

**MEMAHAMI  
PEMROGRAMAN BERORIENTASI OBJEK  
Dengan Pendekatan ALICE 3.0**

Arie Vatresia, PhD<sup>1</sup>  
Ferzha Putra Utama, M. Eng<sup>2</sup>  
Edy Hermansyah, PhD<sup>1</sup>

<sup>1</sup>Informatika, Universitas Bengkulu

<sup>2</sup>Sistem Informasi, Universitas Bengkulu

## KATA PENGANTAR

Alhamdulillahirabbilalamin

Penulis mengucapkan syukur kepada Allah dengan terselesaikannya buku ini. Buku ini merupakan penjelasan sederhana untuk memahami pemrograman berorientasi objek dengan menggunakan ALICE 3.0. Materi yang penulis dapatkan dengan mengikuti pelatihan bersama Oracle Academy yang bekerjasama dengan APTIKOM dan prodi Informatika. Buku ini juga berisi tentang langkah-langkah mudah untuk memahami pemrograman dengan pendekatan animasi yang ditawarkan pada program ALICE 3.0.

Buku ini ditujukan kepada siapa saja yang ingin memahami pemrograman berorientasi objek baik dari kalangan siswa atau mahasiswa yang baru belajar tentang pemrograman berbasis bahasa pemrograman Java. Penulis berterimakasih kepada:

1. Oracle Academy yang telah mengizinkan penulis untuk membahas lebih lanjut hasil pelatihan untuk menjadi buku ajar untuk mahasiswa.
2. Prodi Informatika yang telah memfasilitasi dan menjalin kerja sama dengan oracle academy untuk memungkinkan buku ini untuk diterbitkan.
3. Edy Hermansyah, Ph.D. dan Ruvita Faurina, M.Eng. sebagai tim editor yang telah meluangkan waktu untuk melihat kelayakan dari buku ini.
4. LPPM yang telah memfasilitasi terbitnya buku ini.

Akhir kata, penulis menyampaikan permohonan maaf apabila buku ini masih banyak kekurangan dalam penulisan. Masukan dan saran yang membangun akan difasilitasi untuk meningkatkan kualitas buku ini. Semoga buku ini bermanfaat dan bias digunakan dengan baik.

Penulis

## **DAFTAR ISI**

KATA PENGANTAR	ii
DAFTAR TABEL	xvii
DAFTAR GAMBAR	xix
BAB I Mengenal Alice	1
1.1 Adegan awal	1
1.2 Langkah membuat projek baru	2
1.3 Cara menyimpan projek	3
1.4 Navigasi antara editor	3
1.5 Menambah sebuah objek pada sebuah halaman	4
1.6 Memberikan nama objek	5
1.7 Editor Layar	5
1.8 Tampilan Editor Layar	6
1.9 Galeri	6
1.10 Select a class	7
1.11 Menyimpan versi baru pada program	7
1.12 Editor kode	8
1.13 Methods panel	9
1.14 Menu instance di panel metode	10
1.15 Uji dan debug pengujian animasi anda	12
BAB II Menambah dan Memposisikan Objek	14
2.1 Membuka proyek yang telah ada	14
2.2 Memposisikan objek dalam adegan awal	16
2.3 Fitur pemosision untuk objek	16
2.4 Cara untuk memosisiskan objek	17

2.5 Memilih contoh untuk posisi	18
2.6 Prosedur <i>one shot</i> posisi yang tepat	19
2.7 Posisi objek di adegan – Posisi yang tepat	19
2.8 Properti posisi – Posisi yang tepat	22
2.9 Memodifikasi properti	25
2.10 Memosisikan sub-bagian objek	26
2.11 Terminologi	27
2.12 Ringkasan	28
BAB III Prosedur dan Argumen	29
3.1 Menampilkan <i>code editor</i>	29
3.2 Pilih <i>Instance</i>	30
3.3 Panel metode	30
3.4 Code editor tabs	32
3.5 Control statements	33
3.6 Gerakan objek	33
3.7 Membuat instruksi pemrograman	34
3.8 Menu argumen	37
3.9 Argumen sebagai <i>placeholder</i>	37
3.10 Langkah-langkah untuk mengatur ulang statement pemrograman	37
3.11 <i>Edit statement</i> pemrograman	38
3.12 Menghapus <i>statement</i> pemrograman	38
3.13 <i>Edit</i> dan uji program	39
3.14 <i>Debugging</i> dan pengujian	40
3.15 Masukkan statement pemrograman sementara untuk membantu <i>debugging</i>	40
3.16 Menonaktifkan pernyataan pemrograman	41
3.17 Mengaktifkan kembali <i>statement</i> pemrograman yang dinonaktifkan	41

3.18 Komentar Pemrograman	42
3.19 Menggunakan Komentar untuk Mengatur Program	43
3.20 Terminologi	43
3.21 Ringkasan	44
BAB IV Rotasi dan Pengacakan	45
4.1 Pengembangan program menggunakan pendekatan <i>Top-Down</i>	45
4.2 Langkah-langkah menggunakan pendekatan <i>Top-Down</i> untuk pemrograman	45
4.3 Contoh textual storyboard	46
4.4 Sejajarkan <i>storyboard</i> dengan instruksi pemrograman	48
4.5 Gerakan objek	48
4.6. Contoh Rotasi Sub-Bagian	50
4.7 Pernyataan kontrol	51
4.8 Pernyataan kontrol di <i>code editor</i>	52
4.9 Membuat blok pemrograman	54
4.10 Angka acak	55
4.11 Terminologi	56
4.12 Ringkasan	57
BAB V Mendeklarasikan Prosedur	58
5.1 Gerakan objek	58
5.2 Skenario dan animasi	58
5.3 Storyboard	59
5.4 Format papan cerita	60
5.5 Menggunakan <i>storyboard</i> untuk mengatur program	64
5.6 Warisan	64
5.7 Warisan kelas	64
5.8 Karakteristik yang diwarisi	65

5.9 Membuat metode warisan	65
5.10 Mengidentifikasi perilaku berulang di <i>storyboard</i>	65
5.11 Tab myFirstMethod	66
5.12 Kelas hirarki	66
5.13 Lihat metode kelas	67
5.14 Abstraksi prosedural	67
5.15 Menyatakan prosedur	69
5.16 Abstraksi prosedural dan menggunakan <i>clipboard</i>	73
5.17 Menggunakan prosedur warisan	73
5.18 Deklarasikan prosedur di tingkat <i>superclass</i>	73
5.19 Pengidentifikasi objek “ <i>this</i> ”	74
5.20 Terminologi	75
5.21 Ringkasan	75
<b>BAB VI Control Statements</b>	<b>77</b>
6.1 Argumen	77
6.2 Argumen <i>placeholder</i>	78
6.3 Pergerakan serentak	78
6.4. Prosedur mengimbangi satu sama lain	81
6.5 Prosedur <i>setVehicle</i>	81
6.6 Terminologi	84
6.7 Ringkasan	84
<b>BAB VII Functions</b>	<b>85</b>
7.1 Fungsi	85
7.2 Tab Fungsi di panel metode	86
7.3 Fungsi Memecahkan Masalah Jarak	86
<b>BAB VIII IF dan WHILE</b>	<b>89</b>

8.1 Alice 3	89
8.2 Prosedur dalam Alice 3	90
8.3 Metode panel di Alice 3	90
8.4 Mendeklarasikan prosedur dalam Alice 3	90
8.5 Metode di Java	91
8.6 Metode dalam contoh Java	91
8.7 Keputusan untuk setiap metode	91
8.8 Properti metode	92
8.9 Metode Findmax()	92
8.10 Kelas di Alice 3	93
8.11 Kelas di Java	93
8.12 Kode untuk membuat kelas cat (kucing) di Java	94
8.13 Contoh di Alice 3	94
8.14 Instances di Java	95
8.15 Membuat instance dari class	95
8.16 Control structure	96
8.17 IF control structure	96
8.18 WHILE control structure	97
8.19 Input dan output	98
8.20 Kode input dan output	98
8.21 Terminologi	99
8.22 Ringkasan	99
<b>BAB IX Fungsi</b>	100
9.1 Fungsi	100
9.2 Fungsi tepat menjawab pertanyaan	100
9.3 Tab fungsi	100

9.4 Fungsi memecahkan masalah jarak	101
9.5 Gunakan fungsi get distance to	102
9.6 Langkah-langkah untuk menggunakan fungsi get distance to	102
9.7 Uji fungsinya	103
9.8 Hindari tabrakan	104
9.9 Menggunakan operator matematika	104
9.10 Periksa perhitungan matematika	105
9.11 Tip operator matematika	105
9.12 Langkah-langkah menggunakan operator matematika untuk menghindari tabrakan	105
9.13 Memahami menu matematika contoh	107
9.14 Hapus depth objek dari fungsi	108
9.15 Depth diukur dari pusat objek	109
9.16 Langkah-langkah untuk menghapus dept dari fungsi	109
9.17 Terminologi	109
9.18 Ringkasan	110
<b>BAB X Struktru Kontrol IF dan WHILE</b>	111
10.1 Struktur Control	111
10.2 Struktur kontrol tersedia di Alice 3	111
10.3 Tampilan struktur control	112
10.4 Contoh struktur kontrol	112
10.5 Struktur kontrol bersarang	112
10.6 Contoh kode struktur kontrol bersarang	113
10.7 Eksekusi bersyarat menggunakan struktur kontrol	113
10.8 Struktur kontrol IF	113
10.9 Struktur control IF bagian	114
10.10 Struktur Kontrol WHILE	114

10.11 Struktur kontrol WHILE dan eksekusi perulangan	114
10.12 Menafsirkan struktur kontrol IF	115
10.13 Alur proses struktur kontrol IF	115
10.14 Contoh alur proses struktur kontrol IF	115
10.15 Langkah-langkah untuk memprogram struktur kontrol IF	116
10.16 Contoh struktur kontrol IF	117
10.17 Eksekusi bersyarat	117
10.18 Struktur kontrol WHILE	117
10.19 Alur proses struktur kontrol WHILE	118
10.20 Langkah-langkah untuk memprogram struktur kontrol WHILE	118
10.21 Contoh kode struktur kontrol WHILE	119
10.22 Langkah-langkah untuk menguji struktur kontrol WHILE	119
10.23 Terminologi	120
10.24 Ringkasan	121
<b>BAB XI Ekspresi</b>	<b>122</b>
11.1 Menggunakan ungkapan	122
11.2 Ekspresi di Alice 3	122
11.3 Lokasi operator matematika	122
11.4 Pandangan ekspresi dengan argument jarak jauh	123
11.5 Tampilan fungsi	123
11.6 Masalah jarak	124
11.7 Langkah-langkah untuk membuat ekspresi	124
11.8 Langkah-langkah untuk memindahkan objek, jarak ke objek lainnya	124
11.9 Langkah-langkah untuk memodifikasi jarak menggunakan operator matematika	125
11.10 Langkah-langkah untuk merubah jarak menggunakan operator matematika	125

11.11 Mengubah nilai ekspresi	126
11.12 Contoh nilai ekspresi	126
11.13 Kurangi kedalaman dari ekspresi	127
11.14 Langkah – langkah untuk mengurangi kedalaman dari ekspresi	127
11.15 Menafsirkan ekspresi	127
11.16 Membangun ekspresi contoh	128
11.17 Interpretasi ekspresi	129
11.18 Merumuskan ekspresi	130
11.19 Contoh ekspresi jawaban	130
11.20 Terminologi	130
11.21 Ringkasan	130
<b>BAB XII Variabel</b>	<b>132</b>
12.1 Variable	132
12.2 Contoh variable	132
12.3 Variable untuk penyimpanan data	133
12.4 Properti objek	133
12.5 Jenis data variabel di Alice 3	134
12.6 Mendeklarasikan variabel	135
12.7 Inisialisasi variabel	135
12.8 Mengubah nilai yang diinisialisasi	135
12.9 Langkah-langkah untuk mendeklarasikan variabel	136
12.10 Contoh variabel	136
12.11 Menggunakan variabel dalam prosedur	137
12.12 Menggunakan variabel dalam perhitungan matematika	138
12.13 Langkah-langkah untuk mengacak nilai yang diinisialisasi	138
12.14 Tampilan pengacakan sebuah nilai yang diinisialisasikan	139

