubah ukurannya.

Contoh penambahan komponen ke panel:

```
reset()
var p = component("panel", "")
var b1 = component("button", "Button 1")
var b2 = component("button", "Button 2")
panel_add(p, b1, 10, 10, 200, 50)
panel_add(p, b2, 10, 70, 200, 30)
add(p)
show()
```

Bagaimana kalau kita ingin menambahkan komponen-komponen dengan layout yang kompleks, namun ukuran komponen-komponen tersebut perlu menyesuaikan secara otomatis ketika ukuran frame diubah? Tidak seperti pada panel, kita tidak ingin menentukan posisi komponen secara

absolut. Dan, kita ingin memberikan bobot tertentu pada komponen-komponen yang ditambahkan (komponen mana yang harus lebih besar ketika ada ruang kosong pada kontainer). Lebih lanjut lagi, kita juga ingin menentukan apakah komponen akan mengisi ruang kosong ketika kontainer diubah ukurannya (apabila ya, apakah secara horizontal, vertikal, atau keduanya). Dan untuk membuat layout lebih rapi, kita ingin dapat mengatur padding antar komponen.

Jawabannya adalah dengan komponen grid, yang disediakan sejak Singkong versi 5.3. Grid tersebut nantinya dapat ditambahkan ke frame, panel, atau grid lain.

Gunakanlah fungsi-fungsi built-in berikut ketika bekerja dengan grid:
- grid_add: untuk menambahkan sebuah komponen ke grid
- grid_remove: untuk menghapus sebuah komponen dalam grid
- grid_clear: untuk menghapus semua komponen dalam grid

Contoh:

```
reset()
var g = component("grid", "")
var b1 = component("button", "Button 1")
var b2 = component("button", "Button 2")
var b3 = component("button", "Button 3")
var b4 = component("button", "Button 4")
var b5 = component("button", "Button 5")
grid_add(g, b1, 0, 0, 1, 1, 0.5, 1.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b2, 1, 0, 1, 1, 0.5, 1.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b3, 0, 1, 2, 1, 1.0, 0.5, 3, 0, 5, 5, 5, 5)
grid_add(g, b4, 0, 2, 1, 1, 0.5, 0.5, 0, 1)
grid_add(g, b5, 1, 2, 1, 1, 0.5, 0.5, 0, 2)
add(g)
show()
```

Berikut adalah argumen fungsi grid_add (10):
COMPONENT (grid), COMPONENT, NUMBER (gridx), NUMBER (gridy), NUMBER (gridwidth), NUMBER (gridheight), NUMBER (weightx), NUMBER (weighty), NUMBER (fill;

0=NONE default, 1=HORIZONTAL, 2=VERTICAL, 3=BOTH), NUMBER (anchor; 0=CENTER default, 1=LINE_START, 2=LINE_END)

Sementara, argumen opsionalnya adalah (4): NUMBER (padding top), NUMBER (padding left), NUMBER (padding bottom), NUMBER (padding right)

Apabila jumlah komponen yang akan ditampilkan cukup banyak, kita bisa mengelompokkan komponen-komponen tersebut dalam tab. Sebelumnya, komponen-komponen tersebut perlu ditempatkan dalam sejumlah panel atau grid, dan panel-panel atau grid-grid tersebut yang akan ditambahkan ke dalam tab. Nama panel atau grid akan digunakan sebagai label pada tab. Gunakanlah fungsi-fungsi built-in berikut ketika bekerja dengan tab:

- tab_add: untuk menambahkan sebuah panel ke tab
- tab_remove: untuk menghapus sebuah panel dalam tab
- tab_clear: untuk menghapus semua panel dalam tab

Berikut adalah contoh kode untuk bekerja dengan tab, menggunakan panel dan grid:

```
reset()
var e = component("edit", "")
var c = component("checkbox", "Singkong?")
var p = component("panel", "Panel")
panel_add(p, e, 10, 10, 200, 50)
panel_add(p, c, 10, 70, 100, 30)

var g = component("grid", "Grid")
var b1 = component("button", "Button 1")
var b2 = component("button", "Button 2")
grid_add(g, b1, 0, 0, 1, 1, 0.5, 1.0, 3, 0, 5, 5, 5, 5)
grid_add(g, b2, 1, 0, 1, 1, 0.5, 1.0, 3, 0, 5, 5, 5, 5)

var t = component("tab", "")
tab_add(t, p)
tab_add(t, g)

add(t)
show()
```

Setelah komponen ditampilkan, apabila diperlukan informasi x (screen), y (screen), x, y, lebar, tinggi, nama, dan read-only dari komponen tersebut, gunakanlah fungsi built-in component_info.

Berikut adalah contoh lain penempatan komponen user interface:

```
reset()
var c = component("panel", "")
var g = component("grid", "")
var e = component("button", "E")
var n = component("button", "N")
var w = component("button", "W")

var s1 = component("button", "S 1")
var s2 = component("button", "S 2")
var s3 = component("button", "S 3")
var s = [s1, s2, s3]

add([c, g])
add_e(e)
add_n(n)
add_s(s)
add_w(w)

# panel (absolute positioning);
var items = range(0, 4)
each(items, fn(i, counter) {
  var y = (20 + 80) * counter
  panel_add(c, component("button",
string(i+1)), 50, y, 80, 80)
})
```

```
# grid (grid layout);
var b1 = component("button", "Button 1")
var b2 = component("button", "Button 2")
var b3 = component("button", "Button 3")
var b4 = component("button", "Button 4")
var b5 = component("button", "Button 5")
grid_add(g, b1, 0, 0, 1, 1, 0.5, 1.0, 3, 0, 5,
5, 5, 5)
grid_add(g, b2, 1, 0, 1, 1, 0.5, 1.0, 3, 0, 5,
5, 5, 5)
grid_add(g, b3, 0, 1, 2, 1, 1.0, 0.5, 3, 0, 5,
5, 5, 5)
grid_add(g, b4, 0, 2, 1, 1, 0.5, 0.5, 0, 1, 5,
5, 5, 5)
grid_add(g, b5, 1, 2, 1, 1, 0.5, 0.5, 0, 2, 5,
5, 5, 5)

show()
```

Konfigurasi Komponen

Untuk mengkonfigur sebuah komponen user interface, misal mengubah teks atau isinya, gunakanlah fungsi built-in config. Fungsi ini menerima tiga argumen: COMPONENT, STRING (key, konfigurasi), dan tipe apa saja.

Apabila argumen keempat (opsional, BOOLEAN) fungsi built-in config diberikan nilai true (default adalah false) dan key adalah contents, dan tipe COMPONENT adalah salah satu dari edit/image/password/text/view, serta value diberikan sebagai STRING, maka konten akan didapatkan dari resource yang tersimpan dalam file interpreter (path: /resource/ <value>). Dengan demikian, pada deployment dengan membundel file aplikasi bersama interpreter Singkong, berbagai resource seperti file gambar dan lainnya juga bisa dibundel bersama, dan digunakan dengan cara seperti ini.

Contoh perbedaan yang tidak mengunakan resource terbundel dan yang menggunakan resource terbundel, pada image:

```
var c = component("image", “”)
```

file.png yang digunakan adalah yang ditemukan pada direktori aktif (terpisah dari file jar interpreter):

```
config(c, "contents", “file.png")
```

file.png yang digunakan adalah yang ditemukan terbundel bersama file jar interpreter, dengan path adalah /resource/ file.png (menjadi bagian dari file jar interpreter, lebih mudah untuk didistribusikan):

```
config(c, "contents", "file.png", true)
```

Untuk mendapatkan konfigurasi sebuah komponen user interface, gunakanlah fungsi built-in get. Fungsi ini menerima dua argumen: COMPONENT dan STRING (key), dan mengembalikan nilai yang sesuai (tipe apa saja).

Key atau konfigurasi yang tidak dapat diterapkan akan diabaikan. Tidak ada kesalahan yang akan terjadi.

Berikut adalah semua key yang didukung di Singkong, beserta komponen yang dapat menerima konfigurasi tersebut.

Perhatikanlah bahwa terdapat perbedaan antara config dan get, dalam hal komponen yang menerima konfigurasi, namun tidak ada kesalahan yang akan terjadi apabila fungsi-fungsi tersebut dipanggil dengan key yang tidak didukung. Tidak semua pengaturan lewat config bisa didapatkan kembali dengan get. Sebagai contoh adalah pengaturan untuk barchart atau piechart, atau secara umum pada komponen yang tidak dapat diubah oleh pengguna program.

Key	Tipe	Deskripsi	Komponen
border	STRING (judul)	Menampilkan border dengan judul	Semua komponen yang didukung
enabled	BOOLEAN	Enable/disable komponen	Semua komponen
visible	BOOLEAN	Visible atau tidak visible	Semua komponen
focus	BOOLEAN	Menjadi komponen yang difokuskan	Semua komponen
foreground	STRING (nama warna atau nilai RGB)	Warna foreground komponen	Semua komponen
background	STRING (nama warna atau nilai RGB)	Warna background komponen	Semua komponen
font	ARRAY [STRING (nama font), NUMBER (0=plain, 1=bold, 2=italic, 3=bold dan italic), NUMBER (ukuran)]	Font komponen	Semua komponen

Key	Tipe	Deskripsi	Komponen
text	STRING	label button/ checkbox/ label/radio, item terpilih untuk combobox, title barchart	button/ checkbox/ label/radio, combobox, dan barchart
active	BOOLEAN atau NUMBER	Terpilih atau tidak untuk checkbox/ radio (BOOLEAN), baris terpilih untuk combobox/ table (NUMBER), tab aktif untuk tab (NUMBER), indeks mnemonic untuk button (NUMBER), jumlah kolom untuk text (NUMBER, hanya berlaku pada grid).	checkbox/ radio, combobox/ tab/table, button

Key	Tipe	Deskripsi	Komponen
contents	STRING, ARRAY, ARRAY dari ARRAY, DATE, NUMBER	Isi komponen: edit/mask/ password/text/ view (STRING), nama file untuk image (STRING). Untuk combobox: ARRAY. Untuk table: ARRAY dari ARRAY. Untuk barchart/ piechart: ARRAY dari ARRAY (dari [NUMBER (value), STRING (label), STRING (nama warna name atau nilai RGB)]. Untuk date: DATE/ STRING. Untuk progress/spin: NUMBER.	barchart/ combobox/ date/edit/ password/ piechart/ progress/spin/ table/text/view dan image
margin	ARRAY dari NUMBER	margin: ARRAY dari NUMBER [top, left, bottom, right], hanya berlaku pada grid.	button

Contoh penggunaan config dan get dapat juga dilihat pada contoh kode editor file di awal bab ini, contoh komponen, atau contoh dalam pembahasan event dan timer.

Terdapat beberapa fungsi built-in yang bekerja khusus untuk komponen user interface tertentu, sebagaimana berikut:

Komponen	Fungsi	Deskripsi	Argumen
table	table_add	Menambahkan baris ke tabel	COMPONENT dan ARRAY (dari ARRAY)
table	table_remove	Menghapus baris tertentu dari tabel, dengan indeks mulai dari 0	COMPONENT dan NUMBER
table	table_bottom	Alignment: bawah	COMPONENT dan NUMBER (index kolom)
table	table_center	Alignment: tengah (horizontal)	COMPONENT dan NUMBER (index kolom)
table	table_left	Alignment: kiri	COMPONENT dan NUMBER (index kolom)
table	table_middle	Alignment: tengah (vertikal)	COMPONENT dan NUMBER (index kolom)
table	table_right	Alignment: kanan	COMPONENT dan NUMBER (index kolom)

Komponen	Fungsi	Deskripsi	Argumen
table	table_scroll	Scroll ke baris tertentu, dengan nomor baris mulai dari 0	COMPONENT dan NUMBER
table	table_top	Alignment: atas	COMPONENT dan NUMBER (index kolom)
radio	radio_group	mutual-exclusion set untuk sejumlah radio button	ARRAY COMPONENT (radio)
button	button_image	mengatur icon-icon untuk button	COMPONENT (button), COMPONENT (image), COMPONENT (image, ketika tombol ditekan), COMPONENT (image, ketika tombol di-disable)

Event

SIngkong mendukung penanganan event default untuk beberapa komponen user interface.

Berikut adalah daftar komponennya:

Komponen	Deskripsi Event
button	Ketika tombol ditekan
combobox	Ketika item terpilih berubah
checkbox	Ketika dicek atau tidak dicek
date	Ketika isinya berubah
radio	Ketika dipilih atau tidak dipilih
tab	Ketika tab aktif berubah
table	Ketika baris aktif berubah
edit	Ketika isinya berubah
password	Ketika isinya berubah
spin	Ketika isinya berubah
text	Ketika isinya berubah

Untuk mendaftarkan fungsi yang akan otomatis dipanggil ketika event tersebut terjadi (event handler), gunakanlah fungsi built-in event. Fungsi ini menerima dua argumen: COMPONENT dan FUNCTION. Fungsi yang akan dipanggil tidak dapat menerima argumen.

Event: Mouse dan Mouse Motion

Untuk komponen, event untuk penggunaan dan pergerakan mouse dapat ditangani dengan fungsi, yang sebelumnya telah didaftarkan dengan fungsi built-in event_mouse.

Fungsi ini menerima dua argumen berikut: COMPONENT dan FUNCTION. Fungsi yang terdaftar akan dipanggil otomatis begitu event terkait mouse untuk komponen tersebut perlu ditangani.

Fungsi yang didaftarkan harus menerima sebuah argumen, dimana nantinya, informasi yang terkait event mouse akan dilewatkan pada saat pemanggilan fungsi dilakukan. Argumen tersebut bertipe ARRAY, dengan item-item berikut: [type (STRING: CLICKED, ENTERED,EXITED, PRESSED, RELEASED, MOVED, DRAGGED), x, y, jumlah klik dilakukan].

Berikut adalah contoh sederhana penanganan event mouse untuk sebuah button. Untuk mengujinya, operasikanlah mouse pada tombol tersebut.

```
    reset()
    var b = component("button", "mouse")
    var t = component("table", "TYPE, X, Y,
CLICK COUNT", true)

    var f = fn(x) {
      table_add(t, [x])
    }
    event_mouse(b, f)

    add(t)
    add_s(b)

    show()
```

Selain untuk komponen, event terkait mouse untuk frame juga dapat ditangani, dengan sebelumnya mendaftarkannya terlebih dahulu dengan fungsi built-in event_mouse_frame. Fungsi ini menerima sebuah argumen, yaitu FUNCTION. Perhatikanlah contoh berikut:

```
    reset()
    var t = component("table", "TYPE, X, Y,
CLICK COUNT", true)
```

```
    var f = fn(x) {
      table_add(t, [x])
    }
    event_mouse_frame(f)

    add_w(t)

    show()
```

Event: Keyboard

Untuk frame, event untuk penggunaan keyboard dapat ditangani dengan fungsi, yang sebelumnya telah didaftarkan dengan fungsi built-in event_keyboard_frame. Fungsi ini menerima sebuah argumen, yaitu FUNCTION.

Fungsi yang terdaftar akan dipanggil otomatis begitu event terkait keyboard perlu ditangani. Fungsi yang didaftarkan harus menerima sebuah argumen, dimana nantinya, informasi yang terkait event keyboard akan dilewatkan pada saat pemanggilan fungsi dilakukan. Argumen tersebut bertipe ARRAY, dengan item-item berikut: [type (STRING: TYPED, PRESSED, RELEASED), character (STRING, apabila dimungkinkan), kode (NUMBER), teks key, apakah merupakan action, teks modifier, lokasi key (STRING: STANDARD, LEFT, RIGHT, NUMPAD, UNKNOWN), apakah tombol alt ditekan, apakah tombol alt graph ditekan, apakah tombol control ditekan, apakah tombol meta ditekan, apakah tombol shift ditekan].

Perhatikanlah contoh berikut:

```
    reset()
    var t = component("table", "TYPE, CHAR,
CODE, KEY, ACTION, MODIFIER, LOCATION, ALT,
ALT GRAPH, CONTROL, META, SHIFT", true)

    var f = fn(x) {
```

```
    table_add(t, [x])
  }
  event_keyboard_frame(f)

  add(t)

  show()
```

Event: Frame

Untuk frame, event dapat ditangani dengan fungsi, yang sebelumnya telah didaftarkan dengan fungsi built-in event_frame. Fungsi ini menerima sebuah argumen, yaitu FUNCTION.

Fungsi yang terdaftar akan dipanggil otomatis begitu event frame perlu ditangani. Fungsi yang didaftarkan harus menerima sebuah argumen, dimana nantinya, informasi yang terkait event frame akan dilewatkan pada saat pemanggilan fungsi dilakukan. Argumen tersebut bertipe ARRAY, dengan item-item berikut: [type (STRING: ACTIVATED, CLOSED, DEACTIVATED, DEICONIFIED, GAINEDFOCUS, ICONIFIED, LOSTFOCUS, OPENED, STATECHANGED, RESIZED, MOVED), lebar, tinggi, x, y].

Bacalah juga dokumentasi fungsi built-in closing, apabila diperlukan.

