Mengenal dan Menggunakan Bahasa Pemrograman Singkong

Dr. Noprianto

Penerbit:
PT. Stabil Standar Sinergi
Download gratis buku ini:
http://noprianto.com

ISBN 978-602-52770-1-6

9 786025 277016

Mengenal dan Menggunakan Bahasa Pemrograman Singkong

Penulis: Dr. Noprianto

ISBN: 978-602-52770-1-6

Editor: Dr. Noprianto

Penerbit:
PT. Stabil Standar Sinergi

Redaksi:
Website: http://stabilstandar.com
Email: info@stabilstandar.com

Cetakan 1: Januari 2020

Tentang Penulis

Dr. Noprianto menyukai pemrograman, memiliki sertifikasi Java (OCP), dan telah menulis beberapa buku cetak: Python (2002), Debian GNU/Linux (2006), OpenOffice.org (2006), Java (2018), dan Singkong (2020). Beliau juga menulis beberapa buku elektronik gratis: wxWidgets (2006), Python (2009), SQLiteBoy (2014), dan OpenERP (rekan penulis, 2014).

Noprianto lulus dari jurusan teknik informatika (2003), manajemen sistem informasi (2015), dan doktor ilmu komputer (2019) dari Universitas Bina Nusantara.

Noprianto mengembangkan bahasa pemrograman Singkong dan interpreternya sejak 2019.

Buku dan softwarenya dapat didownload dari noprianto.com

Halaman ini sengaja dikosongkan

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 lebih dari 150 builtin function (fungsi 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, Januari 2020.

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 let seperti contoh berikut:

```
let 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:

```
let 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, 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/editor 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.

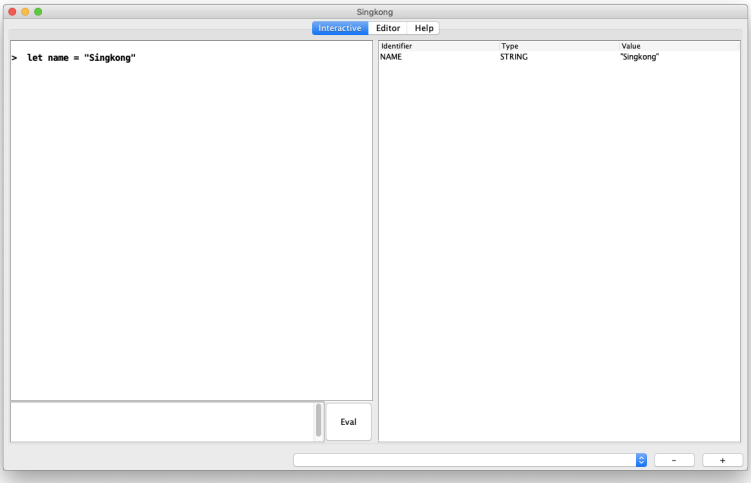
Pada interactive evaluator, nilai dari suatu ekspresi akan langsung ditampilkan, sehingga kita tidak perlu

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 memberikan nilai pada 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.



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

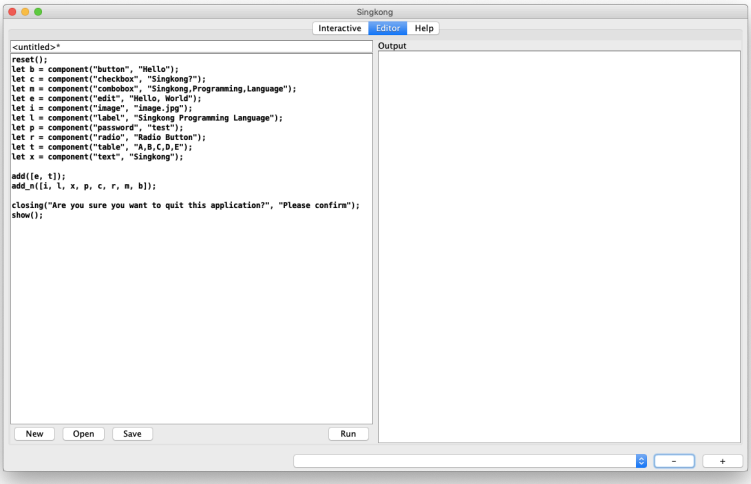
Editor

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.

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.