12.15 Menampilkan kode alice dalam bahasa java	139
12.16 Terminologi Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:	141
12.17 Ringkasan	142
<b>BAB XIII Kontrol pada Keyboard</b>	<b>143</b>
13.1 Penanganan acara	143
13.2 Apa itu acara?	143
13.3 Apa yang terjadi ketika suatu acara terjadi?	143
13.4 Kontrol keyboard	144
13.5 Contoh kontrol keyboard	144
13.6 Event listener	144
13.7 Jenis event listener	145
13.8 Langkah-langkah untuk mengakses event listener	145
13.9 Tab event listener	145
13.10 Pendengar aktivasi adegan	145
13.11 Listener keyboard	145
13.12 Jenis listener keyboard	146
13.13 Program keyboard event listener	146
13.14 Langkah-langkah untuk menambahkan event listener keyboard	146
13.15 Langkah-langkah untuk memprogram struktur kontrol if	147
13.16 Langkah-langkah untuk memilih kunci keyboard untuk diaktifkan gerak objek	147
13.17 Langkah-langkah untuk program gerakan diaktifkan oleh kunci tekan	147
13.18 Program tindakan pendengar tambahan	147
13.19 Langkah-langkah untuk memprogram tindakan pendengar tambahan	148
13.20 Contoh instruksi pemrograman lengkap	148

13.21 Langkah-langkah untuk memindahkan objek menggunakan tombol panah	148
13.22 Langkah-langkah untuk menguji event listener	148
13.23 Menggunakan kelas yang ada dalam animasi lain	148
13.24 Menyimpan file kelas di Alice 3	149
13.25 Menggunakan tab pemula untuk membuat dunia	149
13.26 Menambahkan file kelas di Alice 3	149
13.27 Terminologi	150
13.28 Ringkasan	150
<b>BAB XIV Mengembangkan Aplikasi Lengkap</b>	<b>151</b>
14.1 Animasi	151
14.2 Animasi membutuhkan perencanaan	151
14.3 Contoh dekomposisi fungsional	152
14.4 Proses pengembangan animasi	152
14.5 Format storyboard	154
14.6 Storyboard visual	154
14.7 Storyboard tekstual	155
14.8 Komponen storyboard	156
14.9 Alur proses storyboard	157
14.10 Bagaimana storyboard bermanfaat	157
14.11 Animasi checklist	158
14.12 Buat animasi lengkap	158
14.13 Tugas debugging	159
14.14 Uji elemen animasi anda	160
14.15 Input pengguna untuk mengubah posisi objek saat di run-time	160
14.16 Langkah-langkah untuk menambahkan prosedur add default model manipulation	160

14.17 Menggunakan prosedur add default model manipulation	161
14.18 Meng-upload animasi anda	161
14.19 Presentasikan proyek animasi anda	163
14.20 Terminologi	165
14.21 Ringkasan	166
<b>BAB XV Variabel dan Tipe Data pada Java</b>	<b>167</b>
15.1 Perbedaan Alice 3 dengan Java	167
15.2 Variabel pada Alice 3	167
15.3 Mengatur variabel nilai	168
15.4 Properti objek	168
15.5 Mendeklarasikan variabel dalam Alice 3	168
15.6 Deklarasi variabel di Java	169
15.7 Tipe data di Alice 3	169
15.8 Variabel tipe data di Alice 3	170
15.9 Contoh tipe data di Java	171
15.10 Tipe data dasar pada Java	171
15.11 Operasi aritmatika di Alice 3	172
15.12 Operator aritmatika dalam argument jarak	173
15.13 Operasi aritmatika pada Java	173
15.14 Operator relasional	173
15.15 Operator relasional di Alice 3	174
15.16 Operator relasi di Java	174
15.17 Operator relational dalam contoh Java	175
15.18 Operator logika dalam Alice 3	175
15.19 Contoh operator logika dalam Alice 3	176
15.20 Operator logika di Java	176

15.21 Contoh operator logika di java	177
15.22 Operator tugas di Alice 3	177
15.23 Operator penugasan di java	178
15.24 Menangani operasi standar di Java	179
15.25 Sintaks penugasan untuk operasi lain	179
15.26 Contoh kode sintaks penugasan	179
15.27 Terminologi	180
15.28 Ringkasan	180

## **DAFTAR TABEL**

Tabel 1.1 Tab dan Fungsi pada Galeri	6
Tabel 2.1 Jenis-jenis Handle Style	19
Tabel 3.1 Contoh Prosedur Gerakan	28
Tabel 3.2 Contoh Prosedur Rotasi	28
Tabel 4.1 Storyboard dengan Instruksi Pemrograman	39
Tabel 4.2 Gerakan Objek	40
Tabel 4.3 Pernyataan Kontrol	43
Tabel 5.1 Skenario dan Animasi	59
Tabel 5.2 Komponen Papan Cerita Tekstual	62
Tabel 6.1 pernyataan kontrol do together	68
Tabel 6.2 Textboard Story Berjalan	69
Tabel 8.1 Perbandingan Alice 3 dan Java	89
Tabel 8.2 Properti Metode	92
Tabel 8.3 Control Structure	96
Tabel 12.1 Jenis Variabel	134
Tabel 13.1 Jenis Pendengar Keyboard	145
Tabel 14.1 Contoh Skenario dan Animasi	153
Tabel 14.2 Komponen Storyboard	156
Tabel 14.3 Komponen dan fungsi dalam animasi	158

Tabel 15. 1 Perbedaan Alice dan Java	167
Tabel 15.2 Variabel Tipe Data di Alice	170
Tabel 15.3 Tipe Data Dasar pada Java	171
Tabel 15.4 Operator relasi di Java	174
Tabel 15.5 Operator Logika di Java	177
Tabel 15.6 Sintaks Penugasan	179

## **DAFTAR GAMBAR**

Gambar 2.1 Membuka Project	15
Gambar 2.2 Mencari Project	15
Gambar 2.3 Sumbu X, Y, Z Objek	16
Gambar 2.4 Pergerakan Objek	17
Gambar 2.5 Nama Objek	17
Gambar 2.6 Posisi X, Y, Z Objek	18
Gambar 2.7 Hasil Pengaturan Posisi Objek	18
Gambar 2.8 Lingkaran Objek	19
Gambar 2.9 Posisi Objek Awal	20
Gambar 2.10 Penggunaan One-shot	20
Gambar 2.11 Prosedur Pada Nilai Argumen	21
Gambar 2.12 Contoh Penggunaan Positioning	21
Gambar 2.13 Posisi Objek Menggunakan Koordinat	22
Gambar 2.14 Posisi X, Y, Z	23
Gambar 2.15 Menentukan Posisi Objek Menggunakan Lingkaran dan Panah	23
Gambar 2.16 Penggunaan Handle Style	24
Gambar 2.17 Properti Objek	25
Gambar 2.18 Opacity Properti	25
Gambar 2.19 Pengubahan Properti	26
Gambar 2.20 Menampilkan Properti Instance	26
Gambar 2.21 Mengubah Arah Kepala Objek	26
Gambar 2.22 Posisi Sub-bagian Objek	27
Gambar 3.1 Tampilan Awal	24
Gambar 3.2 Membuat Instance	25
Gambar 3.3 Procedures dan Functions	25
Gambar 3.4 Tab Prosedur	26
Gambar 3.5 Tab Fungsi	27
Gambar 3.6 <i>Code Editor Tab</i>	27
Gambar 3.7 Intruksi Pemrograman	29
Gambar 3.8 Nilai Argumen	30
Gambar 3.9 Menjalankan Program	30
Gambar 3.10 Penggunaan Argumen	31

Gambar 3.11 Penggunaan DecimalNumber	31
Gambar 3.12 Mengatur Ulang Statement Pemrograman	32
Gambar 3.13 Mengubah Statement Pemrograman	32
Gambar 3.14 Menghapus Bagian <i>Statement</i> Pemrograman	32
Gambar 3.15 Menghapus Seluruh Statement Pemrograman	33
Gambar 3.16 Ubah dan Uji Program	33
Gambar 3.17 Debugging dan Pengujian	33
Gambar 3.18 Membatalkan Is Enabled	34
Gambar 3.19 Hasil Penonaktifan Is Enabled	34
Gambar 3.20 Mengaktifkan Is Enabled	35
Gambar 3.21 Hasil Pengaktifkan Is Enabled	35
Gambar 3.22 Memasukkan Komentar	36
Gambar 3.23 Menggunakan Komentar untuk Mengatur Program	36
Gambar 4.1 Penggunaan Textual Storyboard	39
Gambar 4.2 Objek Berbelok dan Menggulung	41
Gambar 4.3 Objek Bergulir	41
Gambar 4.4 Objek Berputar Maju	41
Gambar 4.5 Objek Berputar Mundur	42
Gambar 4.6 Rotasi Sub-bagian	42
Gambar 4.7 Memutar Objek	43
Gambar 4.8 Statemen Kontrol Code Editor	44
Gambar 4.9 Pergerakan Objek	45
Gambar 4.10 Penggunaan Nest Do	46
Gambar 4.11 Penggunaan Bilangan Acak	47
Gambar 4.12 Contoh Animasi Angka Acak	48
Gambar 5.1 Storyboard Visual	51
Gambar 5.2 Storyboard Tekstual	53
Gambar 5.3 Warisan	55
Gambar 5.4 Pewarisan Pada Anjing Dalmatian	55
Gambar 5.5 Tab myFirstMethod	56
Gambar 5.6 Kelas Hirarki	57
Gambar 5.7 Metode Kelas	57

Gambar 5.8 Abstraksi Prosedural Pada Ikan	58
Gambar 5.9 Abstraksi Prosedural Pada Burung	58
Gambar 5.10 Prosedur Pada Kelinci	60
Gambar 5.11 Mendeklarasikan Prosedur	60
Gambar 5.12 Prosedur	61
Gambar 5.13 Nama Prosedur	61
Gambar 5.14 Mengakses dan Mengedit Prosedur yang Dideklarasikan	61
Gambar 5.15 Pemisahan Prosedur	62
Gambar 5.16 Prosedur Warisan	63
Gambar 5.17 Prosedur Tingkat Superclass	63
Gambar 5.18 Identifikasi Objek “this”	64
Gambar 5.19 Prosedur bipedWave	64
Gambar 6.1 Argumen Dalam Prosedur	66
Gambar 6.2 Mengubah Argumen	67
Gambar 6.3 Gerakan Simultan	68
Gambar 6.4 Code Gerakan Berjalan	69
Gambar 6.5 Prosedur Mengimbangi Satu Sama Lain	70
Gambar 6.6 Prosedur setVehicle 1	71
Gambar 6.7 Prosedur setVehicle 2	71
Gambar 6.8 Menggunakan Prosedur setVehicle	71
Gambar 6.9 Menghentikan Prosedur setVehicle	72
Gambar 7.1 Functions	73
Gambar 7.2 Memindahkan Objek	74
Gambar 7.3 Menentukan Jarak	74
Gambar 7.4 Arah dan Placeholder Jarak	75
Gambar 7.5 getDistanceTo ke Nilai Jarak	75
Gambar 8.1 Metode Panel	90
Gambar 8.2 Pendeklarasian Prosedur	90
Gambar 8.3 Metode dalam Java	91
Gambar 8.4 Sintaks Java	91