Perhatikanlah contoh berikut:

```
    reset()
    var t = component("table", "TYPE, WIDTH,
HEIGHT, X, Y", true)

    var f = fn(x) {
        table_add(t, [x])
      }
```

```
    event_frame(f)

    add(t)

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

    show()
```

Berikut adalah contoh lain penanganan event di Singkong:

```
reset()
var b = component("button", "Hello World
(mouse event)")
var c = component("checkbox", "Singkong?")
var r = component("radio", "Radio Button")
var m = component("combobox",
"Singkong,Programming,Language")
var t = component("table", "A,B", true)
var tm = component("table", "TYPE, X, Y, CLICK
COUNT", true)
var tk = component("table", "TYPE, CHAR, CODE,
KEY, ACTION, MODIFIER, LOCATION, ALT, ALT
GRAPH, CONTROL, META, SHIFT", true)
var tf = component("table", "TYPE, WIDTH,
HEIGHT, X, Y", true)
var pm = component("panel", "Mouse")
var pk = component("panel", "Keyboard")
var pf = component("panel", "Frame")
var ta = component("tab", "")
var e = component("edit", "")
var p = component("password", "")
var x = component("text", "")
var v = component("view", "")
```

```
panel_add(pm, tm, 10, 10, 300, 500)
tab_add(ta, pm)
panel_add(pk, tk, 10, 10, 300, 500)
tab_add(ta, pk)
panel_add(pf, tf, 10, 10, 300, 500)
tab_add(ta, pf)

var bb = fn() {
    message(get(b,"text"))
}
event(b, bb)

var cc = fn() {
    message(get(c,"active"))
}
event(c, cc)

var rr = fn() {
    message(get(r,"active"))
}
event(r, rr)

var mm = fn() {
    message(get(m, "text"))
}
event(m, mm)

config(t, "contents", [[1,2],[3,4],[5,6]])
var tt = fn() {
    message(get(t, "contents")[get(t,
"active")])
}
event(t, tt)

var ee = fn() {
    var content = get(e, "contents")
    config(v, "contents", content)
```

```
}
event(e, ee)

var pp = fn() {
    message(get(p, "contents"))
}
event(p, pp)

var xx = fn() {
    message(get(x, "contents"))
}
event(x, xx)

event_mouse(b, fn(m) {
  table_add(tm, [m])
  if (m[0] == "EXITED") {
    var c = len(get(tm, "contents"))
    table_scroll(tm, c-1)
  }
})

event_keyboard_frame(fn(m) {
  table_add(tk, [m])
})

event_frame(fn(m) {
  table_add(tf, [m])
})

event(ta, fn() {
  message(get(ta, "active"))
})

add_n([p, x])
add_s([b, c, r, m])
add([t, e, v, ta])
```

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

Event: Focus

Untuk menangani ketika suatu komponen mendapatkan atau kehilangan fokus, gunakanlah fungsi built-in event_focus. Fungsi ini akan menerima sebuah argumen, yaitu FUNCTION.

Fungsi yang terdaftar akan dipanggil otomatis begitu event terkait fokus tersebut perlu ditangani. Fungsi yang didaftarkan harus menerima sebuah argumen, dimana nantinya, informasi yang terkait event fokus akan dilewatkan pada saat pemanggilan fungsi dilakukan. Argumen tersebut bertipe ARRAY, dengan item-item berikut: [type (STRING: GAINED, LOST), x, y].

Contoh:

```
reset()

var i = component("label", "Focus event: text")
var t = component("table", "TYPE, X, Y", true)
var x = component("text", "Click")

event_focus(x, fn(x) {
  table_add(t, [x])
})

add_n(i)
add(t)
add_s(x)

show()
```

Custom Dialog

Untuk pembuatan dialog yang belum disediakan oleh Singkong, dengan sepenuhnya hanya menggunakan kode Singkong, gunakanlah fungsi built-in panel_dialog.

Panel dialog di Singkong bekerja dengan panel atau grid. Sejumlah komponen yang ingin ditampilkan dalam dialog perlu ditempatkan ke panel atau grid terlebih dahulu, dan panel atau grid tersebut yang akan ditambahkan ke dialog, dengan title dan ukuran tertentu.

Fungsi panel_dialog menerima empat argumen: COMPONENT (panel atau grid), STRING (title), NUMBER (lebar), NUMBER (tinggi). Fungsi akan mengembalikan STRING, berupa STRING kosong, CANCEL, atau OK (sesuai tombol yang diklik).

Perhatikanlah contoh berikut, dimana panel digunakan:

```
reset()
var e = component("edit", "")
var c = component("checkbox", "Singkong?")
var p = component("panel", "")
panel_add(p, e, 10, 10, 200, 50)
panel_add(p, c, 10, 70, 100, 30)
var r = panel_dialog(p, "Form", 300, 200)
if (r == "OK") {
  message(get(e, "contents"))
  message(get(c, "active"))
} else {
  message("Cancel")
}
```

Sebagai alternatif, contoh berikut menggunakan grid:

```
    reset()
    var e = component("edit", "")
    var c = component("checkbox", "Singkong?")
    var g = component("grid", "")
    grid_add(g, e, 0, 0, 1, 1, 1, 1, 3, 0, 5,
5, 5, 5)
    grid_add(g, c, 0, 1, 1, 1, 0, 0, 3, 0, 5,
5, 5, 5)
    var r = panel_dialog(g, "Form", 300, 200)
    if (r == "OK") {
      message(get(e, "contents"))
      message(get(c, "active"))
    } else {
      message("Cancel")
    }
```

Popup

Popup berguna ketika diperlukan untuk menampilkan sebuah window tanpa dekorasi di posisi tertentu. Baik di luar frame ataupun pada komponen tertentu. Sebagai contoh adalah datepicker, dimana pengguna bisa memilih tanggal. Popup dapat ditutup setelah selesai.

Isi dari popup sendiri, sebagai halnya dialog, dapat menggunakan panel ataupun grid.

Fungsi-fungsi berikut dapat digunakan ketika bekerja dengan popup: popup_show, popup_component, dan popup_hide. Terkadang, penggunaan popup juga dikombinasikan dengan event tertentu. Sebagai contoh, popup ditampilkan pada komponen tertentu ketika komponen tersebut menerima fokus.

Contoh menampilkan popup pada posisi tertentu, dengan panel:

```
        reset()
```

```
        var t = component("text", "")
        var b = component("button", "OK")
        var p = component("panel", "")
        panel_add(p, t, 10, 10, 100, 30)
        panel_add(p, b, 10, 50, 80, 30)

        var r = [-1]

        var c = component("button", "popup")
        event(c, fn() {
            var x = popup_show(p, 120, 100, 50, 50,
false)
            set(r, 0, x)
        })
        add_s(c)
        show()
```

Contoh menampilkan popup pada posisi tertentu, dengan grid:

```
        reset()
        var t = component("text", "")
        var b = component("button", "OK")
        var g = component("grid", "")
        grid_add(g, t, 0, 0, 1, 1, 1, 1, 3, 0)
        grid_add(g, b, 0, 1, 1, 1, 1, 1, 1, 0)

        var r = [-1]

        var c = component("button", "popup")
        event(c, fn() {
            var x = popup_show(g, 120, 100, 50, 50,
false)
            set(r, 0, x)
        })
        add_s(c)
        show()
```

Contoh menampilkan popup pada posisi tertentu, dengan grid dan posisi relatif terhadap frame:

```
        reset()
```

```
        var t = component("text", "")
        var b = component("button", "OK")
        var g = component("grid", "")
        grid_add(g, t, 0, 0, 1, 1, 1, 1, 3, 0)
        grid_add(g, b, 0, 1, 1, 1, 1, 1, 1, 0)

        var r = [-1]

        var c = component("button", "popup")
        event(c, fn() {
            var x = popup_show(g, 120, 100, 50, 50,
true)
            set(r, 0, x)
        })
        add_s(c)
        show()
```

Contoh menampilkan popup pada komponen tertentu, dengan grid, ketika event fokus terjadi:

```
        reset()
             var t = component("text", "Focus on
combobox")
        var c = component("combobox", "")
        var g = component("grid", "")
        config(g, "background", "gray")
        var r = [-1]
        event_focus(c, fn(e) {
            if (e[0] == "GAINED") {
                var i = component_info(c)
                var x = popup_component(g, c, i[4],
300, 0, i[5])
                set(r, 0, x)
            }
        })

        add_n([t, c])
        show()
```

Contoh menutup popup dengan tombol yang berada dalam window fokus tersebut. Tombol tersebut sebelumnya ditambahkan dalam grid.

```
        reset()
        var t = component("text", "")
        var b = component("button", "Close")
        var g = component("grid", "")
        grid_add(g, t, 0, 0, 1, 1, 1, 1, 3, 0)
        grid_add(g, b, 0, 1, 1, 1, 1, 1, 1, 0)

        var r = [-1]

        var c = component("button", "popup")
        event(c, fn() {
            var x = popup_show(g, 120, 100, 50, 50,
true)
            set(r, 0, x)
        })
        add_s(c)
        show()

        event(b, fn() {
            popup_hide(r[0])
        })
```

Contoh popup yang tampil ketika komponen menerima fokus, dan tidak lagi tampil ketika komponen tersebut tidak lagi menerima fokus.

```
        reset()
             var t = component("text", "Focus on
combobox")
        var c = component("combobox", "")
        var g = component("grid", "")
        config(g, "background", "gray")
        var r = [-1]
        event_focus(c, fn(e) {
            if (e[0] == "GAINED") {
                var i = component_info(c)
```

```
                var x = popup_component(g, c, i[4],
300, 0, i[5])
                set(r, 0, x)
            } else {
                popup_hide(r[0])
            }
        })

        add_n([t, c])
        show()
```

Timer

Timer dapat digunakan untuk memanggil sebuah fungsi secara berkala setiap jeda waktu tertentu (dalam milidetik). Untuk mendaftarkan timer, gunakanlah fungsi built-in timer. Fungsi ini menerima dua argumen: NUMBER dan FUNCTION.

Untuk menghentikan semua timer, gunakanlah fungsi built-in stop. Untuk menghentikan timer tertentu, gunakanlah fungsi built-in stop_timer. Untuk menjalankan semua timer, gunakanlah fungsi built-in start. Untuk menjalankan timer tertentu, gunakanlah fungsi built-in start_timer. Untuk mengetahui apakah sebuah timer sedang berjalan, gunakanlah fungsi built-in timer_running.

Berikut adalah contoh penggunaan timer:

```
reset()
var l1 = component("label", string(@))
var l2 = component("label", string(@))
var b1 = component("button", "toggle timer 1")
var b2 = component("button", "toggle timer 2")
var b3 = component("button", "stop all
timers")
```

```
var b4 = component("button", "start all
timers")

add([l1, l2])
add_s([b1, b2, b3, b4])

var f1 = fn() {
    config(l1, "text", string(@))
}

var f2 = fn() {
    config(l2, "text", string(@))
}

var t1 = timer(1000, f1)
var t2 = timer(1000, f2)

event(b1, fn() {
    if (timer_running(t1)) {
        stop_timer(t1)
    } else {
        start_timer(t1)
    }
})

event(b2, fn() {
    if (timer_running(t2)) {
        stop_timer(t2)
    } else {
        start_timer(t2)
    }
})

event(b3, fn() {
    stop()
})
```

```
event(b4, fn() {
    start()
})

show()
```

Proses yang Berjalan Lama

Ada kalanya, sebuah proses mungkin membutuhkan waktu lama. Sebagai contoh, ketika memanggil HTTP API. Untuk itu, ada baiknya apabila pengguna program diberikan informasi bahwa sebuah proses sedang berjalan.

Proses yang berjalan tersebut merupakan pemanggilan fungsi, baik fungsi built-in (BUILTIN) ataupun fungsi yang dibuat sendiri (FUNCTION). Sementara, informasi ditampilkan sebagai sebuah label (COMPONENT). Dengan demikian, ukuran font dan atribut lain pada label bisa di-set.

Untuk menampilkan informasi dan memanggil fungsi, gunakanlah fungsi built-in wait. Fungsi ini menerima setidaknya dua argumen: COMPONENT (label) dan (FUNCTION atau BUILTIN). Apabila FUNCTION atau BUILTIN menerima argumen, deretkanlah pada pemanggilan fungsi wait.

Perhatikanlah bahwa pemanggilan fungsi dilakukan pada thread tersendiri. Bacalah juga bab bekerja dengan thread, apabila diperlukan.

Berikut adalah contoh penggunaan wait:

```
        reset()

        var l1 = component("label", "please
wait...")
```

```
        config(l1, "font", ["monospaced", 1,
20])

        var l2 = component("label", "busy...")
        config(l2, "font", ["monospaced", 1,
20])

        var b1 = component("button", "Delay")
        var b2 = component("button", "Busy")

        add_s([b1, b2])

        var busy = fn() {
                delay(2000)
        }

        event(b1, fn() {
                wait(l1, delay, 1000)
        })

        event(b2, fn() {
                wait(l2, busy)
        })

        show()
```

Sebagai alternatif, berikut adalah contoh simulasi proses yang berjalan lama, yang dilakukan pada thread tersendiri, namun disertai dengan update pada komponen user interface:

```
        var p = [0, "Connecting..."]

        var f = fn() {
            delay(random(1000, 1500))
            set(p, 0, random(30, 35))
            set(p, 1, "Downloading...")
            delay(random(2000, 2500))
```

```
    set(p, 0, random(70, 75))
    set(p, 1, "Processing...")
    delay(random(3000, 3500))
    set(p, 0, 100)
    set(p, 1, "Done")
    delay(1000)
}

var t = thread(f, p)

var s = number(@)

reset()
var x = component("progress", "")
add_s(x)

var c = fn() {
  if (thread_alive(t)) {
    var m = number(@) - s
    statusbar(0, m + " ms", true)
    statusbar(1, p[1], true)
    statusbar(2, p[0] + "%", true)
    config(x, "contents", p[0])
  } else {
    stop()
  }
}

timer(random(100, 500), c)
show()
```

Pencetakan ke Printer

Untuk mencetak ARRAY dari STRING ke printer, gunakanlah fungsi built-in printer, yang akan menampilkan dialog print.

Pengaturan sederhana untuk pencetakan seperti nama font, ukuran font, margin atas/kiri dapat dilakukan dengan fungsi printer tersebut.

Fungsi ini menerima empat argumen wajib dan sebuah argumen opsional. Argumen wajib adalah ARRAY (STRING yang ingin dicetak), NUMBER (ukuran font), NUMBER (margin sisi kiri), dan NUMBER (margin sisi atas). Argumen opsionalnya adalah STRING (nama font).

Untuk mencetak isi COMPONENT table ke printer, gunakanlah fungsi built-in table_print. Untuk mencetak isi COMPONENT edit ke printer, gunakanlah fungsi built-in x_edit_print.

Berikut adalah contoh pencetakan sederhana:

```
reset()

var header = component("text", "header")
var footer = component("text", "footer")

var b_print = component("button", "print")
var b_table_print = component("button", "print
table")
var b_edit_print = component("button", "print
edit")

var table = component("table", "Key, Value")
table_add(table, [ ["Singkong", "Programming
Language"] ])

var edit = component("edit", "Singkong
Programming Language")
```

```
add_n([header, footer])
add([table, edit])
add_s([b_print, b_table_print, b_edit_print])

event(b_print, fn() {
    var t = ["Singkong", "Programming",
"Language"]
    var s = 16
    var x = 150
    var y = 150
    var font = "monospaced"
    printer(t, s, x, y, font)
})

event(b_table_print, fn() {
    table_print(table, get(header,
"contents"), get(footer, "contents"))
})

event(b_edit_print, fn() {
    x_edit_print(edit, get(header,
"contents"), get(footer, "contents"))
})

show()
```

Ukuran Layar

Untuk mendapatkan ukuran layar, gunakanlah fungsi built-in screen. Fungsi ini akan mengembalikan sebuah ARRAY berisi lebar dan tinggi layar.

Font

Untuk mendapatkan semua nama font yang tersedia di sistem, gunakanlah fungsi built-in fonts.

Nama font tersebut dapat digunakan ketika ingin mengatur font komponen user interface dengan fungsi built-in config.

Clipboard

Untuk mengeset isi clipboard, gunakanlah fungsi built-in clipboard_set. Untuk mendapatkan isi clipboard, fungsi built-in clipboard_get dapat digunakan.

Suara

Untuk membunyikan beep, gunakanlah fungsi built-in beep. Untuk memainkan suara dari file ataupun resource, gunakanlah fungsi built-in play_sound.

Untuk resource, tambahkanlah semua file yang formatnya dikenal dalam direktori /resource dalam file jar interpreter. Sebagai contoh:

```
play_sound("sound.wav", true)
```

Fungsi tersebut akan memainkan /resource/sound.wav yang tersimpan dalam file jar interpreter. Bacalah juga pembahasan tentang deployment apabila diperlukan.

Untuk memainkan suara dari file:

```
play_sound("sound.wav", false)
```

Fungsi tersebut akan memainkan file sound.wav yang tersimpan di direktori aktif.

Status Bar

Untuk menampilkan informasi tertentu seperti status aplikasi, status bar dapat digunakan.

Frame di Singkong memiliki 8 bagian status bar, yang diwakili dengan indeks dari 0 sampai 7. Status bar ditempatkan di baris paling bawah dari frame.

Untuk mengatur teks dan apakah teks tersebut enabled atau tidak, untuk setiap bagian dari status bar, gunakanlah fungsi bawaan statusbar.

Contoh:

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

Menu Bar

Untuk mengatur menu-menu yang akan ditampilkan pada menu bar, gunakanlah fungsi bawaan menubar.