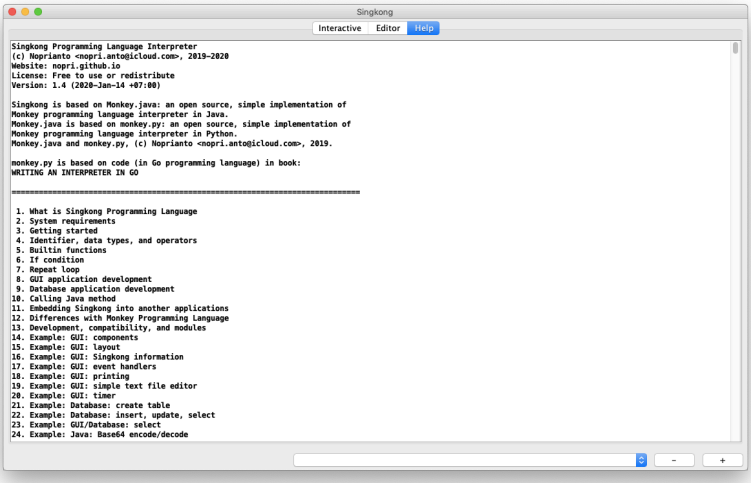


Membaca Dokumentasi

Di dalam file Singkong.jar, sebuah 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
builtin 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 dari website penulis. Pada saat buku ini ditulis, Singkong.jar berukuran lebih besar sedikit dari 200 KB.

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, 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 fungsi bawaan sengaja dinonaktifkan. 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, 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 (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
```

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 akses pada sistem database relasional atau 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
```

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

Halaman ini sengaja dikosongkan

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 let. 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:

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

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

```
let 1x = 12345
PARSER ERROR: expected next token to be IDENT, got INT
instead
```

Pesan kesalahan tersebut dapat diartikan bahwa setelah kata kunci let, nama identifier (variabel) yang valid diharapkan. Namun, justru sebuah INT (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:

```
let info = ""
ERROR: info is a builtin function
```

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

Akhir Statement dan Penanda Blok

Di Singkong, apabila diperlukan, misal ketika ingin mengetikkan lebih dari satu statement pada baris yang sama, akhirilah sebuah statement dengan sebuah titik koma (;).

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

Fungsi Builtin Terkait

Fungsi builtin (bawaan) akan kita bahas tersendiri. Namun, beberapa fungsi berikut terkait dengan tipe data, misal: untuk mengetahui tipe sebuah variabel (type), memeriksa apakah sebuah variabel merupakan tipe tertentu (is), mengetahui tipe-tipe data apa saja yang didukung (types), mendapatkan representasi STRING dari nilai sebuah variabel (string), dan lainnya.

NULL

Di Singkong, NULL merupakan suatu tipe, dengan nilai adalah kata kunci null (tidak dibedakan huruf besar/kecil). Berbeda dengan bahasa pemrograman besar lainnya, 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, sehingga tidak diperlukan nilai default.

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:

```
> let x = 1/2
> x
0.5000
```

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

```
> let x = 1/0
> x
null
```

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 !=.

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 internal selalu 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 dapat digunakan untuk aplikasi finansial, selama 4 digit setelah koma dinilai cukup. Nilai sebuah bilangan adalah seperti apa yang direpresentasikan/ ditampilkan.

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, integer, number, random, round, sort_number, words_en, words_id
Sistem	delay

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	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 builtin digunakan, sebagai 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:

```
>  let 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 builtin 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, call, center, count, cr, crlf, dir, empty, endswith, equals, eval, in, index, join, left, len, lf, lower, matches, md5, newline, quote, random_string, replace, right, set, sha1, sha256, sha384, sha512, slice, sort_string, split, startswith, stat, tab, trim, upper
Sistem	cwd, separator, user
File	abs, append, read, write
Singkong	load

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

```
>  let s = "singkong"
>  s
"singkong"

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

>  s
"Singkong"
```

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

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:

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

>  a == c
false
```

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]
null
```

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

```
>  let 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 builtin len dapat digunakan. Untuk mendapatkan irisan dari sebuah ARRAY, gunakanlah fungsi builtin slice.

Fungsi bawaan terkait:

Kategori	Daftar fungsi
ARRAY	array, 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
Sistem	arguments, system
File	dir

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:

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

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

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

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

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

Untuk mendapatkan jumlah pemetaan dalam HASH, fungsi builtin len dapat digunakan.

Fungsi bawaan terkait:

Kategori	Daftar fungsi
HASH	empty, keys, len, random, sort_hash, values
Sistem	info
File	properties_read, properties_write, stat
Singkong	singkong

DATE

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

Sebuah nilai DATE diberikan sebagaimana perincian dalam tabel berikut:

DATE	Deskripsi
@	Tanggal dan waktu aktif
@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)

DATE	Deskripsi
@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 builtin 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.

Fungsi	Deskripsi
month	Menambahkan/mengurangi bulan. Akan menambahkan/mengurangi tahun apabila diperlukan.
year	Menambahkan/mengurangi tahun

Fungsi bawaan terkait:

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

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 let, menjadi nama fungsi. Berikut adalah contoh pembuatan fungsi dengan nama f, yang tidak menerima parameter dan tidak mengembalikan nilai secara eksplisit.

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

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

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 let. Berikut adalah contoh pemanggilan fungsi yang dibuat sebelumnya:

```
>  f()
Singkong
null
```

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

```
>  let r = f()
Singkong

>  r
null
```

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

```
>  let 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.