Gambar 8.5 Metode FindMax	92
Gambar 8.6 Kelas di Alice	93
Gambar 8.7 Kelas di Java	94
Gambar 8.8 Source code Membuat Kelas Cat	94
Gambar 8.9 Hasil Source Code	95
Gambar 8.10 Instances di Java	95
Gambar 8.11 Instance dari Java	95
Gambar 8.12 Source code IF control Structure	96
Gambar 8.13 Source code contoh	97
Gambar 8.14 Source code While control Structure	97
Gambar 8.15 Contoh While	98
Gambar 8.16 Source code Input dan Output	98
Gambar 9.1 Tab Fungsi	101
Gambar 9.2 Hasil Fungsi Jarak	101
Gambar 9.3 Fungsi Get Distance To	102
Gambar 9.4 Hasil Get Distance To	102
Gambar 9.5 Langkah – Langkah Get Distance to (a)	103
Gambar 9.6 Langkah – Langkah Get Distance to (b)	103
Gambar 9.7 Langkah – Langkah Get Distance to (c)	103
Gambar 9.8 Uji Fungsi	104
Gambar 9.9 Hasil Uji Fungsi	104
Gambar 9.10 Langkah-langkah menghindari tabrakan (a)	106
Gambar 9.11 Langkah-langkah menghindari tabrakan (b)	106
Gambar 9.12 Hasil menghindari Tabrakan	107
Gambar 9.13 Hasil Menghindari Tabrakan dengan waktu yang berbeda	107
Gambar 9.14 Menu Matematika	107
Gambar 9.15 Operator Tambahan (+) pada Menu Matematika	108
Gambar 9.16 Operator Matematika 2 Argumen	108
Gambar 9.17 Menghapus Object DEPTH	108
Gambar 9.18 Menghapus DEPTH	109
Gambar 10.1 Struktur Control	111
Gambar 10.2 Tampilan Struktur Kontrol	112

Gambar 10.3 Kode Struktur Kontrol Bersarang	113
Gambar 10.4 Struktur Kontrol IF	114
Gambar 10.5 Struktur IF	116
Gambar 10.6 Contoh Struktur Kontrol IF	117
Gambar 10.7 Conditiona Loop	117
Gambar 10.8 Infinite Loop	117
Gambar 10.9 Struktur Kontrol While	119
Gambar 10.10 Hasil Kontrol While	119
Gambar 10.11 Pengujian Kontrol While (a)	120
Gambar 10.12 Pengujian Kontrol While (b)	120
Gambar 10.13 Pengujian Kontrol While (c)	120
Gambar 11.1 Ekspresi dengan argumen jarak jauh	123
Gambar 11.2 Tampilan Argumen Fungsi	123
Gambar 11.3 Langkah Membuat Ekspresi	124
Gambar 11.4 Langkah Memindahkan Jarak (a)	124
Gambar 11.5 Langkah Memindahkan Jarak (b)	124
Gambar 11.6 Langkah Memindahkan Jarak dan Objek	125
Gambar 11.7 Langkah Memodifikasi Jarak	125
Gambar 11.8 Langkah Merubah Jarak	125
Gambar 11.9 Megubah Nilai Ekspresi	126
Gambar 11.10 Langkah Nilai Ekspresi	126
Gambar 11.11 Hasil Nilai Ekspresi	126
Gambar 11.12 Langkah Mengurangi Kedalaman Ekspresi	127
Gambar 11.13 Langkah Menafsirkan Ekspresi	127
Gambar 11.14 Langkah Membangun Ekspresi (a)	128
Gambar 11.15 Langkah Membangun Ekspresi (b)	128
Gambar 11.16 Langkah Membangun Ekspresi (c)	128
Gambar 11.17 Langkah Membangun Ekspresi (d)	128
Gambar 11.18 Langkah Membangun Ekspresi (e)	128
Gambar 11.19 Langkah Membangun Ekspresi (f)	128
Gambar 11.20 Langkah Membangun Ekspresi (g)	129
Gambar 11.21 Langkah Membangun Ekspresi (h)	129
Gambar 11.22 Hasil Ekspresi	129

Gambar 11.23 Langkah Interpretasi Ekspresi	129
Gambar 11.24 Hasil Ekspresi Jawaban	130
Gambar 12.1 Nilai Variabel dalmation	133
Gambar 12.2 Variabel Penyimpanan Data	133
Gambar 12.3 Properti Objek	134
Gambar 12.4 Langkah Mendeklarasikan Variabel	136
Gambar 12.5 Tampilan Insert Variabel	136
Gambar 12.6 Contoh Variabel (a)	136
Gambar 12.7 Contoh Variabel (b)	137
Gambar 12.8 Contoh Variabel (c)	137
Gambar 12.9 Langkah Menggunakan Variabel dalam Prosedur (a)	137
Gambar 12.10 Langkah Menggunakan Variabel dalam Prosedur (b)	138
Gambar 12.11 Langkah Variabel dalam Perhitungan Matematika	138
Gambar 12.12 Pengacakan Nilai yang diinisialisasi	139
Gambar 12.13 Menampilkan Kode Alice dalam bahasa Java (a)	140
Gambar 12.14 Menampilkan Kode Alice dalam bahasa Java (b)	140
Gambar 12.15 Menampilkan Kode Alice dalam bahasa Java (c)	141
Gambar 12.16 Menampilkan Kode Alice dalam bahasa Java (d)	141
Gambar 14.2 Storyboard	154
Gambar 14.3 Contoh storyboard Tekstual	156
Gambar 14.4 Animasi Checklist	158
Gambar 14.5 addDefaultModelManipulation	161
Gambar 14.6 Langkah Upload Animasi ke Youtube	162
Gambar 14.7 Tampilan Record layar	162
Gambar 14.8 Tampilan akhir record	163
Gambar 14.9 Ekspor Video to File	163
Gambar 15.1 Properti Objek	168
Gambar 15.2 Deklarasi Variabel dalam Alice	169
Gambar 15.3 Deklarasi Variabel Java	169
Gambar 15.4 Tipe Data di Alice	170

Gambar 15.5 Tipe Data di Alice	171
Gambar 15.6 Operator Aritmatika dalam argumen Jarak	173
Gambar 15.7 Operasi Aritmatika pada Java	173
Gambar 15.8 Operator relasional di Alice	174
Gambar 15.9 Contoh Operator Relational di Java	175
Gambar 15.10 Operator Logika dalam Alice	176
Gambar 15.11 Contoh Operator Logika dalam Alice	176
Gambar 15.12 Operator Logika di Java	177
Gambar 15.13 Operator Tugas di Alice	178
Gambar 15.14 Operator Penugasan di Java	178
Gambar 15.15 Kode Sintaks Penugasan	180

## **BAB I**

### **MENGENAL ALICE**

Pelajaran ini mencakup tujuan-tujuan berikut:

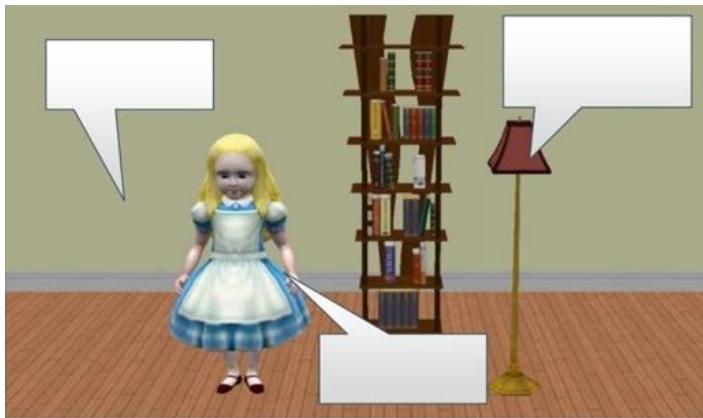
1. Mengenal Komponen Layar Alice
2. Membuat dan Menyimpan Proyek
3. Menambahkan Objek pada Layar
4. Mengkomunikasikan Nilai pada Proses Menyimpan untuk *Multiple* Layar
5. Membuat Kode Instruksi Sederhana
6. Menggunakan *Copy* dan *Undo* Pada Perintah
7. Mengenal Nilai Pengetesan pada *Testing and Debugging*

Alice bukanlah sebuah Bahasa pemrograman, bukan pula Bahasa Object Oriented Programming. Alice merupakan media pemrograman visual berbasis objek yang berguna untuk membantu pembelajaran mendefinisikan objek dan mengelola interaksi antara objek yang ditampilkan melalui animasi. Untuk mendapatkan Alice, anda dapat mengunjungi situs [www.Alice.org](http://www.Alice.org). Unduh Alice 3 yang kompatibel dengan sistem operasi yang anda miliki. Untuk memulai Alice, anda harus telah menginstal Alice 3.

#### **1.1 Adegan awal**

Adegan awal adalah titik awal dari animasi yang akan anda buat. Adegan ini anda akan memilih latar belakang dan posisi dari objek animasi. Sebuah adegan awal terdiri dari tiga komponen yaitu:

- a. Sebuah template latar belakang yang menyediakan langit, lantai, dan cahaya.
- b. Objek tidak bergerak yang menyediakan suatu kondisi.
- c. Objek bergerak beserta dengan aksinya.

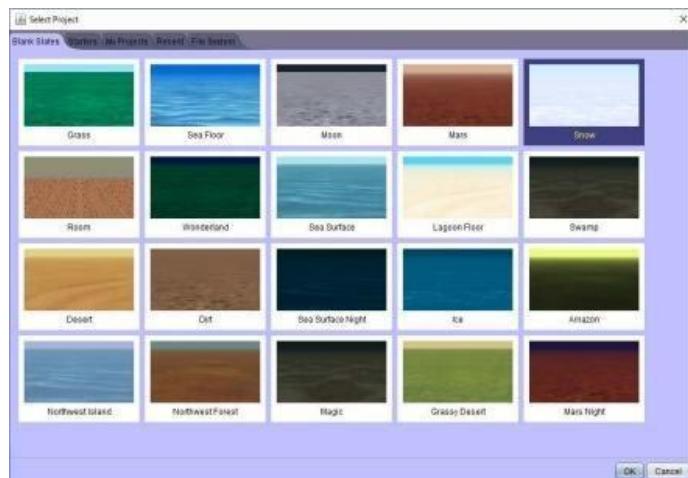


Gambar 1.1 Komponen dari adegan di sebuah ruangan

## 1.2 Langkah membuat projek baru

Langkah yang dilakukan dalam membuat projek baru adalah:

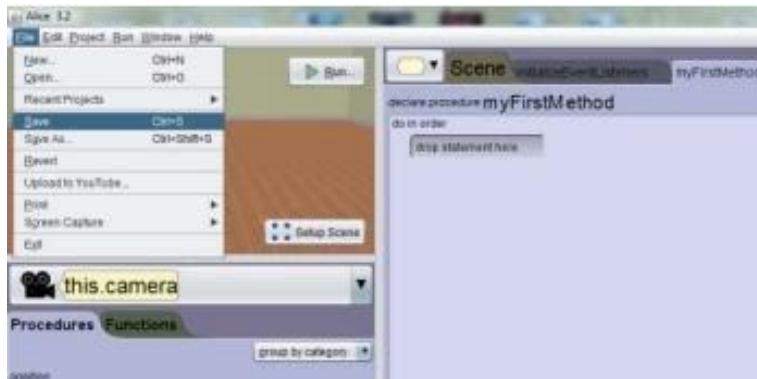
- Jalankan Alice 3
- Pada kotak dialog pilih halaman kosong
- Pilih sebuah template dan klik OK



Gambar 1.2 Memilih template

### **1.3 Cara menyimpan projek**

- a. Pilih menu File, kemudian klik Save As
- b. Pilih lokasi penyimpanan
- c. Buat nama projek
- d. Klik Save

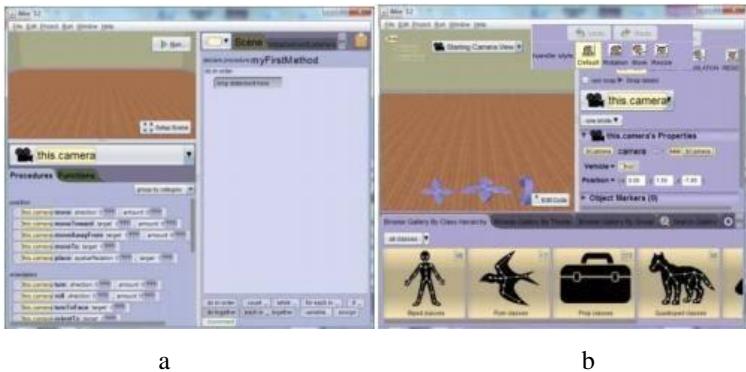


Gambar 1.3 Menyimpan projek

Lakukan Save (simpan) secara berkala untuk menghindari tidak tersimpannya pekerjaan anda.