Fungsi ini membutuhkan sebuah ARRAY, dengan isi sebagai berikut:

```
[
        STRING (menu),
        NUMBER (mnemonic index),
                [] (untuk separator)
                atau
                [
                        STRING (item),
                        NUMBER (mnemonic index),
```

```
                    BOOLEAN (apakah enabled),
                    FUNCTION
            ]
]
```

Sebagai contoh, berikut adalah sebuah menu bar, yang terdiri dari menu File (Exit):

```
var menu = [
  ["File", 0,
    [
      ["Exit", 1, true, fn(){}]
    ]
  ]
]

reset()
menubar(menu)
show()
```

Contoh berikut adalah pengubahan dari menu bar sebelumnya, dimana Exit di set ke disabled:

```
var menu = [
  ["File", 0,
    [
      ["Exit", 1, false, fn(){}]
    ]
  ]
]

reset()
menubar(menu)
show()
```

Contoh berikut menampilkan menu File (Open, <separator>, Save (disabled)):

```
var menu = [
  ["File", 0,
    [
      ["Open", 0, true, fn(){}],
      [],
      ["Save", 0, false, fn(){}]
    ]
  ]
]

reset()
menubar(menu)
show()
```

Lebih lanjut, kita dapat menentukan apa yang akan dilakukan ketika sebuah menu item di klik, seperti contoh berikut:

```
var menu = [
  ["File", 0,
    [
      ["Open", 0, true, fn(){open()}],
      [],
      ["Save", 0, false, fn(){}]
    ]
  ]
]

reset()
menubar(menu)
show()
```

Dan, kita selalu bisa mengatur ulang menu bar secara dinamis, seperti contoh berikut:

```
var file_open = fn() {
  message("File Open")
}
```

```
var file_save = fn() {
  message("File Save")
}

var help_about = fn() {
  message("Help About")
}

var user_login = fn() {
  message("User Login")
}

var a = [
  ["File", 0,
    [
      ["Open", 0, true, file_open],
      [],
      ["Save", 1, false, file_save]
    ]
  ],
  ["Help", 0,
    [
      ["About", 2, true, help_about]
    ]
  ]
]

var b = [
  ["User", 0,
    [
      ["Login", 0, true, user_login]
    ]
  ]
]

reset()
```

```
var btn_a = component("button", "A")
var btn_b = component("button", "B")
add_s([btn_a, btn_b])

event(btn_a, fn() {
  menubar(a)
})

event(btn_b, fn() {
  menubar(b)
})

show()
```

Menggambar

Untuk menggambar pada komponen draw, fungsi-fungsi bawaan berikut dapat digunakan: draw_width, draw_rect, fill_rect, draw_arc, fill_arc, draw_oval, fill_oval, draw_round_rect, fill_round_rect, draw_string, draw_line, draw_polygon, fill_polygon, draw_polyline, draw_read, draw_write_png, draw_write_jpg, draw_write_bmp, draw_get_pixel, draw_set_pixel. Rujuklah ke dokumentasi fungsi untuk informasi selengkapnya.

Berikut adalah contoh penggunaan fungsi-fungsi tersebut.

```
reset()
var dr = component("draw", "500, 500")
add(dr)

config(dr, "foreground", "black")
config(dr, "background", "white")

draw_width(dr, 2)

draw_rect(dr, 10, 10, 100, 50)
fill_rect(dr, 120, 10, 100, 50)
```

```
draw_arc(dr, 10, 80, 100, 50, 90, 270)
fill_arc(dr, 120, 80, 100, 50, 90, 270)

draw_oval(dr, 10, 150, 100, 50)
fill_oval(dr, 120, 150, 100, 50)

draw_round_rect(dr, 10, 220, 100, 50, 30, 30)
fill_round_rect(dr, 120, 220, 100, 50, 30, 30)

config(dr, "font", ["monospaced", 1, 30])
draw_string(dr, "Singkong", 10, 320)

draw_line(dr, 10, 340, 240, 340)

draw_polygon(dr, [10, 50, 90], [400, 360, 400])
fill_polygon(dr, [120, 160, 200], [400, 360, 400])

draw_polyline(dr, [10, 50, 90], [480, 440, 480])

show()
```

Menggunakan modul ui_util

Untuk mendapatkan isi tabel sebagai ARRAY (melewati cell yang kosong), gunakanlah fungsi-fungsi berikut: table_get_array_, table_get_array_number, table_get_array_string.

Untuk konversi tabel ke HTML, gunakanlah fungsi table_to_html.

Berikut adalah contoh mengisi tabel dan konversi tabel ke teks, dengan fungsi-fungsi yang juga disediakan oleh modul ui_util.

```
load_module("ui_util")
reset()
var t = component("table", "A,B,C")
table_left(t, 0)
table_center(t, 1)
table_right(t, 2)
var e = component("edit", "")
```

```
config(e, "font", ["monospaced", 0, 10])
table_add_fill(t, 0)
var tt = table_to_text(t, [10, 20, 30])
config(e, "contents", tt)
add([t, e])
println(tt)
show()
```

Selain GUI, berikut adalah keluaran teks dari kode program tersebut:

```
============================================================
|   A    |        B         |             C              |
============================================================
|0       |        0         |                           0|
------------------------------------------------------------
|0       |        0         |                           0|
------------------------------------------------------------
|0       |        0         |                           0|
============================================================
```

Contoh: Program Painting Sederhana

Berikut adalah contoh program painting sederhana, yang memungkinkan pengguna menggambar dengan mouse, menambahkan teks, membaca/menulis dari/ke file image, mengubah nilai pixel (RGB), dan lainnya. Semuanya dalam kurang dari 200 baris.

```
reset()

var name = "Paint"
var width = 1024
var height = 768
var font_size = 20
size(width, height)
resizable(false)
title(name)
closing("Are you sure you want to quit this
application?", "Please confirm")

var g = component("grid", "")
var d = component("draw", "" + width + "," + height)
config(d, "background", "white")
```

```
var gn = component("grid", "")
var bf = component("button", "Foreground")
var bb = component("button", "Background")
var pf = component("panel", "")
var pb = component("panel", "")
var lx = component("label", "Pixel (+/-)")
var xr = component("text", "0")
var xg = component("text", "0")
var xb = component("text", "0")
var bx = component("button", "Set")
var br = component("button", "Open")
var bw = component("button", "Save (.png)")
var pp = component("panel", "")
var po = component("label", " ")
grid_add(gn, bf, 0, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gn, pf, 1, 0, 1, 1, 0.1, 0, 3, 1, 4, 4, 4, 4)
grid_add(gn, bb, 2, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gn, pb, 3, 0, 1, 1, 0.1, 0, 3, 1, 4, 4, 4, 4)
grid_add(gn, lx, 4, 0, 1, 1, 0.1, 0, 3, 1, 4, 4, 4, 4)
grid_add(gn, xr, 5, 0, 1, 1, 0.1, 0, 3, 1, 4, 4, 4, 4)
grid_add(gn, xg, 6, 0, 1, 1, 0.1, 0, 3, 1, 4, 4, 4, 4)
grid_add(gn, xb, 7, 0, 1, 1, 0.1, 0, 3, 1, 4, 4, 4, 4)
grid_add(gn, bx, 8, 0, 1, 1, 0, 0, 3, 1, 4, 4, 4, 4)
grid_add(gn, br, 9, 0, 1, 1, 0.2, 1, 3, 1, 4, 4, 4, 4)
grid_add(gn, bw, 10, 0, 1, 1, 0.2, 0, 1, 1, 4, 4, 4, 4)
grid_add(gn, pp, 11, 0, 1, 1, 1, 0, 3, 1, 4, 4, 4, 4)
var gd = component("grid", "")
config(gd, "border", "Draw")
var rl = component("radio", "Line")
var rt = component("radio", "Text")
config(rl, "active", true)
var rr = component("radio", "Rectangle")
var ro = component("radio", "Oval")
radio_group([rl, rt, rr, ro])
config(pf, "background", "black")
config(pb, "background", "white")
var cf = component("checkbox", "Fill")
config(cf, "active", true)
var cl = component("label", "Width")
var cw = component("combobox", "1,2,3,4,5,6,7,8")
var gt = component("grid", "")
config(gt, "border", "Text")
var co = component("combobox", join(",", fonts()))
```

```
var ts = component("text", "" + font_size)
var tt = component("text", "")
grid_add(gd, rl, 0, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gd, rt, 1, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gd, rr, 2, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gd, ro, 3, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gd, cf, 4, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gd, cl, 5, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gd, cw, 6, 0, 1, 1, 1, 0, 0, 1, 4, 4, 4, 4)
grid_add(gt, co, 0, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gt, ts, 1, 0, 1, 1, 0, 0, 0, 1, 4, 4, 4, 4)
grid_add(gt, tt, 2, 0, 1, 1, 1, 0, 3, 1, 4, 4, 4, 4)

grid_add(g, gn, 0, 0, 2, 1, 0, 0, 1, 1, 4, 4, 4, 4)
grid_add(g, d, 0, 1, 2, 1, 1, 1, 3, 0, 4, 4, 4, 4)
grid_add(g, gd, 0, 2, 1, 1, 1, 0, 3, 1, 4, 4, 4, 4)
grid_add(g, gt, 1, 2, 1, 1, 1, 0, 3, 1, 4, 4, 4, 4)
add(g)
add_s(po)

var x = [null, null, null, null]

event(bf, fn() {
    var r = color_chooser()
    if (r != null) {
        config(d, "foreground", r)
        config(pf, "background", r)
    }
})

event(bb, fn() {
    var r = color_chooser()
    if (r != null) {
        config(d, "background", r)
        config(pb, "background", r)
    }
})

event_mouse(d, fn(a) {
    config(po, "text", a[1] + ", " + a[2])

    if (a[0] == "PRESSED") {
        set(x, 0, a[1])
```

```
        set(x, 1, a[2])
    }
    if (a[0] == "RELEASED") {
        set(x, 2, a[1])
        set(x, 3, a[2])

        draw_width(d, number(get(cw, "text")))

        var f = get(cf, "active")

        var m = get(rl, "active")
        if (m == true) {
            draw_line(d, x[0], x[1], x[2], x[3])
        }

        var m = get(rt, "active")
        if (m == true) {
            var s = trim(get(tt, "contents"))
            var fa = get(co, "text")
              var fs = number(trim(get(ts, "contents")),
font_size)
            config(ts, "contents", fs)
            if (len(s) > 0) {
                config(d, "font", [fa, 0, fs])
                draw_string(d, s, x[0], x[1])
            }
        }

        var m = get(rr, "active")
        if (m == true) {
            if (f == true) {
                    fill_rect(d, x[0], x[1], x[2]-x[0],
x[3]-x[1])
            } else {
                    draw_rect(d, x[0], x[1], x[2]-x[0],
x[3]-x[1])
            }
        }
        var m = get(ro, "active")
        if (m == true) {
            if (f == true) {
                    fill_oval(d, x[0], x[1], x[2]-x[0],
x[3]-x[1])
```

```
            } else {
                    draw_oval(d, x[0], x[1], x[2]-x[0],
x[3]-x[1])
            }
        }
    }
})

event(bx, fn() {
    var r = number(trim(get(xr, "contents")), 0)
    var g = number(trim(get(xg, "contents")), 0)
    var b = number(trim(get(xb, "contents")), 0)
    each(range(0, width), fn(e, i) {
        each(range(0, height), fn(ee, ii) {
            var p = draw_get_pixel(d, e, ee)
            if (p != null) {
                var q = [p[0] + r, p[1] + g, p[2] + b]
                draw_set_pixel(d, e, ee, q)
            }
        })
    })
})

event(br, fn() {
    var f = open()
    if (!empty(f)) {
        draw_read(d, f)
        title(name + ": " + f)
    }
})

event(bw, fn() {
    var action_save = false
    var f = save()
    if (!empty(f)) {
        if (stat(f)["exists"] == true) {
                var res = confirm("File already exists.
Overwrite?",
                "Please confirm")
            if (res == "OK") {
                var action_save = true
            }
        } else {
```

```
            var action_save = true
        }
        if (action_save == true) {
            if (draw_write_png(d, f) == true) {
                title(name + ": " + f)
                message("File saved")
            }
        }
    }
})

show()
```

Contoh Lain

Berikut adalah contoh mendapatkan informasi tentang Singkong dan menampilkannya di sebuah tabel.

```
reset()
var t = component("table","KEY,VALUE,TYPE",
true)
var l = component("label", "Singkong
Programming Language information")

add_n(l)
add(t)

var s = singkong()
var a = []
var f = fn(x,i) {
    var v = s[x]
    var a = a + [x, v, type(v)]
}
each(keys(s), f)

config(t, "contents", a)
show()
```

6. Pengembangan Aplikasi Database

Singkong dapat digunakan untuk mengembangkan aplikasi yang terhubung ke sistem database relasional. Query dapat diberikan dengan mudah, yang akan dijalankan di dalam transaksi. Apabila terjadi kesalahan, transaksi tersebut akan dibatalkan.

Sejak versi 3.0, Singkong.jar datang dengan driver JDBC berikut (yang kompatibel dengan Java 5.0 atau yang lebih baru):

- Apache Derby (sebuah sistem database relational yang ditulis dengan Java)
  - org.apache.derby.jdbc.EmbeddedDriver
  - org.apache.derby.jdbc.ClientDriver
- PostgreSQL
  - org.postgresql.Driver

Dengan demikian, tanpa harus menggunakan driver JDBC tambahan (cukup dengan Singkong.jar), kita dapat terhubung ke sistem database Apache Derby (embedded ataupun network server) dan PostgreSQL.

Lebih lanjut lagi, Singkong.jar juga datang dengan Network Server Apache Derby (kompatibel dengan Java 5.0 atau yang lebih baru), yang memungkinkan server database tersebut dapat dijalankan (dan melayani koneksi dari client), cukup hanya dengan Singkong.jar (tanpa perlu melakukan instalasi tambahan apapun). Contoh cara menjalankan network server Apache Derby akan dibahas secara terpisah di akhir bab ini.

Apabila Anda perlu bekerja dengan sistem database relasional selain Apache Derby dan PostgreSQL, maka driver JDBC

yang sesuai perlu tersedia terlebih dahulu ketika menjalankan interpreter Singkong. Kita akan membutuhkan driver tersebut (umumnya dalam file jar), dan ketika interpreter Singkong dijalankan secara standalone, kita perlu menginformasikan harus mencari driver ke file (jar) atau direktori apa.

Pemrograman database relasional di Singkong telah diusahakan agar sesederhana dan seringkas mungkin, namun karena terhubung dengan sistem eksternal, beberapa informasi berikut mungkin perlu Anda ketahui terlebih dahulu:

- Selain Apache Derby dan PostgreSQL yang disertakan dalam Singkong.jar: dimana driver JDBC/database bisa didapatkan. Umumnya, driver didistribusikan sebagai file jar dan tersedia di situs web sistem database relasional tersebut.
- Nama class driver database yang terkandung dalam file driver yang telah Anda dapatkan. Umumnya, nama class driver terdokumentasi jelas.
- URL untuk terhubung ke sistem database. Ini umumnya berbeda untuk setiap sistem database yang berbeda dan umumnya terdokumentasi jelas.
- Apabila dibutuhkan authentikasi, Anda perlu mengetahui nama user dan passwordnya.

Apabila menggunakan Apache Derby ataupun PostgreSQL, tidak ada perbedaan dalam menjalankan interpreter Singkong, sebagai contoh:

```
java –jar Singkong.jar
```

(Ataupun klik ganda pada Singkong.jar apabila sistem Anda mendukung. Selengkapnya, bacalah juga tentang Menjalankan Interpreter Singkong di bab pertama.)

Apabila menggunakan sistem database lainnya, setelah memastikan bahwa driver database tersimpan dalam file jar

atau direktori tertentu, jalankanlah interpreter Singkong dengan cara berikut. Perintah diketikan dalam baris yang sama. Sesuaikan pemisah class path, apakah sistem yang Anda gunakan menggunakan : (macOS atau Linux) atau ; (Windows).

```
java –cp Singkong.jar:<jar>
com.noprianto.singkong.Singkong

java –cp Singkong.jar:<dir>
com.noprianto.singkong.Singkong

java –cp Singkong.jar;<jar>
com.noprianto.singkong.Singkong

java –cp Singkong.jar;<dir>
com.noprianto.singkong.Singkong
```

Dalam contoh perintah tersebut, <jar> merujuk pada file jar itu sendiri dan <dir> merujuk pada direktori berisikan file class. Pengunaan file jar untuk driver JDBC lebih umum ditemukan. Untuk merujuk pada direktori aktif, umumnya karakter titik digunakan (tidak diperlukan apabila Anda menggunakan driver berupa file jar).

Dengan Java Runtime Environment tahu perlu mencari driver database ke mana, kita siap untuk melakukan koneksi ke sistem database.

Catatan: bacalah juga pembahasan tentang deployment.

Catatan: Sejak Singkong versi 6.1, sebuah modul bawaan, db_util, disertakan untuk mempermudah pengembangan aplikasi database dengan Singkong, tanpa harus bekerja secara langsung dengan perintah-perintah SQL. Bacalah juga pembahasan penggunaan modul db_util di akhir bab ini.

Koneksi Database

Untuk membuat koneksi database, kita menggunakan fungsi built-in database. Fungsi ini mengembalikan sebuah DATABASE (atau NULL apabila gagal), dan menerima empat argumen:

- Nama class driver (STRING)
- URL untuk koneksi database, disebutkan lengkap, termasuk kata jdbc (STRING)
- Nama user (STRING). Apabila ini tidak diperlukan, gunakanlah STRING kosong.
- Password (STRING). Apabila ini tidak diperlukan, gunakanlah STRING kosong.

Berikut adalah contoh koneksi database ke sistem database Apache Derby embedded (tidak perlu ada server Apache Derby yang berjalan):

Di dalam interactive evaluator Singkong:

```
>  var db =
database("org.apache.derby.jdbc.EmbeddedDriver
", "jdbc:derby:test;create=true", "", "")
>  db
DATABASE (URL=jdbc:derby:test;create=true,
user=,
driver=org.apache.derby.jdbc.EmbeddedDriver)
```

Pada contoh tersebut, sebuah database dengan nama test akan dibuat apabila sebelumnya tidak tersedia. Dalam hal ini, apabila operasi berhasil, di direktori aktif, sebuah direktori dengan nama test (berisikan data) akan dibuat. Untuk mengetahui lokasi direktori aktif, gunakanlah fungsi built-in cwd.