```
>  let 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: 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 builtin 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:

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

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

```
>  let 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 builtin dan harus mengikuti aturan lain penamaan variabel. Berikut adalah contoh yang tidak benar:

```
>  let info = fn(){}
ERROR: info is a builtin function

>  let 10 = fn(){}
PARSER ERROR: expected next token to be IDENT,
got INT instead
```

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

```
>  let 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:

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

>  f(10)
10
1
null

>  x
ERROR: 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:

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

BUILTIN

Pada saat buku ini ditulis, Singkong datang dengan lebih dari 150 fungsi builtin (bawaan) yang menyediakan berbagai fungsionalitas.

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

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

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

```
8
```

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

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

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

Untuk mendapatkan ARRAY berisikan nama semua fungsi builtin yang tersedia, gunakanlah fungsi builtin builtins.

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

Pada contoh berikut, interpreter Singkong dijalankan dengan fungsi builtin info dan system dinonaktifkan:

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

Walaupun dinonaktifkan, kita tetap tidak bisa menggunakan nama fungsi builtin tersebut sebagai variabel:

```
>  let info = 123
ERROR: info is a builtin function


>  info()
ERROR: builtin function "info" is disabled


>  disabled(len)
["info", “system"]
```

Sebuah fungsi builtin bisa dibandingkan dengan fungsi builtin lainnya dengan operator == ataupun !=.

```
>  len == len
true

>  len == print
false

>  info == info
true

>  info == system
false


>  let 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 builtin 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: GUI is not available
```

Oleh karena itu, apabila dirasa perlu, periksalah apakah GUI tersedia atau tidak dengan menggunakan fungsi builtin 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.