### **1.4 Navigasi antara editor**

Alice menyediakan dua buah editor yang berbeda, disebut dengan Perspektif, sehingga anda dapat berpindah Tab sesuai keinginan anda. Dua editor tersebut adalah Code Editor yaitu Tab editor untuk melihat kode program yang posisinya terletak pada bagian kiri dan Tab Scene Editor untuk melihat pengaturan adegan yang terletak di bagian kanan.



a

b

Gambar 1.4 (a) Code Editor (b) Scene Editor

Secara umum Alice akan menampilkan Code Editor pada permulaan projek, untuk mengubah ke Scene Editor. Pilih tombol Setup Scene untuk mengganti ke Scene Editor.

### 1.5 Menambah sebuah objek pada sebuah halaman

Pada layer editor menambahkan sebuah objek pada layer salah satu caranya adalah:

1. Drag sebuah objek dari galeri ke dalam layar, kemudian lengkapi kotak dialog. Alice 3 akan menambahkan objek pada lokasi yang telah anda tentukan pada layar.
2. Klik objek, lengkapi kotak dialog. Kemudian Alice 3 akan menambahkan objek di tengah layar.



Gambar 1.5 Menambahkan objek pada layar

## 1.6 Memberikan nama objek

1. Periksa nama objek yang disediakan untuk objek
2. Ubahlah nama, atau klik OK untuk menyetujui nama dan menambahkan objek pada layar.



Gambar 1.6 Membuat nama objek

## 1.7 Editor Layar

Pada layar editor, anda dapat melakukan:

1. Memilih objek dari galeri untuk ditambahkan pada layar
2. Posisikan objek pada layar menggunakan Handle pallette
3. Ubah property objek menggunakan panel propertis
4. Mengakses code editor untuk menambahkan instruksi pemrograman
5. Jalankan animasi setelah instruksi pemrograman ditambahkan pada code editor.

## 1.8 Tampilan Editor Layar

Layar editor terdiri dari dua panel, yaitu:

1. Pengaturan layar pada bagian atas
2. Galeri pada bagian bawah



Gambar 1.7 Tampilan editor layar

## 1.9 Galeri

**Galeri adalah sekumpulan objek 3 dimensi yang dapat digunakan pada layar.** Galeri dikelompokkan pada beberapa tab, sehingga anda dapat menemukan objek dengan menggunakan tab pada galeri atau dengan memasukkan kata kunci pada fitur pencarian galeri. Galeri memiliki tab yang dijelaskan pada Tabel 1.1.

Tabel 1.1 Tab dan Fungsi pada Galeri

Tab	Fungsi
-----	--------

Jelajahi Galeri menurut Hirarki Kelas	Mengatur objek berdasarkan mobilitas.
Jelajahi Galeri berdasarkan Tema	Mengatur objek berdasarkan wilayah dan konteks cerita rakyat.
Jelajahi Galeri menurut Grup	Mengatur objek berdasarkan kategori.
Galeri Pencarian	Memungkinkan pencarian objek berdasarkan nama.
Shapes/Text	Mengatur bentuk objek, teks 3D, dan billboard.
My Classes	Memungkinkan Anda menambahkan kelas eksternal ke dalam proyek Anda

## 1.10 Select a class

Tab Hirarki Class mengelompokkan objek berdasarkan pada tipe mobilitas data (biped, flyer, dsb.).

*Kelas berisi instruksi yang menentukan penampilan dan pergerakan suatu objek. Semua objek di dalam kelas memiliki properti yang sama. Kelas tersebut memberikan instruksi kepada Alice 3 untuk membuat dan menampilkan objek saat ditambahkan ke adegan Anda.*

Kelas dapat terdiri dari sub-kelas, contohnya kelas Alice merupakan sub kelas dari Kelas Biped. Terdapat dua sub-kelas pada galeri dimana terlihat pada objek yang mewarisi semua sifat dari kelas objek Biped. Contohnya seperti memiliki dua kaki, memiliki kemampuan untuk bergerak.

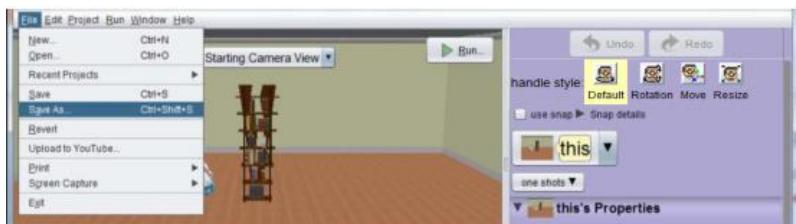
## 1.11 Menyimpan versi baru pada program

Menyimpan versi baru pada project dapat menghemat waktu dalam membuat versi yang beragam dari project yang dibuat pada Alice. Setelah memposisikan objek pada adegan awal, maka kita dapat menyimpan beberapa versi project sesuai dengan kebutuhan dan dengan nama yang berbeda.

Keuntungan dari menyimpan versi project yang beragam adalah kita dapat membuat animasi yang berbeda pada tema yang sama dan dapat menghemat waktu ketika membuat tema dalam mencegah eror pada pengkodingan.

Adapun langkah menyimpan suatu versi project adalah sebagai berikut:

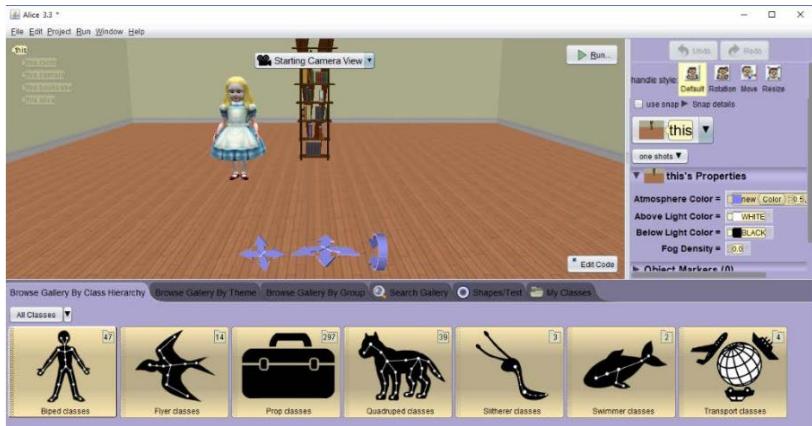
- a. Pada file yang terbuka, pilih Save As
- b. Pilih lokasi untuk menyimpan project
- c. Masukkan nama project
- d. Klik Save
- e. Jangan lupa untuk menyimpan proyek sesering mungkin untuk menghindari kehilangan data pada project.



Gambar 1.8 Tampilan Penyimpanan Proyek Alice

## 1.12 Editor kode

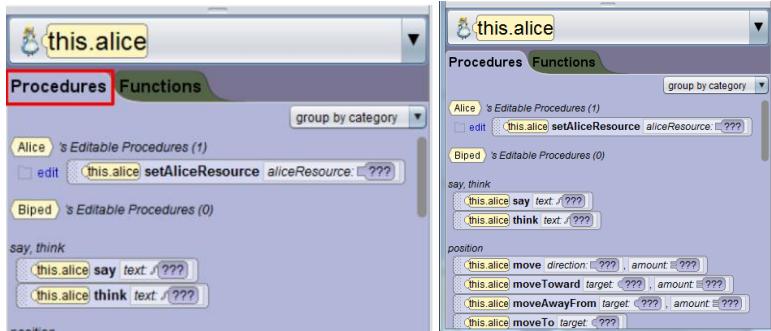
- a. Klik tombol Edit Code untuk menampilkan editor kode
- b. Tampilan Code editor adalah tempat untuk menambahkan instruksi program pada animasi.



Gambar 1.9 Tampilan Code Editor

### 1.13 Methods panel

Panel Method terdapat pada tab Procedures, berlokasi dalam Panel Method pada Code Editor, menampilkan method yang telah didefinisikan untuk memilih instan methods seperti methods yang telah tersedia pada objek dan kelas. **Procedure bisa diartikan sebagai potongan program yang menjelaskan tentang bagaimana sebuah objek dapat melakukan tugas yang ada pada Alice.** Program Alice 3 memiliki sebuah paket prosedur yang terdapat dalam kelas yang perlu dideklarasikan sebagai procedure baru.



Gambar 1.10 Tampilan Procedure dan Methods

## 1.14 Menu instance di panel metode

- Menu contoh ditampilkan di atas tab Prosedur.
- Segitiga penunjuk ke bawah di sisi kanan menu menunjukkan bahwa menu turun saat dipilih.

Untuk membuat prosedur yang baru, dapat di lihat dari Panel Metode, klik dan seret instruksi pemrograman yang diinginkan ke tab myFirstMethod pada editor Kode. Setelah memilih instruksi pemrograman ke tab myFirstMethod, gunakan menu berjenjang untuk memilih nilai untuk setiap argumen yang digunakan dalam metode ini.

*Argumen adalah nilai yang digunakan oleh metode untuk melakukan tindakan.*



Gambar 1.11(a) Instance Procedure(b) MyFisrtMethod

Jenis argumen yang tersedia pada program Alice mencakup arah, jumlah, durasi, dan teks. Alice 3 mengenali berapa banyak argumen yang dibutuhkan untuk setiap instruksi pemrograman. Pilihan jumlah dapat dipilih dari menu berjenjang untuk menentukan nilai untuk setiap argumen tersebut. Untuk menyalin instruksi pemrograman, dapat menggunakan salah satu dari metode ini:

- Metode **CTRL + Drag** menyalin kode.

- b. Klik kanan dan gunakan opsi Salin ke Papan Klip untuk menyalin kode.
- c. Klik dan seret instruksi pemrograman ke clipboard untuk memindahkan kode.

Langkah-langkah untuk Menggunakan Metode CTRL + Drag:

1. Tahan tombol CTRL pada keyboard Anda.
2. Klik dan tahan tanda instruksi pemrograman.
3. Seret tanda ke lokasi yang diinginkan dalam kode, atau ke papan klip.
4. Lepaskan tombol mouse sebelum melepaskan tombol CTRL.