Untuk mendapatkan informasi apakah sebuah DATABASE terkoneksi ke sistem database, gunakanlah fungsi built-in database_connected.

Pemetaan Tipe Data

Berikut adalah pemetaan tipe data dari Singkong ke Java, yang otomatis akan dilakukan ketika query diberikan.

Singkong	Java	Catatan
BOOLEAN	boolean	
DATE	java.sql.Date	
NULL	null	
NUMBER	int	Apabila NUMBER terlihat seperti bilangan bulat
NUMBER	java.math.BigDecimal	
STRING	String	

Singkong mendukung tipe-tipe data tersebut ketika bekerja dengan sistem database. Apabila di dalam query diberikan tipe selain yang didukung, maka representasi STRING akan digunakan.

Pemetaan tipe hasil kembalian query dapat dilihat pada tabel berikut. Selebihnya, representasi STRING akan digunakan.

Database	Singkong	Catatan
CLOB	STRING	apabila gagal, representasi STRING akan digunakan
Date	DATE	Waktu: 00:00:00
Timestamp	DATE	

Database	Singkong	Catatan
Integer, Bigint, Decimal, Serial, Bigserial, Numeric	NUMBER	
null	NULL	
Boolean, bool	BOOLEAN	

Untuk tipe database lain, tipe STRING Singkong atau representasi STRING akan digunakan.

Query

Query database di Singkong dapat dilakukan dengan relatif mudah, secara otomatis dikerjakan di dalam sebuah transaksi, dan nilai yang ingin dilewatkan di dalam query menggunakan parameter berupa sebuah ?.

Query dapat dilakukan dengan menggunakan fungsi built-in query. Fungsi ini menerima dua argumen wajib: DATABASE dan ARRAY (dari ARRAY). Setiap elemen dari ARRAY, yang juga bertipe ARRAY, haruslah merupakan ARRAY dengan dua elemen: STRING (perintah SQL) dan ARRAY (argumen, dengan pemetaan tipe sebagaimana dibahas sebelumnya).

Apabila argumen ketiga (opsional, BOOLEAN) diberikan nilai true (default adalah false), maka hasil query akan berisikan juga label dan tipe kolom, apabila memang query mengembalikan hasil demikian (misal: select).

Apabila tidak terjadi kesalahan selama query, transaksi akan di-commit. Apabila terjadi kesalahan, transaksi akan di-rollback (dibatalkan).

```
        >  var d =
database("org.apache.derby.jdbc.EmbeddedDriver",
"jdbc:derby:test;create=true", "", "")
```

```
    >  var q = [ ["create table test(a integer, b
integer)", []] ]
    >  var r = query(d, q)
    >  r
    [0]
    >  var q = [ ["insert into test(a, b) values(?, ?)",
[1, 1]] ]
    >  var r = query(d, q)
    >  r
    [1]
    >  var q = [ ["insert into test(a, b) values(?, ?)",
[2, 2]] ]
    >  var r = query(d, q)
    >  r
    [1]
    >  var q = [ ["select * from test", []] ]
    >  var r = query(d, q)
    >  r
    [[[1, 1], [2, 2]]]
    >  var r = query(d, q, true)
    >  r
    [[[["A", "INTEGER"], ["B", "INTEGER"]], [1, 1], [2,
2]]]
    >  var q = [ ["select * from test", []], ["insert
into test(a, b) values(?, ?)", [3, 3]] ]
    >  var r = query(d, q, true)
    >  r
    [[[["A", "INTEGER"], ["B", "INTEGER"]], [1, 1], [2,
2]], 1]
```

Berikut adalah contoh query ke sistem database untuk pembuatan sebuah tabel, yang dilanjutkan dengan insert dan update, kemudian select, lalu hasilnya ditampilkan di table (GUI):

Menggunakan modul db_util:

```
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()
```

Menggunakan perintah SQL secara langsung:

```
    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()
```

Network Server Apache Derby

Apabila Anda hanya ingin menjalankan Network Server Apache Derby di lingkungan pengembangan atau uji coba (tanpa security manager dan tanpa authentikasi), maka cara menjalankan Network Server Apache Derby cukup membutuhkan perintah berikut (perintah diketikkan dalam satu baris):

```
java -cp Singkong.jar
org.apache.derby.drda.NetworkServerControl
start -noSecurityManager
```

Pastikanlah Anda tidak menggunakan cara tersebut di lingkungan produksi.

Bagaimana kalau Anda ingin menerapkan security policy tertentu dan menyediakan authentikasi bagi user yang ingin melakukan koneksi ke Network Server? Contoh sederhana berikut dapat digunakan. Sesuaikanlah dengan kebutuhan dan konfigurasi sistem Anda. Rujuklah ke dokumentasi terkait apabila memang diperlukan.

- Buatlah file derby.policy di direktori aktif:

```
grant {
            permission java.io.FilePermission "${user.dir}$
{/}-", "read, write, delete";

            permission java.lang.RuntimePermission
"getFileStoreAttributes";
```

```
            permission java.lang.RuntimePermission
"createClassLoader";
            permission java.lang.RuntimePermission
"accessUserInformation";

            permission java.util.PropertyPermission
"derby.__serverStartedFromCmdLine", "read, write";
            permission java.util.PropertyPermission "user.dir",
"read";

            permission java.net.SocketPermission
"127.0.0.1:1527", "accept, connect, listen,resolve";
            permission java.net.SocketPermission "127.0.0.1",
"accept, resolve";

        };
```

- Buatlah file derby.properties di direktori aktif. Kita akan mengunakan provider authentikasi BUILTIN, dimana username dan password ditentukan di dalam file ini. Sebagai contoh, kita telah sediakan sebuah user (dengan nama admin) dan password (admin).

```
derby.connection.requireAuthentication=true
derby.authentication.provider=BUILTIN
derby.user.admin=admin
```

- Jalankanlah Network Server dengan cara berikut (perintah diketikkan dalam satu baris):

```
java -cp Singkong.jar -Djava.security.manager
-Djava.security.policy=derby.policy
org.apache.derby.drda.NetworkServerControl
start
```

Untuk menghentikan Network Server Apache Derby, Anda dapat menekan kombinasi tombol Ctrl-C (SIGINT, signal untuk interrupt) di sistem yang mendukung.

Penggunaan Database Tool

Untuk terhubung ke sistem database, pastikanlah Driver dan URL telah benar. Apabila dibutuhkan username dan password, maka informasi-informasi tersebut juga perlu diberikan.

Apabila koneksi database berhasil dilakukan, maka daftar tabel dalam database tersebut akan ditampilkan, dan perintah SQL dapat diberikan. Untuk menjalankan perintah SQL SELECT, klik gandalah pada nama tabel (kemudian kliklah tombol Run).

Setiap perintah SQL yang valid untuk database yang terhubung dapat diberikan dan apabila terdapat kesalahan, maka pesan kesalahan akan ditampilkan.

Apabila perintah mengembalikan result set, maka akan ditampilkan di tabel hasil. Untuk perintah yang mengembalikan update count, maka jumlah baris yang ter-update akan ditampilkan.

Perintah-perintah SQL tertentu seperti CREATE TABLE dan DROP TABLE akan membuat daftar tabel dimuat ulang. Pemuatan ulang daftar tabel hanya akan dilakukan apabila perintah-perintah tersebut sukses dilakukan.

Catatan: Embedded Derby

Pada penggunaan Embedded Derby, perhatikanlah bahwa hanya satu Java Virtual Machine yang dapat membuka sebuah database pada satu waktu. Koneksi lain dari Java Virtual Machine yang berbeda tidak dapat dilakukan.

Perhatikanlah contoh berikut. Pada interactive evaluator, kita mencoba membuat koneksi-koneksi secara embedded ke database test, dan semuanya dapat dilakukan tanpa ada

kendala (dengan asumsi tidak sedang ada koneksi dengan Java Virtual Machine lain, ke database tersebut).

```
        >  var d =
database("org.apache.derby.jdbc.EmbeddedDriver
", "jdbc:derby:test;create=true", "", "")

        >  d

        DATABASE
(URL=jdbc:derby:test;create=true, user=,
driver=org.apache.derby.jdbc.EmbeddedDriver)

        >  var d =
database("org.apache.derby.jdbc.EmbeddedDriver
", "jdbc:derby:test;create=true", "", "")

        >  d

        DATABASE
(URL=jdbc:derby:test;create=true, user=,
driver=org.apache.derby.jdbc.EmbeddedDriver)
```

Kemudian, dengan interactive evaluator tersebut tetap berjalan, bukalah interpreter Singkong lainnya:

```
        >  var d =
database("org.apache.derby.jdbc.EmbeddedDriver
", "jdbc:derby:test;create=true", "", "")

        >  d

        >  type(d)

        "NULL"
```

Perhatikanlah bahwa koneksi database gagal dilakukan (null). Untuk kebutuhan banyak koneksi pada satu database dengan Java Virtual Machine berbeda, gunakanlah Network Server Apache Derby.

Menggunakan modul db_util

Fungsi yang tersedia dalam modul ini:

create_field_from_array
db_connect
db_connect_embed
db_connect_embed_
db_connect_embed_user
db_create_table
db_create_table_
db_create_table_derby
db_create_table_derby_
db_create_table_embed
db_create_table_embed_
db_create_table_postgresql
db_create_table_postgresql_
db_delete
db_delete_
db_driver
db_insert
db_insert_
db_last
db_last_derby
db_last_embed
db_last_postgresql
db_query_simple
db_query_single
db_run_query
db_select
db_select_
db_select_all
db_select_all_
db_select_derby

db_select_derby_
db_select_embed
db_select_embed_
db_select_postgresql
db_select_postgresql_
db_update
db_update_
query_result

load_module(“db_util")

A. Koneksi ke database

Fungsi-fungsi yang dapat digunakan:

db_connect
db_connect_embed
db_connect_embed_
db_connect_embed_user

Embedded Derby (nama database: test) di direktori aktif (buat, apabila tidak ada database tersebut):

```
> var d_embed = db_connect_embed("test")
> d_embed
DATABASE (URL=jdbc:derby:test;create=true,
user=, driver=org.apache.derby.jdbc.EmbeddedDriver)
```

Embedded Derby (nama database: test) di home directory

```
> var d_embed_user =
db_connect_embed_user("test")
> d_embed_user
DATABASE (URL=jdbc:derby:/home/test/
test;create=true, user=,
driver=org.apache.derby.jdbc.EmbeddedDriver)
```

Derby di localhost:1527, nama database: test (buat, apabila tidak ada database tersebut), user: admin, password: admin

```
> var d_derby = db_connect("derby", "//
localhost:1527/test;create=true", "admin", "admin")
> d_derby
```

```
            DATABASE (URL=jdbc:derby://localhost:1527/
test;create=true, user=admin,
driver=org.apache.derby.jdbc.ClientDriver)
```

PostgreSQL di localhost, nama database: test, user: test, password: test

```
            >  var d_pgsql = db_connect("postgresql",
"//localhost/test", "test", "test")
            >  d_pgsql
            DATABASE (URL=jdbc:postgresql://localhost/
test, user=test, driver=org.postgresql.Driver)
```

B. Membuat tabel

Fungsi-fungsi yang dapat digunakan:

db_create_table
db_create_table_derby
db_create_table_embed
db_create_table_postgresql

Pemetaan tipe kolom:

Derby atau Embedded Derby:

"id": "integer not null generated always as identity (start with 1, increment by 1)"
"varchar": "varchar (32672)"
"varchar.": "varchar (32672) default ''"
"text": "clob"
"text.": "clob default ''"
"decimal": "decimal(19,4)"
"decimal.": "decimal(19,4) default 0"
"integer": "integer"
"integer.": "integer default 0"
"boolean": "boolean"
"boolean.": "boolean default false"
"date": "date"
"date.": "date default current_date"
"timestamp": "timestamp"
"timestamp.": "timestamp default current_timestamp"

PostgreSQL:

```
"id": "serial"
"varchar": "varchar"
"varchar.": "varchar default ''"
"text": "text"
"text.": "text default ''"
"decimal": "decimal(19,4)"
"decimal.": "decimal(19,4) default 0"
"integer": "integer"
"integer.": "integer default 0"
"boolean": "boolean"
"boolean.": "boolean default false"
"date": "date"
"date.": "date default current_date"
"timestamp": "timestamp"
"timestamp.": "timestamp default
current_timestamp”
```

Contoh:

Derby:

```
                > var r =
db_create_table_derby(d_derby, "test_table", [["a",
"id"], ["b", "varchar."], ["c", "text."], ["d",
"decimal."], ["e", "integer."], ["f", "boolean."], ["g",
"date."], ["h", "timestamp."]])
                > r
                [[0], "create table test_table (a
integer not null generated always as identity (start
with 1, increment by 1),b varchar (32672) default '',c
clob default '',d decimal(19,4) default 0,e integer
default 0,f boolean default false,g date default
current_date,h timestamp default current_timestamp)"]
```

Embedded Derby:

```
                > var r =
db_create_table_embed(d_embed, "test_table", [["a",
"id"], ["b", "varchar."], ["c", "text."], ["d",
"decimal."], ["e", "integer."], ["f", "boolean."], ["g",
"date."], ["h", "timestamp."]])
```

```
> r
[[0], "create table test_table (a
integer not null generated always as identity (start
with 1, increment by 1),b varchar (32672) default '',c
clob default '',d decimal(19,4) default 0,e integer
default 0,f boolean default false,g date default
current_date,h timestamp default current_timestamp)"]
```

PostgreSQL:

```
> var r =
db_create_table_postgresql(d_pgsql, "test_table", [["a",
"id"], ["b", "varchar."], ["c", "text."], ["d",
"decimal."], ["e", "integer."], ["f", "boolean."], ["g",
"date."], ["h", "timestamp."]])
> r
[[0], "create table test_table (a
serial,b varchar default '',c text default '',d
decimal(19,4) default 0,e integer default 0,f boolean
default false,g date default current_date,h timestamp
default current_timestamp)"]
```

Perintah SQL saja (perhatikanlah underscore pada nama fungsi):

```
> var r =
db_create_table_derby_(d_derby, "test_table", [["a",
"id"], ["b", "varchar."], ["c", "text."], ["d",
"decimal."], ["e", "integer."], ["f", "boolean."], ["g",
"date."], ["h", "timestamp."]])
> r
["create table test_table (a integer not
null generated always as identity (start with 1,
increment by 1),b varchar (32672) default '',c clob
default '',d decimal(19,4) default 0,e integer default
0,f boolean default false,g date default current_date,h
timestamp default current_timestamp)", []]
> var r =
db_create_table_embed_(d_embed, "test_table", [["a",
"id"], ["b", "varchar."], ["c", "text."], ["d",
"decimal."], ["e", "integer."], ["f", "boolean."], ["g",
"date."], ["h", "timestamp."]])
> r
["create table test_table (a integer not
null generated always as identity (start with 1,
```

```
increment by 1),b varchar (32672) default '',c clob
default '',d decimal(19,4) default 0,e integer default
0,f boolean default false,g date default current_date,h
timestamp default current_timestamp)", []]
                > var r =
db_create_table_postgresql_(d_pgsql, "test_table",
[["a", "id"], ["b", "varchar."], ["c", "text."], ["d",
"decimal."], ["e", "integer."], ["f", "boolean."], ["g",
"date."], ["h", "timestamp."]])
                > r
                ["create table test_table (a serial,b
varchar default '',c text default '',d decimal(19,4)
default 0,e integer default 0,f boolean default false,g
date default current_date,h timestamp default
current_timestamp)", []]
```

C. Insert

Derby:

```
                > var r = db_insert(d_derby,
"test_table", {"b": "Test", "e": 12345})
                > r
                [1]
```

Embedded Derby:

```
                > var r = db_insert(d_embed,
"test_table", {"b": "Test", "e": 12345})
                > r
                [1]
```

PostgreSQL:

```
                > var r = db_insert(d_pgsql,
"test_table", {"b": "Test", "e": 12345})
                > r
                [1]
```

Perintah SQL saja (perhatikanlah underscore pada nama fungsi):

```
                > db_insert_(d_derby, "test_table",
{"b": "Test", "e": 12345})
                ["insert into test_table (b,e) values
(?,?)", ["Test", 12345]]
```

```
                >  db_insert_(d_embed, "test_table",
{"b": "Test", "e": 12345})
                ["insert into test_table (b,e) values
(?,?)", ["Test", 12345]]
                >  db_insert_(d_pgsql, "test_table",
{"b": "Test", "e": 12345})
                ["insert into test_table (b,e) values
(?,?)", ["Test", 12345]]
```

D. Last insert id

Derby:

```
                >  db_last_derby(d_derby)
                1
```

Embedded Derby:

```
                >  db_last_embed(d_embed)
                1
```

PostgreSQL:

```
                >  db_last_postgresql(d_pgsql)
                1
```

Derby:

```
                >  var r = db_insert(d_derby,
"test_table", {"b": "Test", "e": 12345})
                >  db_last_derby(d_derby)
                2
```

Embedded Derby:

```
                >  var r = db_insert(d_embed,
"test_table", {"b": "Test", "e": 12345})
                >  db_last_embed(d_embed)
                2
```

PostgreSQL:

```
                >  var r = db_insert(d_pgsql,
"test_table", {"b": "Test", "e": 12345})
                >  db_last_postgresql(d_pgsql)
                2
```