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

>  b == b
true

>  let d = b
>  d == b
true
```

Fungsi bawaan terkait:

Kategori	Daftar fungsi
COMPONENT	add, add_e, add_n, add_s, add_w, clear, closing, component, component_type, components, config, event, frame, fonts, get, gui, hide, printer, remove, remove_e, remove_n, remove_s, remove_w, reset, screen, show, size, radio_group, table_add, table_remove, timer, title, stop
Input/Output (versi sederhana apabila GUI tidak tersedia)	confirm, directory, input, open, message, password, save

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, query

Untuk informasi selengkapnya, bacalah bagian pengembangan aplikasi database.

3. Daftar Fungsi Builtin

Fungsi builtin 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 builtin 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 builtin yang disediakan. Pada saat buku ini ditulis, terdapat lebih dari 150 fungsi bawaan yang akan kita lihat di dalam bab ini.

Untuk mendapatkan daftar fungsi bawaan, gunakanlah fungsi builtin 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 builtin 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 builtin lainnya dapat juga dilihat pada berbagai contoh kode program Singkong.

Fungsi	Deskripsi Singkat	Contoh
ABS	Nilai absolut untuk NUMBER atau path absolute untuk file	abs(-1.23)
ADD	Menambahkan COMPONENT atau ARRAY COMPONENT ke region center dari Frame	

Fungsi	Deskripsi Singkat	Contoh
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	
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")

Fungsi	Deskripsi Singkat	Contoh
ARRAY_NUMBER	Mendapatkan ARRAY dengan semua element NUMBER	
ARRAY_STRING	Mendapatkan ARRAY dengan semua elemen adalah representasi STRING dari elemen berbagai tipe	
AVERAGE	Rata-rata elemen NUMBER dalam ARRAY	average([1,2,3,4,5])
BUILTINS	Mendapatkan semua fungsi builtin yang tersedia	
CALL	Memanggil method Java (sesuai aturan modul Singkong)	call("Base64Encode", "Singkong")
CALL_INFO	Mendapatkan informasi method Java yang tersedia (sesuai aturan modul Singkong)	
CENTER	Rata tengah STRING	
CLEAR	Menghapus semua COMPONENT dari Fram	
CLOSING	Mengaktifkan atau menonaktifkan konfirmasi menutup Frame	closing("Apakah yakin ingin keluar dari program?", "Konfirmasi")

Fungsi	Deskripsi Singkat	Contoh
COMPONENT	Membuat komponen GUI	
COMPONENT_TYPE	Mendapatkan tipe sebuah COMPONENT	
COMPONENTS	Mendapatkan daftar semua tipe komponen GUI yang didukung	
CONFIG	Konfigurasi komponen GUI	
CONFIRM	Menampilkan dialog konfirmasi	
COUNT	Mendapatkan jumlah elemen tertentu dalam ARRAY	count([1,2,3], 1)
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	

Fungsi	Deskripsi Singkat	Contoh
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)
DELAY	Sleep untuk sejumlah milidetik	
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 builtin yang dinonaktifkan	
DO	Memanggil sebuah fungsi sejumlah kali	do(5, fn() {println("Singkong")})

Fungsi	Deskripsi Singkat	Contoh
EACH	Untuk setiap elemen dalam ARRAY, panggil sebuah fungsi sejumlah kali, dengan argumen	each([1,2,3], fn(x, y) {println(y + ": " + x)})
EMPTY	Apakah sebuah STRING, HASH, atau ARRAY adalah kosong	
ENDSWITH	Apakah STRING diakhiri dengan STRING tertentu	
EQUALS	Apakah dua STRING sama, tanpa membedakan huruf besar/kecil	
EVAL	Mengevaluasi STRING sebagai kode Singkong	eval("let x = 1000; println(x)") let x = eval("1+2+3")
EVENT	Mendaftarkan event handler untuk komponen GUI	
EXIT	Terminasi program Singkong	
FIRST	Mendapatkan elemen pertama dalam ARRAY	
FONTS	Mendapatkan daftar nama font yang tersedia	

Fungsi	Deskripsi Singkat	Contoh
FORMAT_DATE	Memformat DATE tanpa komponen waktu, dengan format opsional	format_date(@, "MMM DD, YYYY")
FORMAT_DATETIME	Memformat DATE dengan komponen waktu, dengan format opsional	
FORMAT_DIFF	Memformat perbedaan antara dua DATE	format_diff(1.5, " year ", " month ", " day ", 0)
FRAME	Mendapatkan properti Frame	
GET	Mendapatkan konfigurasi komponen GUI	
GUI	Apakah GUI tersedia	
HASH	Mendapatkan hash code sebuah nilai	
HELP	Mendapatkan informasi sebuah fungsi builtin	
HIDE	Menyembunyikan Frame	
HOUR	Menambahkan atau mengurangi jam dari DATE	hour(@, 10)
IN	Apakah ARRAY atau STRING mengandung elemen tertentu	in([1,2,3], 1)

Fungsi	Deskripsi Singkat	Contoh
INDEX	Mendapatkan indeks elemen tertentu dalam ARRAY atau STRING	index([1,2,3,1,2,3], 1)
INFO	Mendapatkan informasi sistem	info()["os.name"]
INPUT	Meminta input	
INTEGER	Konversi NUMBER ke nilai bilangan bulat (integer) saja	
INTERACTIVE	Apakah kode Singkong dijalankan dalam interactive evaluator/editor	
IS	Apakah tipe ekspresi, variabel, dan nilai merupakan tipe tertentu	is([], "ARRAY")
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})
LAST	Mendapatkan elemen terakhir dalam ARRAY	
LEFT	Rata kiri STRING	left("Singkong", 10, "=")

Fungsi	Deskripsi Singkat	Contoh
LEN	Mendapatkan panjang STRING, HASH, atau ARRAY	
LF	Line Feed	
LOAD	Menjalankan file program Singkong lain	load("test.singkong")
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	
MESSAGE	Menampilkan dialog pesan	
MIN	Mendapatkan nilai minimum dalam ARRAY NUMBER	
MINUTE	Menambahkan atau mengurangi menit dari DATE	
MONTH	Menambahkan atau mengurangi bulan dari DATE	

Fungsi	Deskripsi Singkat	Contoh
NEWLINE	Mendapatkan karakter newline dari sistem berjalan	
NUMBER	Konversi STRING atau DATE ke NUMBER	
OPEN	Menampilkan dialog untuk membuka file	
PARAM	Mendapatkan parameter FUNCTION atau parameter wajib BUILTIN	
PART	Mendapatkan ARRAY komponen waktu dari DATE dalam [year, month, day, hour, minute, second]	part(@)
PASSWORD	Meminta input berupa password	
POP	Mendapatkan semua elemen dalam ARRAY kecuali elemen terakhir	
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	
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	

Fungsi	Deskripsi Singkat	Contoh
REMOVE_S	Menghapus semua COMPONENT dari region south Frame	
REMOVE_W	Menghapus semua COMPONENT dari region west Frame	
REPLACE	Mengganti setiap substring dari STRING tertentu dengan STRING lain	
RESET	Menghapus semua COMPONENT dari Frame, mengatur ulang Frame, menghapus semua timer	
REST	Mendapatkan semua elemen dalam ARRAY dari indeks 1	
REVERSE	Membalik ARRAY	reverse(sort_number([3, 1, 2]))
RIGHT	Rata kanan STRING	
ROUND	Membulatkan NUMBER	
SAVE	Menampilkan dialog untuk menyimpan file	
SCREEN	Mendapatkan ukuran layar	

Fungsi	Deskripsi Singkat	Contoh
SECOND	Menambahkan atau mengurangi detik dari DATE	
SEPARATOR	Pemisah direktori dan file di sistem berjalan	
SET	Mengubah STRING, ARRAY, atau HASH	
SHA1	SHA1	
SHA256	SHA256	
SHA384	SHA384	
SHA512	SHA512	
SHOW	Menampilkan Frame	
SHUFFLE	Mengacak urutan elemen dalam ARRAY	
SINGKONG	Mendapatkan informasi Singkong	singkong()["version"]
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]])

Fungsi	Deskripsi Singkat	Contoh
SORT_BOOLEAN	Mengurutkan ARRAY yang semua elemen di dalamnya adalah BOOLEAN	
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", ", ")
STARTSWITH	Apakah STRING diawali dengan STRING tertentu	
STAT	Mendapatkan informasi sebuah file	
STOP	Menghentikan semua timer	

Fungsi	Deskripsi Singkat	Contoh
STRING	Mendapatkan representasi STRING dari sebuah nilai	
SUM	Menjumlahkan semua elemen NUMBER di dalam ARRAY	
SYSTEM	Menjalankan perintah sistem dan mendapatkan outputnya	system(["ls", "-al"])
TAB	Karakter spesial tab	
TABLE_ADD	Menambahkan baris ke dalam komponen GUI table	
TABLE_REMOVE	Menghapus sebuah baris dari komponen GUI table	
TIMER	Memanggil fungsi setiap jeda waktu tertentu (milidetik)	
TITLE	Mengubah title Frame	
TRIM	Mendapatkan STRING dengan whitespace di awal dan akhir STRING dihapus	

Fungsi	Deskripsi Singkat	Contoh
TYPE	Mendapatkan tipe ekspresi, variabel, atau nilai	type(null) type([1,2,3])
TYPES	Mendapatkan semua tipe yang didukung dalam Bahasa Pemrograman Singkong	
UPPER	Konversi STRING ke huruf besar	
USER	Mendapatkan nama user aktif	
VALUES	Mendapatkan semua value dalam pemetaan	values({1:2, 3:4})
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	
YEAR	Menambahkan atau mengurangi tahun dari DATE	

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 builtin 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

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

Contoh 2: penggunaan else

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

Contoh 3: penggunaan operator &

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


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

Contoh 4: penggunaan (dan)