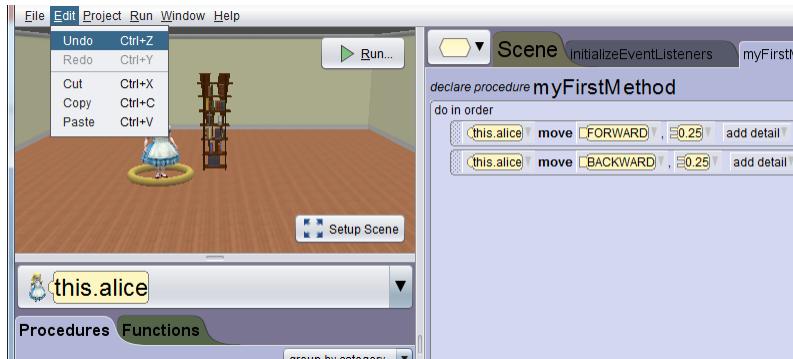
Langkah-langkah untuk Menggunakan Metode Salin Klik Kanan:

1. Klik kanan pada pegangan instruksi pemrograman.
2. Pilih opsi Salin ke Papan Klip.

Langkah-langkah untuk Menggunakan Metode Klik + Drag Ke Papan Klip:

1. Klik dan seret instruksi pemrograman ke ikon clipboard.
2. Papan klip berubah warna saat penunjuk mouse melakukan kontak dengan ikon papan klip.
3. Gunakan metode ini saat memindahkan instruksi pemrograman antar tab.
4. Papan klip dapat menyimpan beberapa instruksi pemrograman pada saat yang bersamaan.
5. Kode ditarik dari clipboard dengan urutan yang berlawanan dengan urutan penempatannya.
6. Papan klip menampilkan jumlah item yang disimpan di dalamnya.

Batalkan tindakan menggunakan opsi Batalkan pada menu Edit, atau pintasan keyboard CTRL + Z.



Gambar 1.12 Menu Pembatalan

### 1.15 Uji dan debug pengujian animasi anda

Setelah membuat instruksi pemrograman untuk animasi, program perlu diuji. Untuk menguji program, klik tombol Run. Jalankan animasi untuk menguji apakah berfungsi dengan baik dan dijalankan sesuai rencana dan tanpa kesalahan. Uji animasi sesering mungkin selama masa pengembangan. Menguji batasan program adalah bagian penting dari proses tersebut. Misalnya, mengubah nilai argumen dalam prosedur untuk sengaja "merusak" kode membuktikan kode berfungsi dalam kondisi ekstrim. Apa yang terjadi jika angkanya sangat besar? atau negatif?

Debugging program mengacu pada siklus yang melibatkan pengujian program, mengidentifikasi kesalahan atau hasil yang tidak diinginkan, Menulis ulang kode dan menguji ulang.

*Program perangkat lunak, seperti animasi, diuji dengan memasukkan perintah yang tidak terduga untuk mencoba dan "memecahkan" kode. Ketika ada sesuatu yang rusak atau tidak berfungsi sebagaimana mestinya dalam program perangkat lunak, hal itu sering disebut sebagai "bug". Debugging adalah proses menemukan bug dalam program perangkat lunak.*

Gunakan beberapa teknik berikut saat memprogram animasi di Alice 3:

1. Sesuaikan argumen yang menentukan arah, jarak, dan durasi pergerakan objek.
2. Sesuaikan ekspresi matematika yang memanipulasi arah, jarak, dan durasi pergerakan objek.
3. Perbaiki atau ganti instruksi dalam kode yang tidak berfungsi sebagaimana mestinya.
4. Mengatasi kesalahan yang dibuat oleh programmer.

## **BAB II**

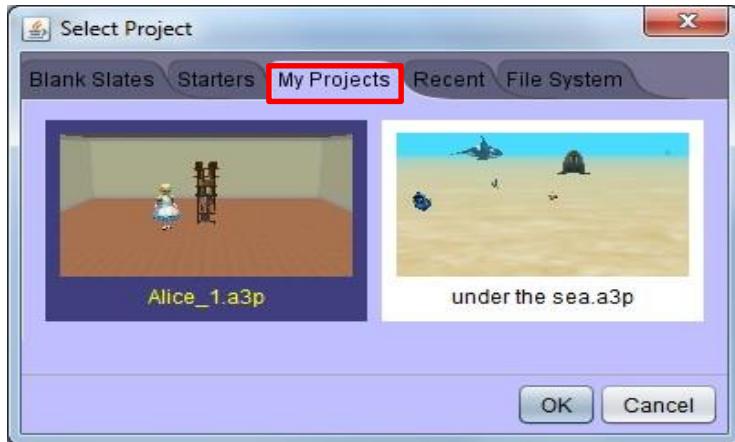
### **MENAMBAH DAN MEMPOSISIKAN OBJEK**

Pelajaran ini mencakup tujuan-tujuan berikut:

1. Membuka proyek yang disimpan
2. Menambahkan beberapa objek ke sebuah adegan
3. Menjelaskan perbedaan antara penentuan posisi yang tepat dan posisi *drag-and-drop* (atau tidak tepat)
4. Menggunakan prosedur *one-shot* untuk memposisikan objek dalam adegan dengan tepat
5. Merubah properti suatu objek di *scene editor*
6. Menjelaskan sumbu dalam pemosisian tiga dimensiMemposisikan sub-bagian dari suatu objek di *scene editor*

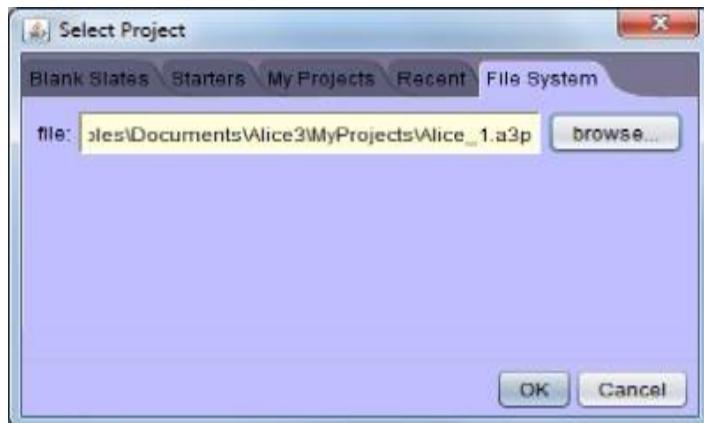
#### **2.1 Membuka proyek yang telah ada**

1. File proyek Alice 3 yang disimpan dapat dibuka dan diubah
2. Ada dua cara untuk membuka *file* proyek Alice 3 yang sudah ada setelah meluncurkan Alice 3:
  - a. Pilih proyek dari tab “*My Project*”
  - b. Atau pilih proyek menggunakan tab “*File System*”
3. Langkah-langkah untuk Membuka Proyek yang Ada Menggunakan Tab “*My Project*”
  - a. Buka Alice 3
  - b. Dari kotak dialog “*Select Project*”, pilih tab “*My Projects*”.
  - c. *Scroll* ke nama atau gambar kecil proyek yang akan dibuka
  - d. Pilih nama atau *thumbnail* proyek dan klik OK.



Gambar 2.1 Membuka Project

- e. Buka Alice 3.
- f. Dari kotak dialog Pilih Proyek, pilih tab “Sistem File”.



Gambar 2.2 Mencari Project

- g. Pilih tombol *browse*
- h. Gunakan *windows* navigasi untuk menavigasi ke struktur direktori di komputer Anda di mana *file* Alice 3 berada

Catatan: Alice 3 tidak dapat membuka animasi yang dibuat dengan versi Alice sebelumnya.

- i. Klik OK setelah file Alice 3 dipilih.

## 2.2 Memposisikan objek dalam adegan awal

Memposisikan objek dalam adegan awal termasuk memilih:

1. Arah yang harus dihadapi objek
2. Orientasi objek yang relatif terhadap objek lain dalam adegan
3. Posisi objek dalam adegan
4. Posisi sub-bagian objek (lengan, kaki, dan lain-lain)

## 2.3 Fitur pemosisan untuk objek

Semua objek Alice 3 berbagi fitur pemosisan yang sama:

1. Koordinat 3D pada sumbu x, y dan z.
2. Titik pusat, tempat sumbunya bersinggungan (biasanya di pusat massa).
3. Subbagian yang bisa bergerak



Gambar 2.3 Sumbu X, Y, Z Objek

4. Objek dan sub-bagiannya bergerak relatif terhadap orientasi mereka sendiri, atau sesuai arah.
5. Objek yang menghadap ke belakang adegan, diprogram untuk bergerak maju 2 meter, bergerak 2 meter lebih jauh ke arah belakang adegan.



Gambar 2.4 Pergerakan Objek

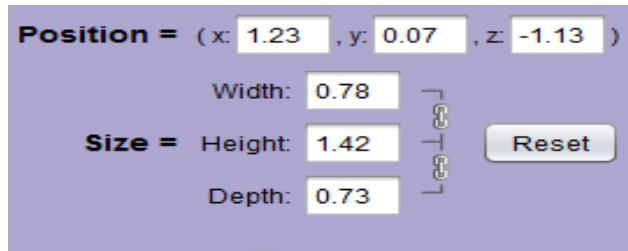
## 2.4 Cara untuk memosisikan objek

Ada dua cara untuk memosisikan objek:

1. Penentuan posisi yang tepat menggunakan salah satu dari dua metode dibawah:
  - a. Gunakan prosedur “*one shot*”.
  - b. Masukkan nilai untuk koordinat x, y, dan z dan tekan tombol *Enter*.
2. Penentuan posisi yang tepat menggunakan metode *drag-and-drop* dengan kursor.



Gambar 2.5 Nama Objek



Gambar 2.6 Posisi X, Y, Z Objek



Gambar 2.7 Hasil Pengaturan Posisi Objek

## 2.5 Memilih contoh untuk posisi

1. Ada dua cara untuk memilih contoh yang ingin diposisikan

- a. Klik nama *instance* dari daftar objek di sudut kiri atas *window “scene”*
  - b. Klik contoh di *window “scene”*
2. Lingkaran atau panah mengelilingi objek setelah objek dipilih



Gambar 2.8 Lingkaran Objek

## 2.6 Prosedur *one shot* posisi yang tepat

Prosedur adalah seperangkat instruksi atau kode yang diprogram untuk bagaimana objek harus melakukan tugas. Prosedur *one-shot* tersedia di *editor scene*. Prosedur *One-shot*:

1. Digunakan untuk melakukan penyesuaian pandangan dan memposisikan objek.
2. Tidak dieksekusi ketika tombol Run diklik untuk memutar animasi.

## 2.7 Posisi objek di adegan – Posisi yang tepat

Setelah menambahkan beberapa objek ke tengah adegan, gunakan prosedur *one-shot* untuk menempatkannya secara tepat di lokasi yang berbeda dalam adegan sehingga semua objek terlihat.



Gambar 2.9 Posisi Objek Awal

Langkah-langkah untuk membuka menu prosedur *one-shot* posisi yang tepat adalah:

1. Klik kanan pada objek di *editor scene*
2. Pilih prosedur  
ATAU
3. Pilih menu prosedur *one-shot* di panel *Properties*
4. Pilih prosedur



Gambar 2.10 Penggunaan *One-shot*

5. Dari menu prosedur *one-shot*, pilih prosedur yang diinginkan.
6. Tentukan nilai argumen (arah dan jarak)

7. Objek akan secara otomatis memposisikan ulang berdasarkan prosedur yang dipilih dan argumen yang ditentukan.



Gambar 2.11 Prosedur Pada Nilai Argumen

Terkadang lebih mudah untuk menempatkan banyak *instance* di tengah adegan dengan *Positioning*.



Gambar 2.12 Contoh Penggunaan *Positioning*

1. Jika ingin menambahkan tiga karakter kartu bermain ke adegan, kita dapat menambahkannya ke tengah layar terlebih dahulu.
2. **Pindahkan objeknya dengan prosedur *one-shot* daripada mencoba *drag-and-drop* ke lokasi yang benar di dalam ruangan.**

## 2.8 Properti posisi – Posisi yang tepat

Properti *Position* memberi tahu kita di mana objek diposisikan dalam adegan pada sumbu x, y, dan z.