E. Select (semua baris, semua kolom)
Derby:

```
                > db_select_all(d_derby, "test_table")
                [[[1, "Test", "", 0, 12345, false,
2021-08-15 00:00:00, 2021-08-15 17:44:40], [2, "Test",
"", 0, 12345, false, 2021-08-15 00:00:00, 2021-08-15
17:47:40]]]
```

Embedded Derby:

```
                > db_select_all(d_embed, "test_table")
                [[[1, "Test", "", 0, 12345, false,
2021-08-15 00:00:00, 2021-08-15 17:44:57], [2, "Test",
"", 0, 12345, false, 2021-08-15 00:00:00, 2021-08-15
17:48:12]]]
```

PostgreSQL:

```
                > db_select_all(d_pgsql, "test_table")
                [[[1, "Test", "", 0, 12345, false,
2021-08-15 00:00:00, 2021-08-15 17:45:06], [2, "Test",
"", 0, 12345, false, 2021-08-15 00:00:00, 2021-08-15
17:48:26]]]
```

Perintah SQL saja:

```
                > db_select_all_(d_derby, "test_table")
                ["select * from test_table ", []]
                > db_select_all_(d_embed, "test_table")
                ["select * from test_table ", []]
                > db_select_all_(d_pgsql, "test_table")
                ["select * from test_table ", []]
```

F. Select (column, where, order, offset, fetch/limit)
Derby:

```
                > db_select_derby(d_derby,
"test_table", [], [], ["a asc", "b desc"], null, null)
                [[[1, "Test", "", 0, 12345, false,
2021-08-15 00:00:00, 2021-08-15 17:44:40], [2, "Test",
"", 0, 12345, false, 2021-08-15 00:00:00, 2021-08-15
17:47:40]]]
                > db_select_derby(d_derby,
"test_table", ["a", "b"], [], ["a desc"], null, null)
```

```
                [[[2, "Test"], [1, "Test"]]]
                >  db_select_derby(d_derby,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], null, null)
                [[[2, "Test", ""], [1, "Test", ""]]]
                >  db_select_derby(d_derby,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 1, null)
                [[[1, "Test", ""]]]
                >  db_select_derby(d_derby,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 0, 1)
                [[[2, "Test", ""]]]
```

Perintah SQL saja:

```
                >  db_select_derby_(d_derby,
"test_table", [], [], ["a asc", "b desc"], null, null)
                ["select * from test_table  order by a
asc,b desc", []]
                >  db_select_derby_(d_derby,
"test_table", ["a", "b"], [], ["a desc"], null, null)
                ["select a,b from test_table  order by a
desc", []]
                >  db_select_derby_(d_derby,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], null, null)
                ["select a,b,c from test_table where a >
? and e =  ?   order by a desc", [0, 12345]]
                >  db_select_derby_(d_derby,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 1, null)
                ["select a,b,c from test_table where a >
? and e =  ?   order by a desc offset 1 rows ", [0,
12345]]
                >  db_select_derby_(d_derby,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 0, 1)
                ["select a,b,c from test_table where a >
? and e =  ?   order by a desc offset 0 rows  fetch
first 1 rows only ", [0, 12345]]
```

Embedded Derby:

```
                > db_select_embed(d_embed,
"test_table", [], [], ["a asc", "b desc"], null, null)
                [[[1, "Test", "", 0, 12345, false,
2021-08-15 00:00:00, 2021-08-15 17:44:57], [2, "Test",
"", 0, 12345, false, 2021-08-15 00:00:00, 2021-08-15
17:48:12]]]
                > db_select_embed(d_embed,
"test_table", ["a", "b"], [], ["a desc"], null, null)
                [[[2, "Test"], [1, "Test"]]]
                > db_select_embed(d_embed,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], null, null)
                [[[2, "Test", ""], [1, "Test", ""]]]
                > db_select_embed(d_embed,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 1, null)
                [[[1, "Test", ""]]]
                > db_select_embed(d_embed,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 0, 1)
                [[[2, "Test", ""]]]
```

Perintah SQL saja:

```
                > db_select_embed_(d_embed,
"test_table", [], [], ["a asc", "b desc"], null, null)
                ["select * from test_table  order by a
asc,b desc", []]
                > db_select_embed_(d_embed,
"test_table", ["a", "b"], [], ["a desc"], null, null)
                ["select a,b from test_table  order by a
desc", []]
                > db_select_embed_(d_embed,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], null, null)
                ["select a,b,c from test_table where a >
? and e =  ?   order by a desc", [0, 12345]]
                > db_select_embed_(d_embed,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 1, null)
                ["select a,b,c from test_table where a >
? and e =  ?   order by a desc offset 1 rows ", [0,
12345]]
```

```
                >  db_select_embed_(d_embed,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 0, 1)
                ["select a,b,c from test_table where a >
? and e =  ?   order by a desc offset 0 rows  fetch
first 1 rows only ", [0, 12345]]
```

PostgreSQL:

```
                >  db_select_postgresql(d_pgsql,
"test_table", [], [], ["a asc", "b desc"], null, null)
                [[[1, "Test", "", 0, 12345, false,
2021-08-15 00:00:00, 2021-08-15 17:45:06], [2, "Test",
"", 0, 12345, false, 2021-08-15 00:00:00, 2021-08-15
17:48:26]]]
                >  db_select_postgresql(d_pgsql,
"test_table", ["a", "b"], [], ["a desc"], null, null)
                [[[2, "Test"], [1, "Test"]]]
                >  db_select_postgresql(d_pgsql,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], null, null)
                [[[2, "Test", ""], [1, "Test", ""]]]
                >  db_select_postgresql(d_pgsql,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 1, null)
                [[[1, "Test", ""]]]
                >  db_select_postgresql(d_pgsql,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 0, 1)
                [[[2, "Test", ""]]]
```

Perintah SQL saja:

```
                >  db_select_postgresql_(d_pgsql,
"test_table", [], [], ["a asc", "b desc"], null, null)
                ["select * from test_table  order by a
asc,b desc", []]
                >  db_select_postgresql_(d_pgsql,
"test_table", ["a", "b"], [], ["a desc"], null, null)
                ["select a,b from test_table  order by a
desc", []]
                >  db_select_postgresql_(d_pgsql,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], null, null)
```

```
                ["select a,b,c from test_table where a >
? and e =  ?   order by a desc", [0, 12345]]
                > db_select_postgresql_(d_pgsql,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 1, null)
                ["select a,b,c from test_table where a >
? and e =  ?   order by a desc offset 1 rows ", [0,
12345]]
                > db_select_postgresql_(d_pgsql,
"test_table", ["a", "b", "c"], [["a > ", 0, "and"], ["e
= ", 12345, ""]], ["a desc"], 0, 1)
                ["select a,b,c from test_table where a >
? and e =  ?   order by a desc offset 0 rows  limit 1",
[0, 12345]]
```

G. Update (semua baris)

Derby:

```
                > db_update(d_derby, "test_table", [],
{"d": 12.34})
                [2]
```

Embedded Derby:

```
                > db_update(d_embed, "test_table", [],
{"d": 12.34})
                [2]
```

PostgreSQL:

```
                > db_update(d_pgsql, "test_table", [],
{"d": 12.34})
                [2]
```

Perintah SQL saja:

```
                > db_update_(d_derby, "test_table", [],
{"d": 12.34})
                ["update test_table set d=?", [12.3400]]
                > db_update_(d_embed, "test_table", [],
{"d": 12.34})
                ["update test_table set d=?", [12.3400]]
                > db_update_(d_pgsql, "test_table", [],
{"d": 12.34})
                ["update test_table set d=?", [12.3400]]
```

H. Update (where)

Derby:

```
                > db_update(d_derby, "test_table", [["e
> ", 10000, "and"], ["a < ", 2, ""]], {"d": 12.345})
                [1]
```

Embedded Derby:

```
                > db_update(d_embed, "test_table", [["e
> ", 10000, "and"], ["a < ", 2, ""]], {"d": 12.345})
                [1]
```

PostgreSQL:

```
                > db_update(d_pgsql, "test_table", [["e
> ", 10000, "and"], ["a < ", 2, ""]], {"d": 12.345})
                [1]
```

Perintah SQL saja:

```
                > db_update_(d_derby, "test_table",
[["e > ", 10000, "and"], ["a < ", 2, ""]], {"d":
12.345})
                ["update test_table set d=? where e >  ?
and a <  ? ", [12.3450, 10000, 2]]
                > db_update_(d_embed, "test_table",
[["e > ", 10000, "and"], ["a < ", 2, ""]], {"d":
12.345})
                ["update test_table set d=? where e >  ?
and a <  ? ", [12.3450, 10000, 2]]
                > db_update_(d_pgsql, "test_table",
[["e > ", 10000, "and"], ["a < ", 2, ""]], {"d":
12.345})
                ["update test_table set d=? where e >  ?
and a <  ? ", [12.3450, 10000, 2]]
```

I. Delete (where)

Derby:

```
                > db_delete(d_derby, "test_table", [["a
> ", 1, ""]])
                [1]
```

Embedded Derby:

```
                > db_delete(d_embed, "test_table", [["a
> ", 1, ""]])
                [1]
```

PostgreSQL:

```
                > db_delete(d_pgsql, "test_table", [["a
> ", 1, ""]])
                [1]
```

Perintah SQL saja:

```
                > db_delete_(d_derby, "test_table",
[["a > ", 1, ""]])
                ["delete from test_table where a >  ? ",
[1]]
                > db_delete_(d_embed, "test_table",
[["a > ", 1, ""]])
                ["delete from test_table where a >  ? ",
[1]]
                > db_delete_(d_pgsql, "test_table",
[["a > ", 1, ""]])
                ["delete from test_table where a >  ? ",
[1]]
```

J. Query sederhana (tanpa argumen)

Derby:

```
                > db_query_simple(d_derby, "select *
from test_table")
                [[[1, "Test", "", 12.3450, 12345, false,
2021-08-15 00:00:00, 2021-08-15 17:44:40]]]
```

Embedded Derby:

```
                > db_query_simple(d_embed, "select *
from test_table")
                [[[1, "Test", "", 12.3450, 12345, false,
2021-08-15 00:00:00, 2021-08-15 17:44:57]]]
```

PostgreSQL:

```
                > db_query_simple(d_pgsql, "select *
from test_table")
                [[[1, "Test", "", 12.3450, 12345, false,
2021-08-15 00:00:00, 2021-08-15 17:45:06]]]
```

K. Query tunggal (dengan argumen)

Derby:

```
               >  db_query_single(d_derby, "select a, b
from test_table where a > ?", [0])
               [[[1, "Test"]]]
```

Embedded Derby:

```
               >  db_query_single(d_embed, "select a, b
from test_table where a > ?", [0])
               [[[1, "Test"]]]
```

PostgreSQL:

```
               >  db_query_single(d_pgsql, "select a, b
from test_table where a > ?", [0])
               [[[1, "Test"]]]
```

L. Delete (semua baris)

Derby:

```
               >  db_delete(d_derby, "test_table", [])
               [1]
               >  db_select_all(d_derby, "test_table")
               [[]]
```

Embedded Derby:

```
               >  db_delete(d_embed, "test_table", [])
               [1]
               >  db_select_all(d_embed, "test_table")
               [[]]
```

PostgreSQL:

```
               >  db_delete(d_pgsql, "test_table", [])
               [1]
               >  db_select_all(d_pgsql, "test_table")
               [[]]
```

Perintah SQL saja:

```
               >  db_delete_(d_derby, "test_table", [])
               ["delete from test_table", []]
               >  db_delete_(d_embed, "test_table", [])
               ["delete from test_table", []]
```

```
> db_delete_(d_pgsql, "test_table", [])
["delete from test_table", []]
```

M. Menggunakan fungsi bawaan: query

Derby:

Query (dengan transaksi) OK

```
> var q1 = db_insert_(d_derby,
"test_table", {"b": "Test", "e": 12345})
> q1
["insert into test_table (b,e) values
(?,?)", ["Test", 12345]]
> var q2 = db_update_(d_derby,
"test_table", [], {"d": 12.34})
> q2
["update test_table set d=?", [12.3400]]
> query(d_derby, [q1, q2])
[1, 1]
> db_select_all(d_derby, "test_table")
[[[3, "Test", "", 12.3400, 12345, false,
2021-08-15 00:00:00, 2021-08-15 18:25:34]]]
```

Query (dengan transaksi) Gagal
(Pada tabel test_table, tidak ada kolom dengan nama ‘error’)

```
> var q3 = db_update_(d_derby,
"test_table", [], {"error": 12.34})
> q1
["insert into test_table (b,e) values
(?,?)", ["Test", 12345]]
> q3
["update test_table set error=?",
[12.3400]]
> println(query(d_derby, [q1, q3]))
null
> db_select_all(d_derby, "test_table")
[[[3, "Test", "", 12.3400, 12345, false,
2021-08-15 00:00:00, 2021-08-15 18:25:34]]]
```

Embedded Derby:

Query (dengan transaksi) OK

```
> var q1 = db_insert_(d_embed,
"test_table", {"b": "Test", "e": 12345})
> q1
```

```
                ["insert into test_table (b,e) values
(?,?)", ["Test", 12345]]
                >  var q2 = db_update_(d_embed,
"test_table", [], {"d": 12.34})
                >  q2
                ["update test_table set d=?", [12.3400]]
                >  query(d_embed, [q1, q2])
                [1, 1]
                >  db_select_all(d_embed, "test_table")
                [[[3, "Test", "", 12.3400, 12345, false,
2021-08-15 00:00:00, 2021-08-15 18:31:27]]]
```

Query (dengan transaksi) Gagal
(Pada tabel test_table, tidak ada kolom dengan nama ‘error’)

```
                >  var q3 = db_update_(d_embed,
"test_table", [], {"error": 12.34})
                >  q1
                ["insert into test_table (b,e) values
(?,?)", ["Test", 12345]]
                >  q3
                ["update test_table set error=?",
[12.3400]]
                >  println(query(d_embed, [q1, q3]))
                null
                >  db_select_all(d_embed, "test_table")
                [[[3, "Test", "", 12.3400, 12345, false,
2021-08-15 00:00:00, 2021-08-15 18:31:27]]]
```

PostgreSQL:
Query (dengan transaksi) OK

```
                >  var q1 = db_insert_(d_pgsql,
"test_table", {"b": "Test", "e": 12345})
                >  q1
                ["insert into test_table (b,e) values
(?,?)", ["Test", 12345]]
                >  var q2 = db_update_(d_pgsql,
"test_table", [], {"d": 12.34})
                >  q2
                ["update test_table set d=?", [12.3400]]
                >  query(d_pgsql, [q1, q2])
                [1, 1]
                >  db_select_all(d_pgsql, "test_table")
```

```
                [[[3, "Test", "", 12.3400, 12345, false,
2021-08-15 00:00:00, 2021-08-15 18:34:53]]]
```

Query (dengan transaksi) Gagal
(Pada tabel test_table, tidak ada kolom dengan nama ‘error’)

```
                >  var q3 = db_update_(d_pgsql,
"test_table", [], {"error": 12.34})
                >  q1
                ["insert into test_table (b,e) values
(?,?)", ["Test", 12345]]
                >  q3
                ["update test_table set error=?",
[12.3400]]
                >  println(query(d_pgsql, [q1, q3]))
                null
                >  db_select_all(d_pgsql, "test_table")
                [[[3, "Test", "", 12.3400, 12345, false,
2021-08-15 00:00:00, 2021-08-15 18:34:53]]]
```

7. Pengembangan Aplikasi Web

Singkong mendukung pengembangan aplikasi web sederhana dalam bentuk CGI (Common Gateway Interface). Walaupun CGI memiliki keterbatasan diantaranya karena interpreter Singkong perlu dijalankan setiap kali request diproses (membutuhkan sumber daya sistem yang relatif lebih besar dibanding cara kerja lain), untuk saat ini, hanya CGI lah yang didukung oleh Singkong. Penulis sendiri juga menggunakan Singkong dan CGI sebagai bagian dari backend aplikasi.

Pengembangan aplikasi web dengan Singkong membutuhkan sebuah HTTP server dengan dukungan CGI, sebagai contoh Apache HTTP server.

Sebelum memulai, pastikanlah dukungan CGI telah diaktifkan pada HTTP server dan Anda dapat menempatkan skrip yang akan dijalankan pada direktori yang ditentukan. Konfigurasi HTTP Server berada diluar cakupan buku ini.

Shell Script / Batch File

Request HTTP akan diterima terlebih dahulu oleh shell script atau batch file, sesuai dengan sistem operasi yang Anda gunakan. Dengan demikian, ekstensi file ini perlu dikonfigur agar dikenal sebagai skrip CGI.

Shell script atau batch file inilah yang akan menjalankan interpreter Singkong dan program Singkong yang Anda buat. Sebagai contoh, di dalam shell script (index.sh) atau batch file (index.bat):

```
java –jar Singkong.jar index.singkong
```

File program index.singkong sendiri disarankan agar ditempatkan di luar direktori yang bisa diakses secara publik

(misal di luar direktori yang akan diproses oleh HTTP Server yang Anda gunakan).

Apabila terdapat fitur untuk command echoing, maka nonaktifkanlah terlebih dahulu. Kita akan melihatnya di dalam contoh-contoh skrip CGI di bab ini.

Pastikanlah hak akses file yang diperlukan telah di set (misal: executable pada sistem operasi yang mendukung).