```
let 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

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

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:

```
let actions = {
      1: fn(){
            println("1")
      },
      2: fn() {
            println("2")
      },
      3: fn() {
            println("3")
      }
}

let 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

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

Contoh 2: nilai kembalian perulangan

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

println(r)
```

Fungsi Builtin: do

Ada kalanya, kita hanya ingin melakukan tindakan tertentu beberapa kali. Fungsi builtin 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

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

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

Fungsi Builtin: 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

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

Contoh 2: menggunakan nilai indeks yang dilewatkan

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

Contoh 3: memroses hanya elemen tertentu berdasarkan indeks

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

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

let ss = fn() {
```

```
    let f = save();
    if (!empty(f)) {
        let 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 Singkong hanya dapat bekerja dengan satu Frame per program. Frame dalam hal ini merupakan top level window.

Selain itu, programmer juga tidak dapat membuat dialog baru. Sebagai alternatif, gunakanlah berbagai fungsi builtin yang menyediakan dialog siap pakai: confirm, directory, input, open, message, password, save.

Fungsi builtin terkait Frame:

Fungsi	Contoh
title	title("Hello, World")
size	size(400, 300)
frame	frame()
reset	reset()

Fungsi	Contoh
closing	closing("Apakah yakin ingin keluar dari program?", "Konfirmasi")

Gunakanlah fungsi builtin 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.

Komponen GUI

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

```
> components()
["button", "checkbox", "combobox", "edit",
"image", "label", "password", "radio",
"table", “text"]
```

Komponen	Deskripsi
button	Tombol
checkbox	Check box
combobox	Combo box, tidak dapat diedit
edit	Editor teks lebih dari satu baris, otomatis dilengkapi dengan scroll bar. Dapat diatur agar tidak dapat diedit.
image	Image, dapat digunakan untuk menampilkan gambar dari file

Komponen	Deskripsi
label	Label, dapat digunakan untuk menampilkan teks
password	Input teks berupa password. Dapat diatur agar tidak dapat diedit.
radio	Radio button, dapat berdiri sendiri ataupun merupakan mutual-exclusion set
table	Tabel, dapat digunakan untuk menampilkan data tabular. Dapat diatur agar tidak dapat diedit.
text	Input teks. Dapat diatur agar tidak dapat diedit.

Untuk membuat komponen user interface, gunakanlah fungsi builtin 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
button	Label
checkbox	Label
combobox	Item dalam combobox, dipisahkan koma

Komponen	Interpretasi nama komponen
edit	Teks
image	Nama file
label	Label
password	Teks
radio	Label
table	Kolom tabel, dipisahkan koma
text	Teks

Berikut adalah contoh pembuatan komponen user interface:

```
let b = component("button", "Hello");
let c = component("checkbox", "Singkong?");
let m = component("combobox",
"Singkong,Programming,Language");
let e = component("edit", "Hello, World");
let i = component("image", "image.jpg");
let l = component("label", "Singkong
Programming Language");
let p = component("password", "test");
let r = component("radio", "Radio Button");
let t = component("table", "A,B,C,D,E");
let x = component("text", "Singkong");
```

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

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

Menambahkan/Menghapus Komponen

Untuk menambahkan komponen ke dalam Frame, gunakanlah salah satu dari fungsi builtin 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 builtin 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 builtin clear.