Gambar 2.13 Posisi Objek Menggunakan Koordinat

Langkah-langkah untuk memposisikan objek menggunakan koordinat adalah sebagai berikut:

1. Di panel Properti, temukan properti Posisi
2. Masukkan nilai dalam sumbu x dan tekan enter
3. Masukkan nilai dalam sumbu y dan tekan enter
4. Masukkan nilai dalam sumbu z dan tekan enter
5. Objek akan secara otomatis memposisikan ulang ke koordinat yang dimasukkan dalam bidang sumbu x, y, dan z.
6. Kita harus menekan tombol enter setelah memasukkan setiap nilai.



Gambar 2.14 Posisi X, Y, Z

Langkah-langkah untuk menggunakan metode posisi yang tepat adalah sebagai berikut:

1. Pilih objek atau sub-bagian dari objek menggunakan kursor
2. Pilih *handle style*
3. Setiap *handle style* menyajikan lingkaran atau panah untuk membantu menentukan posisi
4. Misalnya, *Move handle style* akan menyajikan tiga panah untuk digunakan dalam memposisikan objek di sepanjang sumbu x, y, dan z.
5. Posisikan objek dengan kursor yang memilih dan menarik lingkaran yang mengelilingi objek.



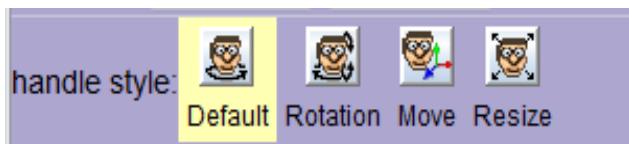
Gambar 2.15 Menentukan Posisi Objek Menggunakan Lingkaran dan Panah

Jenis-jenis *Handle Style* dapat dilihat pada tabel 2.1.

Tabel 2.1 Jenis-jenis *Handle Style*

<b>Handle Style</b>	<b>Keterangan</b>
---------------------	-------------------

<i>Default</i>	Rotasi dan gerakan sederhana
<i>Rotation</i>	Putar tentang sumbu x, y, dan z.
<i>Move</i>	Bergeraklah sepanjang sumbu x, y, dan z.
<i>Resize</i>	Ubah ukuran objek dan rentangkan sepanjang sumbu x, y, dan z. Catatan: Jika kita memilih objek yang sangat besar dalam adegan, dan kemudian memilih <i>handle</i> Ubah Ukuran, panah pemosisan yang membantu mengubah ukuran objek mungkin muncul dari layar. Jika ini terjadi, pilih <i>handle</i> Ubah Ukuran dan kemudian gunakan tombol gulir pada mouse Anda untuk mengubah ukuran objek.



Gambar 2.16 Penggunaan Handle Style

Menu **properties** objek terpilih dalam *editor scene* menyediakan kemampuan untuk mengubah properti *instance* yang saat ini dipilih dalam adegan.



Gambar 2.17 Properti Objek

## 2.9 Memodifikasi properti

Properti dapat dimodifikasi selama pengaturan adegan dan selama eksekusi animasi. Sebagai contoh:

1. Ubah properti *Opacity* dari *instance* ke 0 selama pengaturan adegan untuk membuatnya menghilang.
2. Buat pernyataan pemrograman untuk mengatur properti *Opacity* dari *instance* ke 1 sehingga *instance* muncul kembali di adegan selama eksekusi animasi.



Gambar 2.18 *Opacity* Properti

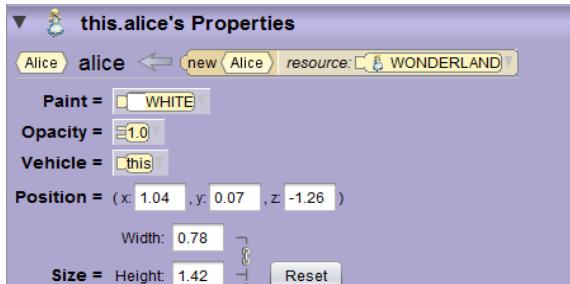
Langkah-langkah untuk mengubah properti *instance* di *scene editor* adalah:

1. Pilih *instance* di *editor scene*
2. Ubah properti seperti cat (warna), *opacity*, atau ukuran menggunakan menu di panel objek.



Gambar 2.19 Pengubahan Properti

3. Segitiga di depan tombol properti objek dipilih dapat menyembunyikan atau menampilkan properti dari instance.
4. Jika bidang properti untuk instance tidak ditampilkan, klik segitiga untuk menampilkannya.



Gambar 2.20 Menampilkan Properti *Instance*

## 2.10 Memosisikan sub-bagian objek

Gunakan *handle style* untuk memposisikan sub-bagian objek selama pengaturan pemandangan. Sebagai contoh: kita mungkin ingin kepala objek melihat ke kanan ketika animasi dimulai.



Gambar 2.21 Mengubah Arah Kepala Objek

Langkah-langkah untuk memposisikan sub-bagian objek adalah sebagai berikut:

1. Pilih contoh dari menu properti objek
2. Segitiga pengarah kanan di sebelah nama *instance* menunjukkan bahwa ada menu tambahan untuk sub-bagian *instance*
3. Pilih sub-bagian yang ingin diposisikan
4. Gunakan cincin di sekitar sub-bagian untuk memposisikannya
5. Ulangi langkah ini untuk memposisikan sub-bagian tambahan
6. Gunakan fitur *undo* bila perlu



Gambar 2.22 Posisi Sub-bagian Objek

## 2.11 Terminologi

Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:

1. Orientasi

2. Prosedur *One-shot*
3. Prosedur
4. Properti

## 2.12 Ringkasan

Dalam pelajaran ini, kita seharusnya belajar bagaimana caranya:

1. Buka versi proyek yang disimpan
2. Tambahkan beberapa objek ke sebuah adegan
3. Jelaskan perbedaan antara penentuan posisi yang tepat dan posisi *drag-and-drop* (atau tidak tepat)
4. Gunakan prosedur *one-shot* untuk memposisikan objek dalam adegan dengan tepat
5. Edit properti suatu objek di *editor scene*
6. Jelaskan sumbu pemosisian tiga dimensi
7. Posisikan sub-bagian dari suatu objek di *editor scene*

### BAB III

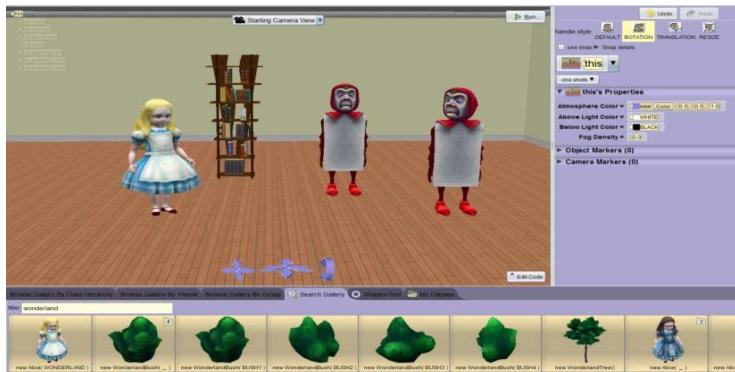
## PROSEDUR DAN ARGUMEN

Pelajaran ini mencakup tujuan-tujuan berikut:

1. Berilah antara *editor scene* dan *editor code*, serta menjelaskan perbedaan visual antara *editor scene* dan *editor code*.
2. Menemukan dan menjelaskan tujuan panel metode dan tab “procedures”
3. Menggunakan *procedures* untuk memindahkan objek
4. Tambahkan prosedur pemrograman Java ke *editor code*
5. Menunjukkan bagaimana nilai prosedur dapat diubah
6. Membuat komentar pemrograman
7. Susun ulang, *edit*, hapus, salin, dan nonaktifkan pernyataan pemrograman
8. Uji dan *debug* animasi

### 3.1 Menampilkan *code editor*

1. Klik *Edit Code* (dari *scene editor*) untuk menampilkan *code editor*

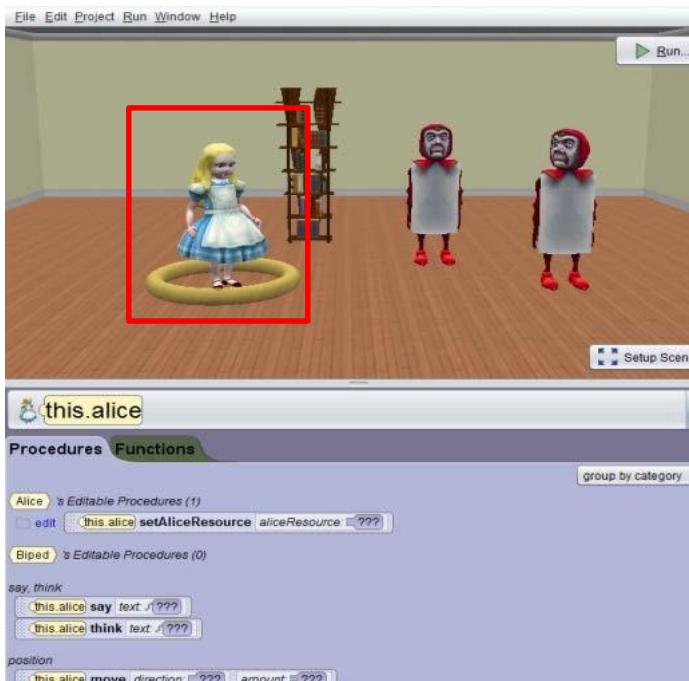


Gambar 3.1 Tampilan Awal

2. *Code editor* adalah tempat untuk memprogram animasi yang diinginkan

### 3.2 Pilih Instance

- Pertama, pilih *instance* yang ingin diprogram.



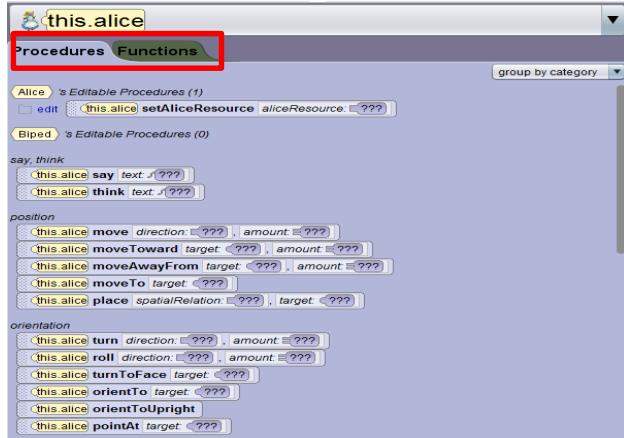
Gambar 3.2 Membuat *Instance*

- Pilih suatu *instance* di *small window scene* atau dengan menggunakan menu *pull down* di bawah *small window scene*.