Header

Setiap program CGI perlu mengirimkan header (mencetak ke standard output). Di singkong, fungsi built-in cgi_header dapat digunakan. Apabila fungsi ini dipanggil tanpa argumen apapun, maka header berikut akan dikirimkan (dicetak ke standard output):

Content-Type: text/html

Apabila Anda perlu mengirimkan response berupa JSON, maka contoh header berikut dapat digunakan:

```
var h = {"Content-Type": "application/json"}
cgi_header(h)
```

Konten

Untuk mengirimkan konten, kita dapat menggunakan fungsi yang mencetak ke standard output, seperti fungsi built-in print atau println. Walau, untuk mempermudah, fungsi built-in cgi_contents disediakan.

Pastikanlah konten dikirimkan setelah header.

HTTP GET

Untuk mendapatkan QUERY_STRING (dalam format x-www-form-urlencoded) dari request HTTP GET sebagai HASH (dengan key/value telah di-decode), gunakanlah fungsi built-in cgi_get.

Anda mungkin ingin menggunakan fungsi built-in env untuk mendapatkan environment variable.

HTTP POST

Untuk mendapatkan body (dalam format x-www-form-urlencoded) dari request HTTP POST sebagai HASH (dengan key/value telah di-decode), gunakanlah fungsi built-in cgi_post.

Untuk mendapatkan body (dalam format x-www-form-urlencoded, representasi STRING dari HASH) dari request HTTP POST sebagai HASH (dengan key/value telah di-decode), gunakanlah fungsi built-in cgi_post_hash.

Sebagaimana halnya ketika memroses request GET, Anda juga mungkin ingin mendapatkan environment variable dengan fungsi built-in env.

Contoh: Variabel

Apabila Anda menggunakan shell Bash, berikut adalah isi file index.sh. Pastikanlah path ke binary bash telah benar dan index.sh telah memiliki hak akses executable.

```
    #!/bin/bash

    # Please put this file in CGI-enabled directory of the web
server

    # Please adjust the variables appropriately
    JAVA=java
```

```
SINGKONG=/home/user/singkong/Singkong.jar
PROGRAM=/home/user/singkong/index.singkong

$JAVA -jar $SINGKONG $PROGRAM
```

Apabila Anda menggunakan batch file, berikut adalah isi index.bat:

```
@echo off

REM Please put this file in CGI-enabled directory of the
web server

REM Please adjust the variables appropriately
set JAVA=java
set SINGKONG=c:\singkong\Singkong.jar
set PROGRAM=c:\singkong\index.singkong

%JAVA% -jar %SINGKONG% %PROGRAM%
```

Berikut ini adalah isi file index.singkong:

```
var e = env()
var c = [
    "<!DOCTYPE html>
    <html lang='en'>
        <head>
            <meta charset='UTF-8'>
            <title>Singkong</title>
        </head>
        <body>
            Remote address: " + e["REMOTE_ADDR"] + "<br>
            HTTP user agent: " + e["HTTP_USER_AGENT"] + "
        </body>
    </html>"
]

cgi_header()
cgi_contents(c)
```

Contoh: Method GET

Berikut adalah contoh request GET yang membutuhkan parameter min dan max (query string: min=<number>&max=<number>). Respon akan diberikan dalam format JSON.

Apabila min dan max diberikan valid, maka nilai acak antara keduanya akan dikembalikan.

Apabila Anda menggunakan shell Bash, berikut adalah isi file random.sh. Pastikanlah path ke binary bash telah benar dan random.sh telah memiliki hak akses executable.

```
    #!/bin/bash

    # Please put this file in CGI-enabled directory of the web
server

    # Please adjust the variables appropriately
    JAVA=java
    SINGKONG=/home/user/singkong/Singkong.jar
    PROGRAM=/home/user/singkong/random.singkong

    $JAVA -jar $SINGKONG $PROGRAM
```

Apabila Anda menggunakan batch file, berikut adalah isi random.bat:

```
    @echo off

    REM Please put this file in CGI-enabled directory of the
web server

    REM Please adjust the variables appropriately
    set JAVA=java
    set SINGKONG=c:\singkong\Singkong.jar
    set PROGRAM=c:\singkong\random.singkong

    %JAVA% -jar %SINGKONG% %PROGRAM%
```

Berikut ini adalah isi file random.singkong:

```
var h = {"Content-Type": "application/json"}
var r = {"random": ""}
var p = cgi_get()
var x = p["min"]
var y = p["max"]

if (is(x, "STRING") & is(y, "STRING")) {
    var z = random(number(x), number(y))
    if (z != null) {
        set(r, "random", string(z))
    }
}

cgi_header(h)
println(r)
```

Contoh: Method POST

Berikut adalah contoh request POST yang membutuhkan parameter username dan password (body: username=<string>&password=<string>). Respon akan diberikan dalam format JSON.

Apabila username dan password masing-masing bernilai admin, maka authentikasi akan dianggap berhasil (true).

Apabila Anda menggunakan shell Bash, berikut adalah isi file auth.sh. Pastikanlah path ke binary bash telah benar dan auth.sh telah memiliki hak akses executable.

```
#!/bin/bash

# Please put this file in CGI-enabled directory of the web
server

# Please adjust the variables appropriately
JAVA=java
SINGKONG=/home/user/singkong/Singkong.jar
PROGRAM=/home/user/singkong/auth.singkong
```

```
$JAVA -jar $SINGKONG $PROGRAM
```

Apabila Anda menggunakan batch file, berikut adalah isi auth.bat:

```
@echo off

REM Please put this file in CGI-enabled directory of the
web server

REM Please adjust the variables appropriately
set JAVA=java
set SINGKONG=c:\singkong\Singkong.jar
set PROGRAM=c:\singkong\auth.singkong

%JAVA% -jar %SINGKONG% %PROGRAM%
```

Berikut ini adalah isi file auth.singkong:

```
var h = {"Content-Type": "application/json"}
var r = {"result": false}
var p = cgi_post()
var x = p["username"]
var y = p["password"]

if (is(x, "STRING") & is(y, "STRING")) {
    set(r, "result", x == "admin" & y == "admin")
}

cgi_header(h)
println(r)
```

Sebagai HTTP client untuk contoh method POST ini, gunakanlah contoh program login.singkong berikut (HTTP client akan dibahas pada bab berikut):

```
var url = trim(input("URL", "Login"))
if (!startswith(url, "http")) {
    exit()
}

var login = login_dialog("Login")
```

```
if (len(login) == 2) {
    var u = login[0]
    var p = login[1]
    var data = "username=" + u + "&password=" + p
    var res = http_post(url, data)
    if (res != null) {
        if (is(res, "ARRAY")) {
            if (len(res) == 3) {
                var r = parse_hash(res[2])
                message(r["result"])
            }
        }
    }
}
```

8. HTTP Client

Singkong datang dengan beberapa fungsi built-in untuk melakukan request HTTP (dalam berbagai method).

Nilai Default

Berikut ini adalah nilai-nilai default yang digunakan dalam setiap kali request:

- String user agent adalah Singkong <versi>.
- Timeout adalah 10 detik.
- Accept-Charset: UTF-8

Nilai-nilai tersebut bisa diubah, baik sebagai argumen dalam pemanggilan fungsi ataupun dengan pengaturan header request.

URL Encode/Decode

Untuk melakukan encoding ke atau decoding dari x-www-form-urlencoded, gunakanlah fungsi-fungsi built-in url_encode atau url_decode.

Contoh:

```
>  var e = url_encode("Singkong Programming
Language")
>  e
“Singkong+Programming+Language"
>  var d = url_decode(e)
>  d
"Singkong Programming Language”
```

HEAD

Fungsi built-in yang digunakan untuk metode ini adalah http_head. Fungsi ini menerima argumen wajib URL (STRING) dan argumen opsional timeout (NUMBER, milidetik) dan header (HASH).

Apabila header respon yang diterima mengandung Location, maka redirect tersebut akan diikuti, namun hanya sekali. Redirect yang didukung adalah 301, 302, 303. Redireksi dari HTTP ke HTTPS juga didukung.

Apabila terjadi kegagalan, fungsi akan mengembalikan NULL. Apabila berhasil, fungsi mengembalikan ARRAY (headers (HASH), response code (NUMBER), data (STRING)).

GET

Fungsi built-in pertama yang digunakan untuk metode ini adalah http_get. Fungsi ini menerima argumen wajib URL (STRING) dan argumen opsional timeout (NUMBER, milidetik) dan header (HASH).

Apabila header respon yang diterima mengandung Location, maka redirect tersebut akan diikuti, namun hanya sekali. Redirect yang didukung adalah 301, 302, 303. Redireksi dari HTTP ke HTTPS juga didukung.

Apabila terjadi kegagalan, fungsi akan mengembalikan NULL. Apabila berhasil, fungsi mengembalikan ARRAY (headers (HASH), response code (NUMBER), data (STRING)).

Untuk mendownload file, gunakanlah fungsi built-in http_get_file. Argumen wajib dalam pemanggilan fungsi ini adalah URL (STRING) dan nama file yang akan disimpan (STRING). Selebihnya, fungsi ini menerima argumen opsional dan memiliki cara kerja serta return value yang sama dengan http_get.

POST

Fungsi built-in pertama yang digunakan untuk metode ini adalah http_post. Fungsi ini menerima argumen wajib URL (STRING) dan body (STRING), serta argumen opsional timeout (NUMBER, milidetik) dan header (HASH).

Catatan:
- Content-Type: application/x-www-form-urlencoded;charset=UTF-8
- body request dapat berupa STRING kosong. URL encoding diperlukan.

Apabila terjadi kegagalan, fungsi akan mengembalikan NULL. Apabila berhasil, fungsi mengembalikan ARRAY (headers (HASH), response code (NUMBER), data (STRING)).

Untuk melakukan request POST dengan X-HTTP-Method-Override, gunakanlah fungsi built-in http_post_override. Argumen wajib dalam pemanggilan fungsi ini adalah URL (STRING), body (STRING), dan method (STRING). Selebihnya, fungsi ini menerima argumen opsional dan memiliki cara kerja serta return value yang sama dengan http_post.

PUT

Fungsi built-in yang digunakan untuk metode ini adalah http_put. Fungsi ini menerima argumen wajib URL (STRING) dan body (STRING), serta argumen opsional timeout (NUMBER, milidetik) dan header (HASH).

Catatan:
- Content-Type: application/x-www-form-urlencoded;charset=UTF-8
- body request dapat berupa STRING kosong. URL encoding diperlukan.

Apabila terjadi kegagalan, fungsi akan mengembalikan NULL. Apabila berhasil, fungsi mengembalikan ARRAY (headers (HASH), response code (NUMBER), data (STRING)).

DELETE

Fungsi built-in yang digunakan untuk metode ini adalah http_delete. Fungsi ini menerima argumen wajib URL (STRING), serta argumen opsional timeout (NUMBER, milidetik) dan header (HASH).

Catatan:
- Content-Type: application/x-www-form-urlencoded;charset=UTF-8
- body request tidak dapat diberikan / tidak didukung.

Apabila terjadi kegagalan, fungsi akan mengembalikan NULL. Apabila berhasil, fungsi mengembalikan ARRAY (headers (HASH), response code (NUMBER), data (STRING)).

Method yang Tidak Didukung

Apabila server mendukung X-HTTP-Method-Override, fungsi built-in http_post_override barangkali dapat digunakan.

Sebagai contoh, misal ketika method PATCH diperlukan dan server mendukung X-HTTP-Method-Override.

JSON

Secara eksplisit, JSON tidak didukung di Singkong. Namun, HASH barangkali dapat digunakan.

Untuk melakukan konversi dari HASH ke STRING (misal ketika mengirimkan request), gunakanlah fungsi built-in string.

Untuk melakukan konversi dari STRING ke HASH (misal ketika menerima respon), gunakanlah fungsi built-in parse_hash.

Perhatikanlah contoh berikut:

```
>  var h = {"name": "Singkong"}
>  h["name"]
"Singkong"

>  var s = string(h)
>  s
"{"name": "Singkong"}"

>  var j = parse_hash(s)
>  j
{"name": "Singkong"}

>  type(s)
"STRING"

>  type(j)
"HASH"

>  type(parse_hash("", false))
"HASH"

>  type(parse_hash(""))
“NULL"
```

Base64

Encoding STRING ke Base64 dan decoding STRING dari Base64 dapat menggunakan fungsi-fungsi built-in base64_encode dan base64_decode. Sementara, untuk decoding/encoding Base64 pada file, gunakanlah fungsi-fungsi built-in x_base64_decode_file/x_base64_encode_file.

```
>  var e = base64_encode("Singkong")
>  e
"U2luZ2tvbmc="

>  var d = base64_decode(e)
>  d
```

```
"Singkong"
```

Fungsi tambahan

Fungsi built-in http_response_ok: untuk request HTTP yang berhasil, yang akan mengembalikan ARRAY (headers (HASH), response code (NUMBER), data (STRING)), apabila response status code adalah 200, maka data akan dikembalikan. Selain itu, NULL akan dikembalikan.

9. Bekerja dengan Thread

Konkurensi memungkinkan beberapa tugas dapat berjalan dalam waktu yang saling overlap, yang apabila dapat dilakukan dengan tepat, maka akan menjadikan program lebih nyaman digunakan.

Sebagai contoh, ketika sedang berkomunikasi dengan HTTP Server, sebuah program dapat menampilkan sejauh mana komunikasi tersebut telah dilakukan, dalam bentuk persentase atau progress bar.

Salah satu contoh implementasi konkurensi adalah multithreading, dimana sejumlah thread of control dapat dibuat. Dalam aplikasi, apabila dilakukan dengan tepat, dapat membuat aplikasi lebih responsif. Multithreading juga dapat berguna apabila aplikasi bekerja dengan operasi yang terkait I/O, dan pengguna perlu diberikan informasi progres.

Singkong mulai versi 5.0 menyediakan dukungan multithreading sederhana.

Membuat dan Menjalankan Thread

Untuk membuat dan menjalankan thread baru, gunakanlah fungsi bawaan thread. Fungsi ini membutuhkan sebuah argumen wajib berupa FUNCTION yang akan dijalankan pada thread tersebut. Thread yang dibuat akan segera dijalankan dan fungsi akan mengembalikan sebuah NUMBER yang merupakan thread id.

Sebagai perbandingan, berikut adalah contoh program single-threaded. Output program adalah [100000].

```
var a = [0]

var f = fn() {
```

```
        var i = 0
        repeat {
                set(a, 0, a[0] + 1)
                var i = i + 1
                if (i >= 100000) {
                        return i
                }
        }
}

f()
println(a)
```

Dan, berikut adalah contoh program yang melakukan tugas serupa, namun dilakukan oleh dua thread yang berbeda. Harapannya, output adalah [200000], karena fungsi serupa dijalankan oleh dua thread, namun pada kenyataannya, output bisa berbeda-beda setiap kali program dijalankan. Tentu bukan ini yang diharapkan.

```
var a = [0]

var f = fn() {
        var i = 0
        repeat {
                set(a, 0, a[0] + 1)
                var i = i + 1
                if (i >= 100000) {
                        return i
                }
        }
}

var t1 = thread(f)
var t2 = thread(f)
thread_join(t1)
```

```
thread_join(t2)

println(a)
```

Intrinsic Lock

Pada contoh sebelumnya, masing-masing thread, dalam perulangan sampai 100.000 kali, untuk setiap perulangan dilakukan, menambahkan satu ke nilai sebelumnya (variabel global yang sama). Penambahan demikian bukan merupakan operasi yang atomik dan thread switching dapat terjadi.

Di Singkong, kita dapat mengatur agar akses oleh beberapa thread dilakukan secara tersinkronisasi. Caranya, kita melewatkan argumen intrinsic lock yang akan digunakan, pada saat pembuatan thread.

Berikut adalah perbaikan dari contoh sebelumnya, dimana output adalah [200000].

```
var a = [0]

var f = fn() {
        var i = 0
        repeat {
                set(a, 0, a[0] + 1)
                var i = i + 1
                if (i >= 100000) {
                        return i
                }
        }
}

var t1 = thread(f, a)
var t2 = thread(f, a)
thread_join(t1)
```

```
thread_join(t2)

println(a)
```

Apabila diinginkan, untuk contoh tersebut, kita bisa saja menggunakan null, true, atau false sebagai lock, sebagaimana contoh berikut. Output akan tetap [200000].

```
var a = [0]

var f = fn() {
        var i = 0
        repeat {
                set(a, 0, a[0] + 1)
                var i = i + 1
                if (i >= 100000) {
                        return i
                }
        }
}

var t1 = thread(f, null)
var t2 = thread(f, null)
thread_join(t1)
thread_join(t2)

println(a)
```

Di sisi lain, pada contoh berikut, thread-thread tidaklah menggunakan lock yang sama:

```
var a = [0]

var f = fn() {
        var i = 0
        repeat {
                set(a, 0, a[0] + 1)
                var i = i + 1
                if (i >= 100000) {
                        return i
                }
        }
}

var t1 = thread(f, "Singkong")
var t2 = thread(f, "Singkong")
thread_join(t1)
thread_join(t2)

println(a)
```

Status Thread

Untuk mengetahui apakah sebuah thread masih berjalan, gunakanlah fungsi bawaan thread_alive. Fungsi ini membutuhkan argumen berupa thread id, yang dikembalikan dari pemanggilan fungsi thread.

Contoh penggunaan thread_alive akan disajikan bersama dengan simulasi proses yang berjalan lama di GUI.

Menunggu Thread Selesai

Untuk menunggu agar sebuah thread selesai dijalankan, gunakanlah fungsi bawaan thread_join. Fungsi ini membutuhkan argumen berupa thread id, yang dikembalikan dari pemanggilan fungsi thread.

Pada contoh-contoh sebelumnya, kita menggunakan thread_join untuk menunggu perhitungan oleh masing-masing thread selesai dilakukan.