Untuk menghapus semua komponen dari Frame, mengubah kembali title dan ukuran ke nilai default, serta menghentikan semua timer yang ada, 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. Ukuran komponen akan dihitung agar mengisi semua ruang yang tersedia, sesuai dengan ukuran Frame.

Berikut adalah contoh penempatan komponen user interface:

```
reset();
let c = component("button", "C");
let e = component("button", "E");
let n = component("button", "N");
let w = component("button", "W");

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

add(c);
add_e(e);
add_n(n);
add_s(s);
add_w(w);

show();
```

Konfigurasi Komponen

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

Untuk mendapatkan konfigurasi sebuah komponen user interface, gunakanlah fungsi builtin 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:

Key	Tipe	Deskripsi	Komponen
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

Key	Tipe	Deskripsi	Komponen
font	ARRAY [STRING (nama font), NUMBER (0=plain, 1=bold, 2=italic, 3=bold dan italic), NUMBER (ukuran)]	Font komponen	Semua komponen
text	STRING	label button/ checkbox/ label/radio, item terpilih untuk combobox	button/ checkbox/ label/radio dan combobox
active	BOOLEAN atau NUMBER	Terpilih atau tidak untuk checkbox/ radio (BOOLEAN), baris terpilih untuk combobox/ table (NUMBER)	checkbox/ radio dan combobox/ table

Key	Tipe	Deskripsi	Komponen
contents	STRING, ARRAY, ARRAY dari ARRAY	Isi komponen: combobox/ edit/password/ table/text, nama file untuk image. Untuk combobox: ARRAY. Untuk table: ARRAY dari ARRAY.	combobox/ edit/password/ table/text dan image

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

Terdapat beberapa fungsi builtin yang bekerja khusus untuk komponen user interface tertentu, sebagai dirinci 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
radio	radio_group	mutual-exclusion set untuk sejumlah radio button	ARRAY COMPONENT (radio)

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
radio	Ketika dipilih atau tidak dipilih
table	Ketika baris aktif berubah
edit	Ketika isinya berubah
password	Ketika isinya berubah
text	Ketika isinya berubah

Untuk mendaftarkan fungsi yang akan otomatis dipanggil ketika event tersebut terjadi, gunakanlah fungsi builtin event. Fungsi ini menerima dua argumen: COMPONENT dan FUNCTION. Fungsi yang akan dipanggil tidak dapat menerima argumen.

Berikut adalah contoh penanganan event di Singkong:

```
reset();
let b = component("button", "Hello, World");
let c = component("checkbox", "Singkong?");
let r = component("radio", "Radio Button");
let m = component("combobox",
"Singkong,Programming,Language");
let t = component("table", "A,B", true);
```

```
let e = component("edit", "");
let p = component("password", "");
let x = component("text", "");

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

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

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

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

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

let ee = fn() {
    message(get(e, "contents"));
};
event(e, ee);

let pp = fn() {
```

```
    message(get(p, "contents"));
};
event(p, pp);

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

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

Timer

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

Untuk menghentikan semua timer, gunakanlah fungsi builtin stop.

Berikut adalah contoh penggunaan timer:

```
reset();
let l = component("label", string(@));
let b1 = component("button", "start");
let b2 = component("button", "stop");

add(l);
add_s([b1, b2]);
```

```
let f = fn() {
    config(l, "text", string(@));
}

let timer_start = fn() {
    timer(1000, f);
}

let timer_stop = fn() {
    stop();
}

event(b1, timer_start);
event(b2, timer_stop);

show();
```

Pencetakan ke Printer

Untuk mencetak ARRAY dari STRING ke printer, gunakanlah fungsi builtin 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).

Berikut adalah contoh pencetakan sederhana:

```
reset();
let t = ["Singkong", "Programming",
"Language"];
let s = 16;
let x = 150;
let y = 150;
let font = "monospaced";
printer(t, s, x, y, font);
```

Fungsi Lain

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

Untuk mendapatkan semua nama font yang tersedia di sistem, gunakanlah fungsi builtin fonts.

Contoh Lain

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

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

add_n(l);
add(t);
```

```
let s = singkong();
let a = []
let f = fn(x,i) {
    let v = s[x];
    let 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.

Berbeda dengan semua pembahasan sebelumnya, cara interpreter Singkong dijalankan standalone akan berbeda ketika perlu bekerja dengan sistem database relasional. Alasannya adalah karena Singkong.jar tidak datang bersama driver database relasional apapun. Kita akan membutuhkan driver (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 di Singkong telah diusahakan agar sesederhana dan seringkas mungkin, namun karena terhubung dengan sistem eksternal, beberapa informasi berikut mungkin perlu Anda ketahui terlebih dahulu:

- Dimana driver 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.

Setelah memastikan bahwa driver database tersimpan dalam file jar atau direktori tertentu, jalankanlah interpreter Singkong dengan cara demikian. 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 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.

Koneksi Database

Untuk membuat koneksi database, kita menggunakan fungsi builtin database. Fungsi ini mengembalikan sebuah DATABASE, 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 H2, dimana driver disimpan pada file h2.jar, di dalam direktori yang sama dengan Singkong.jar, yaitu di direktori aktif. Sistem operasi yang digunakan adalah macOS sehingga pemisah class path adalah titik dua (:). Perintah diketikkan dalam baris yang sama.