### 3.3 Panel metode

**Panel Metode** berisi dua tab:

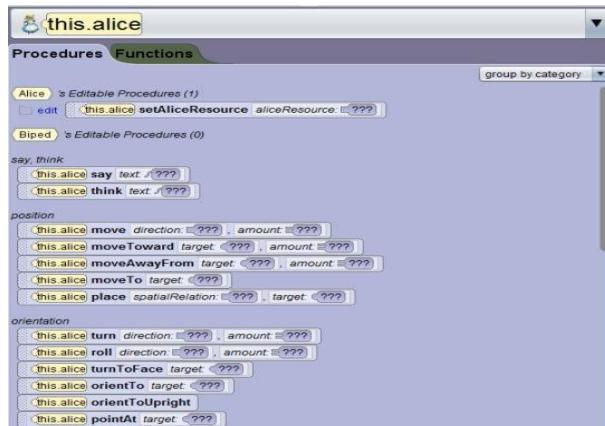
- Procedures*: Semua prosedur yang telah ditentukan untuk suatu objek
- Functions*: Semua fungsi yang ditentukan sebelumnya untuk objek



Gambar 3.3 Procedures dan Functions

### Procedures Tab

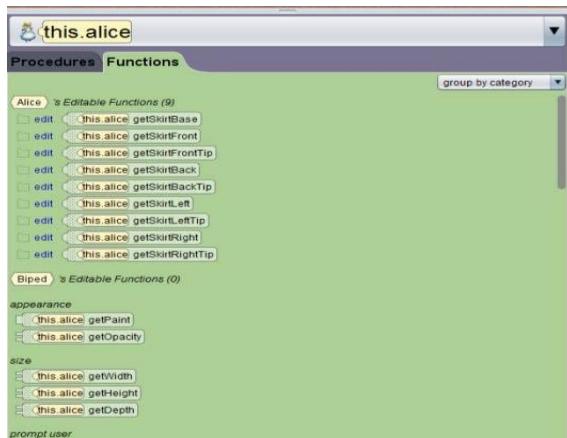
Prosedur adalah bagian dari kode program yang menentukan bagaimana objek harus menjalankan tugas. Alice 3 memiliki serangkaian prosedur untuk setiap kelas, namun pengguna dapat membuat (“declare”) prosedur baru. Tab prosedur menampilkan prosedur yang telah ditentukan untuk *instance* yang dipilih, serta prosedur kita sendiri yang ingin ditetapkan.



Gambar 3.4 Tab Prosedur

### Functions Tab

Suatu fungsi menghitung dan menjawab pertanyaan tentang suatu objek, seperti, "Berapa lebar atau tingginya?", Atau "Berapa jaraknya dari objek lain?" Alice 3 memiliki serangkaian fungsi untuk setiap kelas, namun pengguna dapat mendeklarasikan fungsi baru. *Tab Fungsi* menampilkan fungsi yang telah ditentukan untuk *instance* yang dipilih, serta fungsi kita sendiri yang ingin ditetapkan.



Gambar 3.5 Tab Fungsi



### 3.4 Code editor tabs

1. Menu Kelas ditampilkan di sebelah kiri *scene tab*
2. Pengguna membuat instruksi pemrograman pada *tab myFirstMethod*
3. Secara *default*, Alice membuat statement kontrol *do in order* di prosedur *myFirstMethod*.
4. Area berlabel pernyataan penurunan di sini adalah lokasi di mana pengguna akan menempatkan instruksi pemrograman.



Gambar 3.6 *Code Editor Tab*

### 3.5 Control statements

Di bagian bawah tab *myFirstMethod* akan ditemukan *control statements* Alice 3.

### 3.6 Gerakan objek

1. Gerakan objek bersifat **egosentrис, yaitu objek bergerak berdasarkan arah yang dihadapi.**
2. Suatu objek dapat bergerak dalam enam arah, yaitu naik, turun, ke depan, ke belakang, kanan dan kiri.

Contoh prosedur gerakan adalah:

Tabel 3.1 Contoh Prosedur Gerakan

Prosedur	Keterangan
<i>Move</i>	Memindahkan objek ke salah satu dari enam arahnya
<i>Move Toward</i>	Memindahkan objek ke objek lain
<i>Move Away From</i>	Memindahkan objek dari objek lain

<b>Move To</b>	Memindahkan objek dari posisi saat ini ke titik tengah objek target
<b>Move and Orient To</b>	Memindahkan objek dari posisi saat ini ke titik tengah objek target dan menyesuaikan orientasi objek agar sesuai dengan orientasi objek target
<b>Delay</b>	Menghentikan pergerakan objek selama beberapa detik. Delay dapat digunakan untuk memperlambat eksekusi suatu animasi.
<b>Say</b>	Membuat gelembung info dengan teks yang membuat objek tampak berbicara

Contoh *rotation procedures* adalah:

Tabel 3.2 Contoh Prosedur Rotasi

Prosedur	Keterangan
<b>Turn</b>	Putar objek ke kiri, kanan, maju, atau mundur pada titik tengahnya. Belok kiri dan kanan pada sumbu vertikal objek; maju dan mundur pada sumbu horizontal objek.
<b>Roll</b>	Memutar objek ke kiri atau kanan pada titik tengahnya menggunakan sumbu horizontal objek.

### 3.7 Membuat instruksi pemrograman

Dari tab *procedures*, klik dan seret prosedur yang diinginkan ke *myFirstMethod* di *code editor*.



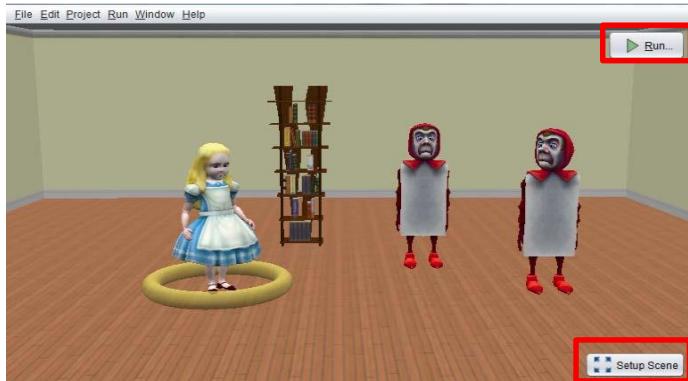
Gambar 3.7 Intruksi Pemrograman

1. Pilih dan tetapkan nilai argumen
  - a. Setelah membuat pernyataan pemrograman, gunakan menu *drop-down* untuk mengatur nilai untuk setiap argumen.
  - b. Untuk mengakses menu *drop-down* argumen, klik pada segitiga yang mengarah ke bawah di sebelah kanan nilai argumen.



Gambar 3.8 Nilai Argumen

2. Jalankan Program
  - a. Klik tombol *Run* untuk menjalankan instruksi pemrograman
  - b. Jalankan program sesering mungkin untuk menguji hasil yang diinginkan dan ubah nilai argumen yang diperlukan.

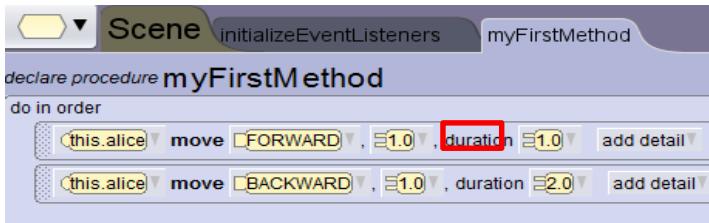


Gambar 3.9 Menjalankan Program

3. Argumen

Argumen adalah nilai yang digunakan prosedur untuk menyelesaikan tugasnya. Program komputer menggunakan argumen untuk memberi tahu cara menerapkan prosedur.

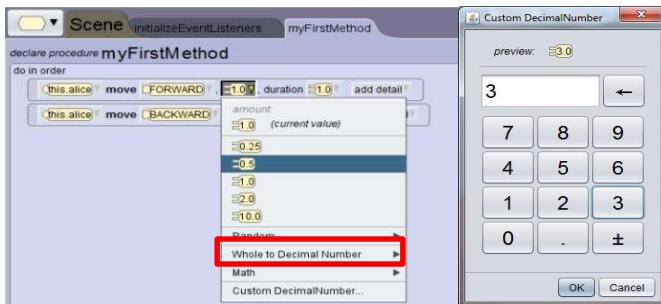
- a. Argumen dipilih setelah prosedur diletakkan ke *code editor*.
- b. Jenis argumen dapat meliputi: objek, arah, jumlah arah, durasi waktu dan teks.



Gambar 3.10 Penggunaan Argumen

### 3.8 Menu argumen

1. Menu argumen menawarkan nilai argumen *default* untuk dipilih
2. Jika tidak ada *default* yang cocok, pilih opsi **Custom DecimalNumber...** sehingga pengguna dapat menentukan nilai argumen yang lebih akurat.



Gambar 3.11 Penggunaan *DecimalNumber*

### 3.9 Argumen sebagai *placeholder*

1. Ketika prosedur diletakkan ke *code editor*, semua nilai argumen harus telah ditentukan.
2. Ada saatnya pengguna memilih nilai argumen sebagai nilai pengganti sementara yang akan diganti nanti.
3. Misalnya, pengguna mungkin ingin objek bergerak maju tetapi tidak yakin seberapa jauh.
4. Pilih nilai *placeholder* 2 meter, jalankan animasi, tentukan bahwa diperlukan nilai yang berbeda, lalu ubah nilainya.

- Pengguna juga bisa menentukan nilai *placeholder* yang akan pengguna ganti nanti dengan fungsi atau variabel.

### 3.10 Langkah-langkah untuk mengatur ulang statement pemrograman

- Gunakan "drag and drop", pilih pegangan pada *statement* pemrograman.
- Seret pernyataan pemrograman ke posisi barunya
- Letakkan pernyataan pemrograman ke posisi barunya dengan *de-select holder*.\*

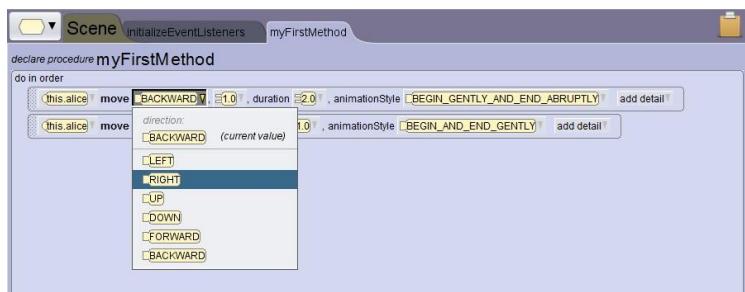


Gambar 3.12 Mengatur Ulang Statement Pemrograman

\* Catatan: Garis indikator posisi hijau akan muncul untuk membantu menyelaraskan *statement* pemrograman ke posisi yang diinginkan

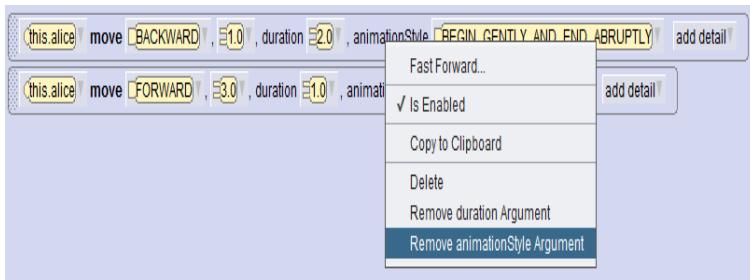
### 3.11 Edit statement pemrograman

Gunakan daftar *drop-down* untuk mengedit nilai-nilai *statement* pemrograman.



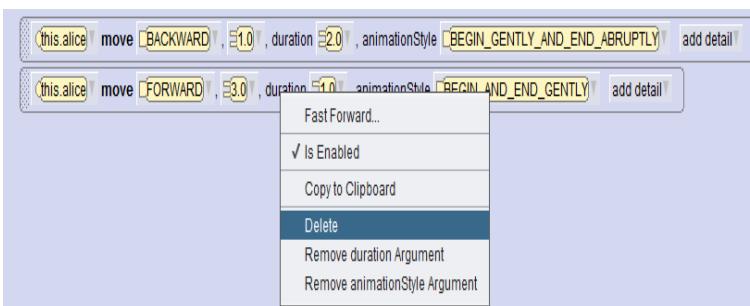
Gambar 3.13 Mengubah Statement Pemrograman

### 3.12 Menghapus *statement* pemrograman



Gambar 3.14 Menghapus Bagian *Statement* Pemrograman

Klik kanan *statement* pemrograman untuk menghapus bagian dari *statement* itu atau hapus seluruh *statement* dengan cara yang ada di gambar 3.15.