Simulasi Proses yang Berjalan Lama (di GUI)

Pada pembahasan pengembangan aplikasi GUI, disertakan sebuah contoh bagaimana mengupdate komponen user interface untuk tugas yang dilakukan dari thread lain.

Contoh lebih sederhana berikut mensimulasikan proses yang berjalan lama, pada GUI.

```
var f = fn() {
  delay(random(5000, 10000))
}

var t = thread(f)

var s = number(@)

var c = fn() {
  if (thread_alive(t)) {
    var m = number(@) - s
    statusbar(0, m + " ms", true)
    statusbar(1, "busy", true)
  } else {
    statusbar(1, "idle", true)
    stop()
  }
}

reset()
timer(random(100, 500), c)
show()
```

10. Memanggil Method Java

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.

Walau demikian, terdapat beberapa keterbatasan berikut:

- Method yang dipanggil adalah method static dengan nama singkong
- Method singkong tersebut harus menerima sebuah String[][]
- Method singkong tersebut harus mengembalikan salah satu dari:
  - String
  - String[]
  - String[][]
- Sebagaimana terlihat, kita hanya bekerja dengan STRING. Singkong telah menyediakan sejumlah fungsi built-in untuk bekerja dengan STRING ataupun konversi ke tipe lain.
- Dengan fungsi built-in eval, String yang dikembalikan dari pemanggilan method Java dapat dievaluasi. Ini artinya, method Java tersebut dapat membuat kode program Singkong secara dinamis dan eval akan mengevaluasi kode tersebut.

Sama seperti pada pengembangan aplikasi database (dengan JDBC driver selain Apache Derby dan PostgreSQL), kita perlu memberitahu Java Runtime Environment ke mana harus mencari class yang berisi method tersebut.

Oleh karena itu, apabila dijalankan secara standalone, kita perlu menjalankan interpreter Singkong dengan cara berikut. Perintah diketikan dalam baris yang sama. Sesuaikan pemisah class path, apakah sistem yang Anda gunakan menggunakan : (macOS atau Linux) atau ; (Windows).

```
java -cp Singkong.jar:<jar>
com.noprianto.singkong.Singkong

java -cp Singkong.jar:<dir>
com.noprianto.singkong.Singkong

java -cp Singkong.jar;<jar>
com.noprianto.singkong.Singkong

java -cp Singkong.jar;<dir>
com.noprianto.singkong.Singkong
```

Dalam contoh perintah tersebut, <jar> merujuk pada file jar itu sendiri dan <dir> merujuk pada direktori berisikan file class. Untuk merujuk pada direktori aktif, umumnya karakter titik digunakan (tidak diperlukan apabila Anda menggunakan file jar).

Catatan: bacalah juga pembahasan tentang deployment, terutama ketika bundel file-file class bersama interpreter Singkong diperlukan.

Untuk memanggil method yang ditulis dalam Java, gunakanlah fungsi built-in call. Fungsi ini menerima dua argumen: STRING (nama class Java yang dapat ditemukan di class path) dan tipe apa saja. Fungsi ini mengembalikan STRING, ARRAY (dari STRING), ARRAY (dari ARRAY (dari STRING)), atau NULL (ketika terjadi kesalahan).

Untuk mendapatkan informasi tentang method Java yang dapat dipanggil dari Singkong (apabila informasi memang disediakan), gunakanlah fungsi built-in call_info. Fungsi ini menerima argumen berupa STRING (nama class Java yang dapat ditemukan di class path) dan mengembalikan STRING. Method Java yang menyediakan informasi tersebut haruslah berupa method static dengan nama singkong_info, tanpa parameter, dan mengembalikan String.

Argumen Method

Sebagaimana dibahas sebelumnya, argumen terakhir untuk fungsi built-in call dapat berupa tipe apa saja. Nilai ini akan dilewatkan ketika method Java dipanggil. Dan, kita tahu bahwa method Java ini menerima String[][].

Ini disediakan untuk kenyamanan programmer Singkong, dimana interpreter Singkong akan melakukan hal berikut:

- Representasi STRING dari nilai yang dilewatkan akan digunakan, apabila memang diperlukan (misal ketika argumen berupa NUMBER dilewatkan ke fungsi call). Singkong tidak akan melewatkan null ke pemanggilan method Java. Dengan demikian, pengembang method Java tersebut bisa yakin bahwa argumen yang diterima adalah String[][] yang setidaknya berisi satu String[], yang setidaknya mengandung satu elemen String.
- Apabila yang dilewatkan bukanlah ARRAY:
  - Sebuah String[1][] akan dibuat
  - Elemen pertama dari array adalah String[1], yang berisi elemen tunggal berupa representasi STRING nilai yang dilewatkan tersebut.
- Apabila yang dilewatkan adalah ARRAY:
  - Sebuah String[][] akan dibuat, dengan panjang array adalah panjang ARRAY yang dilewatkan

- Untuk setiap elemen dalam ARRAY:
  - Apabila bukan merupakan ARRAY:
    - String[1] akan dibuat, berisikan representasi STRING elemen tersebut
  - Apabila merupakan ARRAY:
    - String[] akan dibuat, dengan panjang array adalah panjang elemen ARRAY tersebut
    - String[] yang dibuat ini akan dipopulasikan dengan representasi STRING untuk setiap elemen dalam ARRAY tersebut

Nilai Kembalian

Apabila terdapat exception ketika method dipanggil, interpreter Singkong akan mengembalikan NULL untuk fungsi built-in call.

Apabila method Java mengembalikan selain String, String[], atau String[][], NULL akan dikembalikan.

Kesalahan lain yang mungkin terjadi juga akan menyebabkan NULL dikembalikan.

Contoh: String

HelloWorld.java yang tersimpan di direktori examples:

```
public class HelloWorld {
    public static String singkong_info() {
        return "HelloWorld: Description, license, ...";
    }

    public static String singkong(String[][] args) {
        StringBuilder builder = new StringBuilder();
        for (int i=0; i<args.length; i++) {
            String[] r = args[i];
            for (int j=0; j<r.length; j++) {
                String s = r[j];
                builder.append(s);
```

```
                    builder.append(", ");
                }
                builder.append("; ");
            }
            return builder.toString();
        }

        public static void main(String[] args) {
            System.out.println("HelloWorld: Singkong
Programming Language module");
        }
    }
```

File ini akan dikompilasi menjadi HelloWorld.class.

```
cd examples
javac HelloWorld.java
cd ..
```

Karena HelloWorld.class disimpan di dalam direktori examples relatif dari direktori aktif, kita menjalankan Singkong dengan cara demikian. Sistem operasi yang digunakan adalah macOS sehingga pemisah class path adalah titik dua (:). Perintah diketikkan dalam baris yang sama.

```
java -cp Singkong.jar:examples
com.noprianto.singkong.Singkong
```

Sebagai alternatif, apabila ingin membundel file HelloWorld.class (atau file class lainnya) bersama Singkong.jar:

- Kopikanlah HelloWorld.class ke direktori aktif
- Jalankanlah perintah berikut:

```
jar uf Singkong.jar HelloWorld.class
```

(Bacalah juga pembahasan tentang deployment)

Program Singkong selanjutnya dapat memanggil method tersebut, yang akan mengembalikan STRING:

```
    >  var x = call("HelloWorld", [[], [1],
[2,3], [4,5,6]])
    >  x
    "; 1, ; 2, 3, ; 4, 5, 6, ; "

    >  type(x)
    "STRING"
```

Contoh: String[]

HelloWorldArray.java yang tersimpan di direktori examples:

```
   public class HelloWorldArray {
       public static String singkong_info() {
           return "HelloWorldArray: Description,
license, ...";
       }

       public static String[] singkong(String[][] args) {
           String[] ret = new String[args.length];
           for (int i=0; i<args.length; i++) {
               StringBuilder builder = new StringBuilder();
               String[] r = args[i];
               for (int j=0; j<r.length; j++) {
                   String s = r[j];
                   builder.append(s);
                   builder.append(", ");
               }
               ret[i] = builder.toString();
           }
           return ret;
       }

       public static void main(String[] args) {
           System.out.println("HelloWorldArray: Singkong
Programming Language module");
```

```
        }
    }
```

File ini akan dikompilasi menjadi HelloWorldArray.class.

```
cd examples
javac HelloWorldArray.java
cd ..
```

Menjalankan interpreter Singkong (sesuaikanlah pemisah class path dengan sistem operasi yang Anda gunakan). Perintah diketikkan dalam baris yang sama.

```
java -cp Singkong.jar:examples
com.noprianto.singkong.Singkong
```

Program Singkong selanjutnya dapat memanggil method tersebut, yang akan mengembalikan ARRAY (dari STRING):

```
> var x = call("HelloWorldArray", [[],
[1], [2,3], [4,5,6]])
> x
["", "1, ", "2, 3, ", "4, 5, 6, "]

> type(x)
"ARRAY"

> each(x, fn(x, y){ println(y + ": " +
type(x) + ": " + x)})
0: STRING:
1: STRING: 1,
```

```
2: STRING: 2, 3,
3: STRING: 4, 5, 6,
```

Contoh: String[][]

HelloWorldArrayArray.java yang tersimpan di direktori examples:

```
public class HelloWorldArrayArray {
    public static String singkong_info() {
        return "HelloWorldArrayArray: Description, license,
...";
    }

    public static String[][] singkong(String[][] args) {
        String[][] ret = new String[args.length][];
        for (int i=0; i<args.length; i++) {
            String[] r = args[i];
            String[] x = new String[r.length];
            for (int j=0; j<r.length; j++) {
                String s = r[j];
                x[j] = s;
            }
            ret[i] = x;
        }
        return ret;
    }

    public static void main(String[] args) {
        System.out.println("HelloWorldArrayArray: Singkong
Programming Language module");
    }
}
```

File ini akan dikompilasi menjadi HelloWorldArrayArray.class.

```
cd examples
```

```
javac HelloWorldArrayArray.java
cd ..
```

Menjalankan interpreter Singkong (sesuaikanlah pemisah class path dengan sistem operasi yang Anda gunakan). Perintah diketikkan dalam baris yang sama.

```
java -cp Singkong.jar:examples
com.noprianto.singkong.Singkong
```

Program Singkong selanjutnya dapat memanggil method tersebut, yang akan mengembalikan ARRAY (dari ARRAY (dari STRING)):

```
    >  var x = call("HelloWorldArrayArray",
[[], [1], [2,3], [4,5,6]])
    >  x
    [[], ["1"], ["2", "3"], ["4", "5", "6"]]

    >  each(x, fn(x, y){ println(y + ": " +
type(x) + ": " + x)})
    0: ARRAY: []
    1: ARRAY: ["1"]
    2: ARRAY: ["2", "3"]
    3: ARRAY: ["4", "5", "6"]
```

Contoh: String (eval)

HelloWorldEval.java yang tersimpan di direktori examples:

```
public class HelloWorldEval {
```

```
        public static String singkong_info() {
            return "HelloWorldEval: Description, license, ...";
        }

        public static String singkong(String[][] args) {
            StringBuilder builder = new StringBuilder();
            for (int i=0; i<args.length; i++) {
                String[] r = args[i];
                for (int j=0; j<r.length; j++) {
                    String s = r[j];
                    builder.append(s);
                    builder.append(", ");
                }
                builder.append("; ");
            }

            return String.format("component(\"button\",
\"%s\")",
                builder.toString());
        }

        public static void main(String[] args) {
            System.out.println("HelloWorldEval: Singkong
Programming Language module");
        }
    }
```

Setelah kompilasi dilakukan dan interpreter Singkong dijalankan, program Singkong dapat memanggil method tersebut seperti contoh berikut:

```
    >  var code = call("HelloWorldEval", [[],
[1], [2,3], [4,5,6]])
    >  code
    "component("button", "; 1, ; 2, 3, ; 4, 5,
6, ; ")"

    >  var b = eval(code)
    >  b
    COMPONENT: button (; 1, ; 2, 3, ; 4, 5, 6,
; )
```

```
> add(b)


> show()
```

Contoh: Informasi

Berikut adalah contoh-contoh penggunaan fungsi built-in call_info:

```
> call_info("HelloWorld")
"HelloWorld: Description, license, ..."

> call_info("HelloWorldArray")
"HelloWorldArray: Description,
license, ..."

> call_info("HelloWorldArrayArray")
"HelloWorldArrayArray: Description,
license, ..."

> call_info("HelloWorldEval")
"HelloWorldEval: Description,
license, ..."
```

Contoh: Dialog

Catatan: untuk dialog custom yang lebih sederhana (namun lebih mudah dibuat), yang dapat dibuat hanya dengan kode Singkong, gunakanlah fungsi built-in panel_dialog.

Dialog.java (disimpan di dalam direktori examples, dikompilasi ke Dialog.class: javac Dialog.java)

```
    import java.awt.BorderLayout;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import javax.swing.JButton;
    import javax.swing.JDialog;
    import javax.swing.JTextField;

    public class Dialog extends JDialog implements
ActionListener{
        private String value;
        private String param;
        private JTextField text;
        private JButton button;

        public Dialog(String param) {
            setModal(true);

            value = "";
            this.param = param;

            text = new JTextField();
            button = new JButton(param);
            button.addActionListener(this);

            add(text);
            add(button, BorderLayout.SOUTH);
            pack();
        }

        public String getValue() {
            setVisible(true);
            return value;
        }
```

```
        public void actionPerformed(ActionEvent e) {
            value = text.getText();
            setVisible(false);
            dispose();
        }

        public static String singkong_info() {
            return "Dialog: Custom dialog in Java";
        }

        public static String singkong(String[][] args) {
            String param = "";
            if (args.length > 0) {
                String[] r = args[0];
                if (r.length > 0) {
                    param = r[0];
                }
            }
            Dialog dialog = new Dialog(param);
            return dialog.getValue();
        }

        public static void main(String[] args) {
            System.out.println("Dialog: Singkong Programming
Language module");
        }
    }
```

Menjalankan interpreter Singkong (sesuaikanlah pemisah class path dengan sistem operasi yang Anda gunakan). Perintah diketikkan dalam baris yang sama.

```
java -cp Singkong.jar:examples
com.noprianto.singkong.Singkong
```

Program Singkong selanjutnya dapat memanggil method tersebut, yang akan mengembalikan STRING:

```
> var value = call("Dialog", "Singkong")

(ketikkan Hello World pada text field di
dialog)
(klik tombol Singkong)

> value
"Hello World"
```

11. Embedding Singkong

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.

Class yang digunakan, sama seperti pembahasan topik sebelumnya, adalah main class:

```
com.noprianto.singkong.Singkong
```

Method yang dapat digunakan adalah sebagai berikut:

Method	Deskripsi
public static java.lang.String evaluatorString(java.lang.String)	Interpretasi kode, ouput dikembalikan sebagai String. Cara yang disarankan apabila tidak membutuhkan melewatkan nilai dari program Java ke interpreter Singkong, dengan semua fungsi built-in tersedia.
public static java.lang.String evaluatorString(java.lang.String, java.lang.String[])	Interpretasi kode, ouput dikembalikan sebagai String. Cara yang disarankan apabila tidak membutuhkan melewatkan nilai dari program Java ke interpreter Singkong, namun ingin mendisable fungsi built-in tertentu.

Method	Deskripsi
public static java.lang.String evaluatorString(java.lang.String, com.noprianto.singkong.SingkongEnvironment)	Interpretasi kode, ouput dikembalikan sebagai String. Cara yang disarankan apabila ingin melewatkan nilai dari program Java ke interpreter Singkong, dengan semua fungsi built-in tersedia.
public static java.lang.String evaluatorString(java.lang.String, com.noprianto.singkong.SingkongEnvironment,java.lang.String[])	Interpretasi kode, ouput dikembalikan sebagai String. Cara yang disarankan apabila ingin melewatkan nilai dari program Java ke interpreter Singkong, dan ingin mendisable fungsi built-in tertentu.
public static void evaluatorString(java.lang.String, com.noprianto.singkong.SingkongEnvironment,java.io.PrintStream)	Interpretasi kode dengan output PrintStream tersendiri. Dapat digunakan apabila ingin melewatkan nilai dari program Java ke interpreter Singkong, dengan semua fungsi built-in tersedia.
public static void evaluatorString(java.lang.String, com.noprianto.singkong.SingkongEnvironment,java.io.PrintStream, java.lang.String[])	Interpretasi kode dengan output PrintStream tersendiri. Dapat digunakan apabila ingin melewatkan nilai dari program Java ke interpreter Singkong, dan ingin mendisable fungsi built-in tertentu.

Untuk melewatkan nilai dari program Java ke Singkong, sebagai variabel di Singkong, kita bisa simpan semua nilai tersebut dalam pemetaan Map<String, Object>, dimana key adalah nama variabel.

Untuk melakukan konversi dari Map ke SingkongEnvironment, gunakanlah method:

```
public static
com.noprianto.singkong.SingkongEnvironment
environmentFromMap(java.util.Map)
```

Catatan: pemanggilan System.exit() secara eksplisit mungkin diperlukan untuk terminasi aplikasi.