```
java -cp Singkong.jar:h2.jar
com.noprianto.singkong.Singkong
```

Di dalam interactive evaluator Singkong:

```
>  let db = database("org.h2.Driver",
"jdbc:h2:/tmp/test", "admin", "admin")
>  db
DATABASE (URL=jdbc:h2:/tmp/test, user=admin,
driver=org.h2.Driver)
```

Untuk mendapatkan informasi apakah sebuah DATABASE terkoneksi ke sistem database, gunakanlah fungsi builtin 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	
NUMBER	int	Apabila NUMBER terlihat seperti bilangan bulat
NUMBER	java.math.BigDecimal	
STRING	String	

Singkong hanya mendukung tipe data tersebut ketika bekerja dengan sistem database. Apabila di dalam query diberikan tipe selain yang didukung, maka akan terjadi kesalahan dan transaksi akan dibatalkan.

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 builtin query. Fungsi ini menerima dua argument: 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 tipe sebagaimana yang dirinci pada tabel sebelumnya).

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

Berikut adalah contoh query ke sistem database untuk pembuatan sebuah tabel:

```
let d = database("org.h2.Driver", "jdbc:h2:/
tmp/test", "admin", "admin");
if (database_connected(d)) {
    let q = [ ["create table test(a integer, b
varchar(64))", []] ]
    let r = query(d, q);
    println(r);
}
```

Berikut adalah contoh query ke sistem database untuk insert, update, dan select:

```
let d = database("org.h2.Driver", "jdbc:h2:/
tmp/test", "admin", "admin");
if (database_connected(d)) {
    let q = [
        ["insert into test(a,b) values(?, ?)",
[random(1,10), "hello"]],
        ["update test set b=? where b=?",
["Hello World", "hello"]]
    ]
    let r = query(d, q);
    println(r);

    let q = [ ["select * from test", []] ]
    let r = query(d, q);
    println(r);
}
```

Contoh Lain

Berikut adalah contoh menampilkan isi sebuah tabel ke dalam tabel GUI:

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

let d = database("org.h2.Driver", "jdbc:h2:/
tmp/test", "admin", "admin");
if (database_connected(d)) {
    let q = [ ["select a,b from test", []] ]
    let r = query(d, q);
    if (!empty(r)) {
        config(t, "contents", r[0]);
    }
}

show();
```

7. 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 builtin dan barangkali 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 builtin untuk bekerja dengan STRING ataupun konversi ke tipe lain.
- Dengan fungsi builtin 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, kita perlu memberitahu Java Runtime Environment ke mana harus mencari class yang berisi method tersebut. Tentu saja, di dalam Singkong.jar, tidak terdapat class-class 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).

Untuk memanggil method yang ditulis dalam Java, gunakanlah fungsi builtin 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 builtin 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 builtin 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 builtin 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
```

Program Singkong selanjutnya dapat memanggil method tersebut, yang akan mengembalikan STRING:

```
    >  let 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):

```
    >  let 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,
    null
```

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)):

```
    >  let 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"]
    null
```

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:

```
> let code = call("HelloWorldEval", [[],
[1], [2,3], [4,5,6]])
> code
"component("button", "; 1, ; 2, 3, ; 4, 5,
6, ; ")"

> let b = eval(code)
> b
COMPONENT: button (; 1, ; 2, 3, ; 4, 5, 6,
; )

> add(b)
null

> show()
null
```

Contoh: Informasi

Berikut adalah contoh-contoh penggunaan fungsi builtin 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: Base64

Contoh berikut membutuhkan Java 8.

Base64Encode.java (disimpan di dalam direktori examples, dikompilasi ke Base64Encode.class: javac Base64Encode.java)

```
   public class Base64Encode {
        public static String singkong_info() {
            return "Base64Encode: Base64 encode String (using
Java 8 or later)";
        }

        public static String singkong(String[][] args) {
            String ret = "";
            if (args.length > 0) {
                String[] r = args[0];
                if (r.length > 0) {
                    String s = r[0];
                    ret =
java.util.Base64.getEncoder().encodeToString(s.getBytes());
                }
            }
            return ret;
        }

        public static void main(String[] args) {
            System.out.println("Base64Encode: Singkong
Programming Language module (using Java 8 or later)");
        }
    }
```

Base64Decode.java (disimpan di dalam direktori examples, dikompilasi ke Base64Decode.class: javac Base64Decode.java)

```
   public class Base64Decode {
        public static String singkong_info() {
            return "Base64Decode: Base64 decode String (using
Java 8 or later)";
        }

        public static String singkong(String[][] args) {
            String ret = "";
            if (args.length > 0) {
                String[] r = args[0];
                if (r.length > 0) {
                    String s = r[0];
                    byte[] b =
java.util.Base64.getDecoder().decode(s);
```

```
                    ret = new String(b);
                }
            }
            return ret;
        }

        public static void main(String[] args) {
            System.out.println("Base64Decode: Singkong
Programming Language module (using Java 8 or later)");
        }
    }
```

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-method tersebut, yang masing-masing akan mengembalikan STRING:

```
> let s = "Singkong Programming Language"
> let e = call("Base64Encode", s)
> e
"U2luZ2tvbmcgUHJvZ3JhbW1pbmcgTGFuZ3VhZ2U="

> let d = call("Base64Decode", e)
> d
"Singkong Programming Language"

> s == d
true
```

Halaman ini sengaja dikosongkan

8. 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 builtin 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 builtin 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 builtin 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 builtin 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 builtin 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 builtin 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)
```

Interpretasi

Isi file Test.java

```
import com.noprianto.singkong.Singkong;

public class Test {
    public static void main(String[] args) {
        String code = "let 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]
null
```

Interpretasi, Builtin

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: builtin 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
null
```

Interpretasi, Environment, Builtin

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: builtin 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
null
```

Interpretasi, Environment, Builtin, 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: builtin function "info" is disabled
```

Contoh: Bahasa Pemrograman Lain

Bukan merupakan embedding Singkong yang sesungguhnya, namun barangkali bisa berguna.

Python dengan semua fungsi builtin tersedia:

```
>>> import subprocess
>>> c = ['java', '-jar', 'Singkong.jar',
'puts("singkong")']
>>> o = subprocess.check_output(c)
>>> print(o)
singkong
null
```

Python dengan fungsi builtin tertentu dinonaktifkan:

```
>>> import subprocess
>>> c = ['java', '-DDISABLE=info', '-jar',
'Singkong.jar', 'info()']
>>> o = subprocess.check_output(c)
>>> print(o)
ERROR: builtin function "info" is disabled
```

9. 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.

Dalam perjalanan, Singkong menemukan jalannya sendiri dan walaupun secara sintaks masih mirip, terdapat sejumlah perbedaan yang barangkali perlu dibahas. Source code Singkong pada saat buku ini ditulis setidaknya berukuran sekitar 5 kali Monkey.java.

Walau demikian, tanpa Monkey, tidak terbayangkan akan adanya Singkong.

Berikut adalah perbedaan antara Singkong dan Monkey:

- Singkong tidak membedakan huruf besar dan huruf kecil (case-insensitive)
- Nama variabel Singkong dimulai dengan huruf atau underscore dan secara opsional dapat diikuti dengan huruf, bilangan, dan underscore
- Singkong menyediakan literal null
- 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 lebih dari 150 fungsi builtin 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 builtin: do, each
- Sejumlah fungsi builtin dapat dinonaktifkan ketika interpreter Singkong dijalankan

Pengembangan, Kompatibilitas, Modul

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 dijalankan dan diuji pada Java 8 (utama), pada Java 5.0, dan pada Java versi terbaru.

Singkong akan berusaha sebaik mungkin agar backward compatible, sehingga kode Singkong yang telah Anda buat diharapkan tetap berjalan pada versi-versi Singkong yang akan datang.

Daftar modul eksternal Singkong (fungsionalitas tambahan yang ditulis dengan Java) dikelola di website penulis. Apabila Anda mengembangkan modul untuk Singkong, mohon berkenan untuk informasikan juga kepada penulis.

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 lebih dari 150 builtin function (fungsi 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 menyukai pemrograman, memiliki sertifikasi Java (OCP), dan telah menulis beberapa buku cetak: Python (2002), Debian GNU/Linux (2006), OpenOffice.org (2006), Java (2018), dan Singkong (2020). Beliau juga menulis beberapa buku elektronik gratis: wxWidgets (2006), Python (2009), SQLiteBoy (2014), dan OpenERP (rekan penulis, 2014).

Noprianto lulus dari jurusan teknik informatika (2003), manajemen sistem informasi (2015), dan doktor ilmu komputer (2019) dari Universitas Bina Nusantara.

Noprianto mengembangkan bahasa pemrograman Singkong dan interpreternya sejak 2019. Buku dan softwarenya dapat didownload dari noprianto.com

---

Penerbit:
PT. Stabil Standar Sinergi
Download gratis buku ini:
http://noprianto.com

ISBN 978-602-52770-1-6

9 786025 277016