Interpretasi

Isi file Test.java

```
import com.noprianto.singkong.Singkong;


public class Test {
    public static void main(String[] args) {
        String code = "var list = [1,2,3] println(list)";
        String output = Singkong.evaluatorString(code);
        System.out.println(output);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
[1, 2, 3]
```

Interpretasi, Built-in

Isi file Test.java

```
import com.noprianto.singkong.Singkong;


public class Test {
    public static void main(String[] args) {
        String code = "println([1,2,3]) system()";
        String output = Singkong.evaluatorString(code, new
String[]{"system"});
        System.out.println(output);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
[1, 2, 3]
ERROR: [line: 1] built-in function "system" is
disabled
```

Interpretasi, Environment

Isi file Test.java

```
import java.util.Map;
import java.util.HashMap;
import com.noprianto.singkong.Singkong;

public class Test {
    public static void main(String[] args) {
        Map<String, Object> map = new HashMap<String,
Object>();
        map.put("hello", "Hello, World");
        map.put("test", true);
```

```
        String code = "println(hello) println(test)";
        String output = Singkong.evaluatorString(code,
            Singkong.environmentFromMap(map));

        System.out.println(output);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
Hello, World
true
```

Interpretasi, Environment, Built-in

Isi file Test.java

```
import java.util.Map;
import java.util.HashMap;
import com.noprianto.singkong.Singkong;

public class Test {
    public static void main(String[] args) {
        Map<String, Object> map = new HashMap<String,
Object>();
        map.put("hello", "Hello, World");

        String code = "println(hello) info()";
        String output = Singkong.evaluatorString(code,
            Singkong.environmentFromMap(map),
            new String[]{"system", "info"});

        System.out.println(output);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
Hello, World
ERROR: [line: 1] built-in function "info" is
disabled
```

Interpretasi, Environment, PrintStream

Isi file Test.java

```
import java.io.ByteArrayOutputStream;
import java.io.PrintStream;
import java.util.HashMap;
import java.util.Map;
import com.noprianto.singkong.Singkong;

public class Test {
    public static void main(String[] args) {
        String result = "";

        Map<String, Object> map = new HashMap<String,
Object>();
        map.put("hello", "Hello, World");
        map.put("test", true);

        ByteArrayOutputStream outputStream = new
ByteArrayOutputStream();
        try {
            PrintStream output = new PrintStream(outputStream);
            Singkong.evaluatorString("println(hello)
println(test)",
                    Singkong.environmentFromMap(map), output);
            result = outputStream.toString();
        } catch (Exception e) {
        }
```

```
        System.out.println(result);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
Hello, World
true
```

Interpretasi, Environment, Built-in, PrintStream

Isi file Test.java

```
import java.io.ByteArrayOutputStream;
import java.io.PrintStream;
import java.util.HashMap;
import java.util.Map;
import com.noprianto.singkong.Singkong;

public class Test {
    public static void main(String[] args) {
        String result = "";

        Map<String, Object> map = new HashMap<String,
Object>();
        map.put("hello", "Hello, World");

        ByteArrayOutputStream outputStream = new
ByteArrayOutputStream();
        try {
            PrintStream output = new PrintStream(outputStream);
            Singkong.evaluatorString("println(hello) info()",
                    Singkong.environmentFromMap(map), output,
                    new String[]{"system", "info"});
            result = outputStream.toString();
        } catch (Exception e) {
        }
```

```
        System.out.println(result);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
Hello, World
ERROR: [line: 1] built-in function "info" is
disabled
```

Contoh: Bahasa Pemrograman Lain

Bukan merupakan embedding Singkong yang sesungguhnya, namun barangkali bisa berguna.

Python dengan semua fungsi built-in tersedia:

```
>>> import subprocess
>>> c = ['java', '-jar', 'Singkong.jar',
'println("singkong")']
>>> o = subprocess.check_output(c)
>>> print(o)
singkong
```

Python dengan fungsi built-in tertentu dinonaktifkan:

```
>>> import subprocess
>>> c = ['java', '-DDISABLE=info', '-jar',
'Singkong.jar', 'info()']
>>> o = subprocess.check_output(c)
>>> print(o)
ERROR: [line: 1] built-in function "info" is
disabled
```

12. Deployment

Setelah aplikasi selesai dikembangkan dalam bahasa pemrograman Singkong, kita barangkali ingin menjalankannya di komputer-komputer lain. Dan terdapat kemungkinan bukan kita selaku pengembang aplikasi yang melakukan instalasi dan menjalankannya sendiri. Sementara, pengguna aplikasi kita bisa saja belum terbiasa dengan instalasi program, menjalankan command line/terminal emulator, dan lainnya.

Dalam bentuk yang paling mempermudah pengguna, secara *teknis* tentu saja Anda dapat membundle aplikasi Anda bersama dengan Java Runtime Environment dan interpreter Singkong. Anda bahkan dapat memaketkannya sebagai sebuah installer. Namun, hal-hal ini diluar cakupan dokumentasi Singkong dan buku ini.

Java Runtime Environment umum ditemukan terinstal di berbagai sistem, dan inilah yang akan kita asumsikan. Interpreter Singkong telah diusahakan agar hanya membutuhkan Java 5.0 atau yang lebih baru (serta telah diuji berjalan pada Java versi terbaru pada saat buku ini ditulis). Setidaknya, aplikasi yang Anda kembangkan dengan Singkong berpotensi untuk berjalan pada berbagai sistem operasi, bahkan untuk yang dirilis sebelum tahun 2000. Selama komputer pengguna telah terinstal Java, besar kemungkinan, interpreter Singkong dan aplikasi Anda akan dapat dijalankan.

Sebelum Singkong versi 3.5 dirilis, pengguna aplikasi Anda perlu mendownload Singkong.jar dan file aplikasi yang Anda kembangkan (*.singkong), kemudian menjalankan aplikasi tersebut dengan beberapa opsi berikut (yang tidak lagi disarankan pada saat buku ini ditulis).

1. Menjalankan Singkong.jar, membuka file, menjalankan aplikasi (dimana GUI diasumsikan tersedia)

```
java –jar Singkong.jar
```

(Ataupun klik ganda pada Singkong.jar apabila sistem Anda mendukung)

Aktiflah di tab Editor dan buka file aplikasi dengan klik pada tombol Open. Setelah itu, jalankanlah aplikasi dengan klik pada tombol Run.

Catatan: opsi ini membutuhkan beberapa langkah dan pengguna diharapkan mengetahui path file aplikasi (menggunakan dialog file). Ini menjadi tidak praktis dan terlalu teknis.

Selain itu, apabila klik ganda tidak bekerja atau tidak didukung, maka opsi ini mengharuskan pengguna untuk mengakses command line atau terminal emulator, yang mana akan menambahkan langkah yang diperlukan.

2. Menjalankan Singkong.jar dengan argumen

```
java –jar Singkong.jar <file>
```

Catatan: opsi ini mengharuskan pengguna untuk mengakses command line atau terminal emulator. Selain itu, pengguna juga diharapkan mengetahui path file aplikasi. Ini mungkin menjadi terlalu teknis.

3. Embedding interpreter Singkong

Catatan: opsi ini membutuhkan penulisan kode program, sebagai contoh, dalam bahasa Java.

Sejak Singkong versi 3.5 dirilis, opsi berikut tersedia.

4. Membundel file aplikasi dengan interpreter Singkong

Opsi ini akan menghasilkan satu file jar yang dapat dijalankan, yang menjadikannya lebih mudah dalam distribusi aplikasi Anda.

Lakukanlah langkah-langkah berikut untuk membundel file aplikasi dengan interpreter Singkong:
- Nama file aplikasi haruslah diganti menjadi, atau dikopikan sebagai: main.singkong
- File tersebut harus ditambahkan ke Singkong.jar dengan program yang kompatibel dengan format jar/zip. Sebagai contoh: jar

```
jar uf Singkong.jar main.singkong
```

- Singkong.jar kemudian harus diganti menjadi nama lain, yang tidak mengandung kata “Singkong”.

- Sebagai langkah opsional: package-package lain (class Java) dapat pula ditambahkan bersama file aplikasi Anda.

- Juga merupakan langkah opsional: untuk menambahkan file-file resource (misal: file gambar, suara, atau kode Singkong) ke dalam file jar interpreter, sehingga memudahkan distribusi aplikasi (cukup sebuah file jar saja):
  - Tambahkanlah semua file tersebut di direktori /resource dalam file jar interpreter. Sebagai contoh: /resource/file.png dan /resource/test.txt.
  - Bacalah juga tentang Pengembangan Aplikasi GUI, dimana dicontohkan cara menggunakan resource yang terbundel.
  - Untuk mengopikan sebuah resource ke file, gunakanlah fungsi built-in copy_resource.
  - Untuk mengevaluasi/menjalankan kode program Singkong yang tersimpan sebagai resource, gunakanlah fungsi built-in load_resource.

- Untuk mengevaluasi/menjalankan kode program Singkong (file di direktori aktif atau (jika gagal dibaca), dari /resource di file interpreter), gunakanlah fungsi built-in load_file_or_resource.

13. Bekerja dengan file Comma-Separated Values

File Comma-Separated Values (CSV) umum digunakan, diantaranya sebagai format data exchange. Berbagai program spreadsheet juga dapat membuka dan menyimpan ke file CSV. Kita dapat membuka dan mengedit file CSV dengan editor teks.

Selain karakter koma, untuk pemisah/separator field, karakter lain juga dapat digunakan. Apabila sebuah nilai mengandung karakter pemisah tersebut, maka nilai tersebut perlu dituliskan dalam kutip. Sebuah nilai dapat pula terdiri dari sejumlah baris (embedded newline).

Sejak Singkong versi 4.6, disertakan modul bawaan, ditulis sepenuhnya dengan Singkong, dengan nama **csv**, yang dapat digunakan untuk:

- Membaca dari STRING (berisikan data CSV) ke ARRAY, baik dengan separator berupa koma ataupun lainnya.
- Dari ARRAY ke STRING CSV yang siap disimpan ke file. Sama seperti ketika membaca, karakter separator juga bisa ditentukan.
- Mengaplikasikan fungsi tertentu untuk setiap nilai (misal: mengubah dari STRING ke NUMBER), dalam setiap baris, dalam ARRAY (yang didapatkan dari STRING CSV).

Berikut adalah contoh singkat penggunaan modul tersebut.

Isi file test.csv dalam direktori aktif:

```
id,name
1,"Hello,World"
2,Singkong
```

Pertama, load-lah terlebih dahulu modul csv:

```
> load_module(“csv")
```

Setelah modul ini di-load, sejumlah fungsi berikut akan tersedia:

- CSV_FROM_STRING
- CSV_FROM_STRING_DEFAULT
- CSV_TO_STRING
- CSV_TO_STRING_DEFAULT
- CSV_FUNCTIONS

Membaca file test.csv dan mendapatkan ARRAY dari STRING CSV tersebut:

```
> var a = csv_from_string(",",
read("test.csv"))
> a
[["id", "name"], ["1", "Hello,World"], ["2",
"Singkong"], []]
```

Dari ARRAY ke STRING CSV, kemudian menulis ke file output.csv, dengan separator berupa titik koma:

```
> write("output.csv", csv_to_string(";", a))
true
```

Mengubah STRING CSV dengan separator berupa titik koma:

```
> var b = csv_from_string(";",
read("output.csv"))
> b
[["id", "name"], ["1", "Hello,World"], ["2",
“Singkong"]]
```

Mengaplikasikan fungsi:

```
    >  var c = csv_functions(slice(b, 1, len(b)),
[number])
    >  c
    [[1, "Hello,World"], [2, “Singkong"]]
```

Perhatikanlah bahwa kutip akan ditambahkan secara otomatis, apabila diperlukan. Nilai “Hello,World” mengandung koma. Apabila koma digunakan sebagai separator, maka “Hello,World” tersebut perlu dituliskan dalam kutip. Berbeda misal dengan “Singkong”, yang dapat dituliskan apa adanya sebagai Singkong.

```
    >  b
    [["id", "name"], ["1", "Hello,World"], ["2",
"Singkong"]]

    >  write("output-test.csv",
csv_to_string_default(b))
    true

    >  print(read("output-test.csv"))
    id,name
    1,"Hello,World"
    2,Singkong
```

Embedded newline juga dapat digunakan:

```
    >  write("out.csv", csv_to_string_default([[1,
"singkong" + newline() + "programming language"]]))
    true

    >  print(read("out.csv"))
    1,"singkong
    programming language"

    >  var d =
csv_from_string_default(read("out.csv"))
    >  d
```

```
[["1", "singkong
programming language"]]

>  rect_array_size(d)
[1, 2]
```

Untuk saat ini, kutip ganda yang di-embed, belum didukung.

14. Perbedaan dengan Bahasa Monkey

Bahasa Pemrograman Singkong berbasiskan pada Monkey.java, dimana Monkey.java sendiri berbasiskan pada monkey.py (Python), dan monkey.py berbasiskan pada kode dalam bahasa Go, dalam buku WRITING AN INTERPRETER IN GO.

Penulis mengembangkan monkey.py dan Monkey.java karena lebih terbiasa dengan Java (dan Python), sebagai latihan dalam mempelajari buku tersebut. Monkey.java dan monkey.py dilisensikan free/open source dan dapat didownload dari website penulis.

Seiring perkembangan, Singkong menemukan jalannya sendiri dan walaupun secara sintaks masih mirip, terdapat sejumlah perbedaan yang perlu dibahas. Source code Singkong pada saat buku ini ditulis berukuran lebih dari 11 kali Monkey.java, dalam sekitar 33.000 baris kode Java dan Singkong.

Walau demikian, tanpa Monkey, tidak terbayangkan akan adanya Singkong.

Tersedia pula contoh implementasi Monkey dengan Singkong (dalam sekitar 2.100 baris kode Singkong, free/open source), yang dapat didownload dari:

- monkey.singkong: https://nopri.github.io/monkey.singkong
- monkey.singkong runnable jar (bundel dengan interpreter Singkong): https://nopri.github.io/monkeyinterpreter.jar

Berikut adalah perbedaan antara Singkong dan Monkey:

- Singkong tidak membedakan huruf besar dan huruf kecil (case-insensitive)
- Assignment dilakukan dengan statement var (let tetap didukung, untuk kompatibilitas)
- Nama variabel Singkong dimulai dengan huruf atau underscore dan secara opsional dapat diikuti dengan huruf, bilangan, dan underscore
- Singkong menyediakan literal null
- Nilai null tidak dicetak atau ditampilkan. Untuk mencetaknya, gunakanlah fungsi print, println, puts, atau message.
- Singkong tidak mengenal tipe data INTEGER. Singkong memiliki tipe data NUMBER yang kompatibel dengan INTEGER, dengan dukungan akan bilangan desimal.
- Singkong menyediakan operator tambahan berikut untuk NUMBER: % ^ <= >=
- Singkong menyediakan operator tambahan berikut untuk BOOLEAN: & |
- Di Singkong, key dan value HASH dapat berupa tipe apa saja. HASH juga terurut sesuai waktu pemetaan ditambahkan.
- Singkong menyediakan 339 fungsi built-in dan 6 modul built-in pada saat buku ini ditulis.
- Singkong datang dengan tipe data tambahan:
  - DATE (tanggal dan waktu)
  - COMPONENT (untuk pengembangan aplikasi GUI)
  - DATABASE (koneksi database, untuk pengembangan aplikasi database)
- Singkong menggunakan operator semaksimal mungkin ketika bekerja dengan berbagai tipe data. Sebagai contoh, * untuk perulangan STRING, - untuk menghapus karakter dalam STRING, - untuk menghapus elemen dalam ARRAY, dan lainnya.
- Perulangan dengan repeat dan fungsi built-in: do, each
- Sejumlah fungsi built-in dapat dinonaktifkan ketika interpreter Singkong dijalankan

- Komentar pada source code diawali dengan # dan diakhiri dengan ; (titik koma). Komentar dapat dituliskan lebih dari satu baris.
- Singkong mendukung documentation string pada FUNCTION.

Halaman ini sengaja dikosongkan

Referensi

Dokumentasi Bahasa Pemrograman Singkong:
- Disertakan dalam Singkong.jar
- Atau, tersedia online: https://nopri.github.io/Singkong.txt

Bahasa pemrograman Singkong merupakan bahasa pemrograman yang case-insensitive (tidak membedakan huruf besar/kecil), dynamically typed (tipe data ditentukan secara dinamis pada saat program berjalan), prosedural, dan interpreted, yang berjalan pada Java Virtual Machine (Java 5.0 atau lebih baru).

Singkong mendukung tipe data number, boolean, string, array, hash, date, function (first-class function), component (GUI), dan database. Untuk memudahkan pemrograman, Singkong juga datang dengan 339 built-in function (fungsi bawaan) dan 6 built-in module (modul bawaan). Singkong dapat digunakan untuk mengembangkan berbagai jenis aplikasi yang dapat dilengkapi dengan Graphical User Interface dan terhubung ke berbagai sistem database relasional. Aplikasi yang dikembangkan tersebut dapat dijalankan pada berbagai sistem operasi dimana Java tersedia.

Singkong juga dapat di-embed ke dalam aplikasi lain (misal untuk kebutuhan scripting sederhana) dan dapat memanggil method Java (yang menyediakan fungsionalitas tambahan).

Dengan dirancang hanya membutuhkan Java 5.0 (yang dirilis pada tahun 2004, 15 tahun sebelum Singkong dikembangkan), namun dapat berjalan pada Java versi terbaru (pada saat buku ini ditulis), Singkong diharapkan dapat digunakan oleh siapa pun, dengan komputer apapun, termasuk untuk mempelajari pemrograman.

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

Dr. Noprianto mengembangkan bahasa pemrograman Singkong dan interpreternya sejak akhir 2019. Beliau mendirikan/mengelola Singkong.dev (PT. Stabil Standar Sinergi), menyukai pemrograman, dan telah menulis beberapa buku cetak, termasuk Python, Java, dan Singkong. Buku dan softwarenya dapat didownload dari https://nopri.github.io

---

ISBN 978-602-52770-1-6

Penerbit:
PT. Stabil Standar Sinergi
Download gratis buku ini:
https://nopri.github.io

9 786025 277016