Gambar 3.15 Menghapus Seluruh *Statement* Pemrograman

### 3.13 Edit dan uji program



Gambar 3.16 Ubah dan Uji Program

1. Jalankan animasi untuk mengujinya, dan *edit* kode yang diperlukan.
2. Mungkin diperlukan beberapa siklus pengujian dan pengubahan untuk menjalankan animasi sesuai keinginan.

### 3.14 Debugging dan pengujian



Gambar 3.17 Debugging dan Pengujian

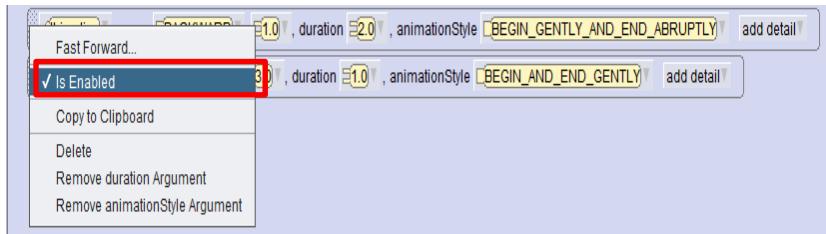
1. *Debugging* dan pengujian adalah proses menjalankan animasi berkali-kali, dan menyesuaikan *statement* kontrol, prosedur dan argumen setelah setiap eksekusi.
2. Selalu simpan saat *debug* program.

### 3.15 Masukkan **statement pemrograman sementara untuk membantu debugging**

1. Pengguna dapat memasukkan *statement* pemrograman sementara ke dalam kode untuk membantu *debugging*. Contoh: bayangkan kita memiliki objek yang tidak bergerak maju.
2. Untuk sementara, masukkan *statement* pemrograman *say* yang mengumumkan bahwa objek akan bergerak maju.
3. Uji program untuk melihat apakah pernyataan pemrograman *say* dijalankan atau tidak
4. Jika pernyataan *say* dijalankan tetapi objek tidak bergerak, ini menunjukkan satu jenis masalah.
5. Jika pernyataan *say* dan *move* tidak dieksekusi, ini mungkin mengindikasikan jenis masalah lain.

### 3.16 Menonaktifkan pernyataan pemrograman

1. *Statement* pemrograman dapat dinonaktifkan di *code editor*
2. Nonaktifkan pernyataan pemrograman untuk:
  - a. Membantu mengisolasi bagian-bagian kode selama pengujian.
  - b. Membantu fokus pada pemrograman, pengujian dan *debugging* instruksi tertentu.
3. Langkah-langkah untuk menonaktifkan *statement* pemrograman adalah:
  - a. Klik kanan pada pernyataan pemrograman
  - b. Batalkan pilihan "Is Enabled" dari daftar *drop-down*



Gambar 3.18 Membatalkan *Is Enabled*

- c. Warna pernyataan pemrograman akan berubah menjadi abu-abu tanda *hash* untuk menunjukkan bahwa itu dinonaktifkan



Gambar 3.19 Hasil Penonaktifan *Is Enabled*

### 3.17 Mengaktifkan kembali *statement* pemrograman yang dinonaktifkan

Langkah-langkah untuk mengaktifkan kembali statement pemrograman yang dinonaktifkan adalah:

1. Klik kanan pada pernyataan pemrograman yang telah dinonaktifkan
2. Pilih “*Is Enabled*” dari daftar drop-down



Gambar 3.20 Mengaktifkan *Is Enabled*

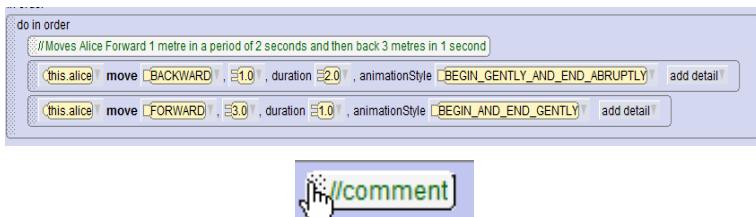
3. Ini mengaktifkan kembali kode dan menghilangkan garis abu-abu



Gambar 3.21 Hasil Pengaktifkan *Is Enabled*

### 3.18 Komentar Pemrograman

1. Termasuk komentar pemrograman dalam animasi membantu manusia memahami aliran pemrograman
2. **Komentar:**
  - a. Menjelaskan maksud dari instruksi pemrograman
  - b. Jangan memengaruhi fungsionalitas program karena hal itu diabaikan selama pelaksanaannya
  - c. Biasanya ditempatkan di atas blok pernyataan pemrograman yang dijelaskan
  - d. Sering ditulis pertama kali, dalam program besar sebagai garis besar instruksi pemrograman
3. Langkah-langkah untuk memasukkan komentar dalam program
  - a. Seret dan lepas komentar di atas segmen kode
  - b. Komentar terletak di bagian bawah *myFirstMethod*
  - c. Ketik komentar yang menggambarkan urutan tindakan di segmen kode.



Gambar 3.22 Memasukkan Komentar

### 3.19 Menggunakan Komentar untuk Mengatur Program

1. Komentar dapat menjadi alat yang sangat baik untuk mengatur pengembangan program
2. Untuk program besar, buat komentar yang menunjukkan *End of Program*
3. Komentar “*End of Program*” membantu meminimalkan penguliran saat menambahkan *statement* pemrograman ke *myFirstMethod* yang panjang.



Gambar 3.23 Menggunakan Komentar untuk Mengatur Program

### 3.20 Terminologi

Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:

1. Argumen
2. *Code Editor*
3. Komentar
4. Fungsi
5. Panel metode
6. Orientasi
7. Prosedur

### 3.21 Ringkasan

Dalam pelajaran ini, kita seharusnya belajar bagaimana:

1. Berilah antara *editor Scene* dan *Code editor* dan menjelaskan perbedaan visual antara *editor Scene* dan *code editor*
2. Temukan dan jelaskan tujuan panel metode dan *tab procedure*
3. Gunakan prosedur untuk memindahkan objek

4. Tambahkan prosedur pemrograman Java ke *Code editor*
5. Menunjukkan bagaimana nilai prosedur dapat diubah
6. Membuat komentar pemrograman
7. Susun ulang, ubah, hapus, salin, dan nonaktifkan *statement* pemrograman
8. Uji dan *debug* animasi

## **BAB IV**

### **ROTASI DAN PENGACAKAN**

Pelajaran ini meliputi tujuan-tujuan sebagai berikut:

1. Mengkorelasikan pernyataan *storyboard* dengan tugas pelaksanaan program
2. Menambahkan pernyataan kontrol ke *code editor*
3. Menggunakan angka-angka acak untuk mengacak gerakan

#### **4.1 Pengembangan program menggunakan pendekatan *Top-Down***

1. Sama seperti *homebuilder* menggunakan seperangkat *Blueprint* dan mengikuti serangkaian langkah, seorang *programmer* menggunakan rencana dan mengikuti serangkaian langkah untuk membangun sebuah program.
2. *Homebuilders* bekerja dengan tujuan dan menetapkan tujuan spesifik saat mereka membangun — kamarnya hemat energi, rumah memenuhi kode bangunan, dan lain-lain.
3. *Programmer* juga bekerja dengan tujuan dan menetapkan tujuan, karena tanpa hal itu, kesuksesan tidak dapat diukur.
4. Istilah lain untuk sasaran atau tujuan animasi adalah skenario tingkat tinggi - sebuah cerita dengan tujuan.

#### **4.2 Langkah-langkah menggunakan pendekatan *Top-Down* untuk pemrograman**

1. Tentukan skenario tingkat tinggi (maksud/tujuan program)
2. Dokumentasikan tindakan langkah demi langkah dalam *Storyboard textual*.
3. Ini membantu untuk mendapatkan pemahaman menyeluruh tentang tindakan yang perlu diprogram

4. Buat tabel untuk menyelaraskan langkah-langkah *Storyboard* dengan instruksi pemrograman
5. Tinjau tabel selama pengembangan animasi untuk memastikan animasi memenuhi spesifikasi
6. Merevisi tabel yang diperlukan

#### 4.3 Contoh **textual storyboard**

Program tindakan sesuai dengan perintah:

1. Kelinci masuk
2. Alice berbalik untuk melihat Kelinci
3. Kelinci berkata, "Apakah ini pestanya!"
4. Alice berkata, "Oh, tidak!"
5. Kelinci belok ke kiri
6. Program tindakan yang dilakukan bersama:
  - a. Kelinci berjalan pergi dengan cepat
  - b. Alice menggelengkan kepalanya



Gambar 4.1 Penggunaan *Textual Storyboard*

#### 4.4 Sejajarkan *storyboard* dengan instruksi pemrograman

Tabel 4.1 *Storyboard* dengan Instruksi Pemrograman

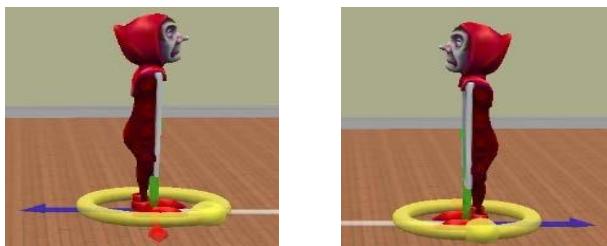
Perintah Instruksi	Tindakan Storyboard	Instruksi Pemograman
Lakukan sesuai perintah	Kelinci masuk	Bunny bergerak maju sejauh 2 meter
	Alice berbalik untuk melihat kelinci	Alice berbalik untuk menghadapi kelinci
	Kelinci berkata, "Apakah ini pestanya!"	Kelinci berkata, "Apakah ini pestanya!"
	Alice berkata, "Oh, tidak!"	Alice berkata, "Oh, tidak!"
	Kelinci belok ke kiri	Kelinci belok kiri 0,25
Lakukan bersama	Kelinci berjalan pergi dengan cepat Alice menggelengkan kepalanya.	Kelinci bergerak maju 4 meter dalam 1 detik Alice memutar kepalanya ke kiri dan ke kanan lagi.

#### 4.5 Gerakan objek

1. Gerakan objek bersifat egosentris, yaitu objek bergerak berdasarkan arah yang mereka hadapi.
2. Objek dapat bergerak dalam enam arah yaitu naik, turun, ke depan, ke belakang, kanan dan kiri.

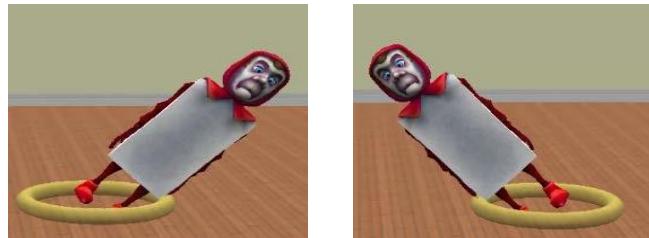
Tabel 4.2 Gerakan Objek

Prosedur	Deskripsi
Berbelok	Putar objek ke kiri, kanan, maju, atau mundur pada titik tengahnya. Belok kiri dan kanan pada sumbu vertikal objek; maju dan mundur pada sumbu horizontal objek.
Bergulung	Menggulung objek ke kiri atau kanan pada titik tengahnya menggunakan sumbu horizontal objek.



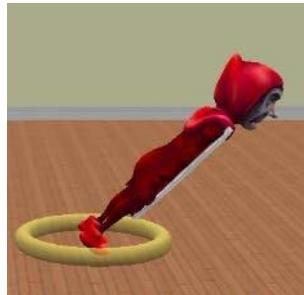
Gambar 4.2 Objek Berbelok dan Menggulung

3. Objek bergulir ke kiri dan kanan pada titik tengahnya menggunakan sumbu horizontal.



Gambar 4.3 Objek Bergulir

4. Objek berputar maju pada titik pusatnya menggunakan sumbu horizontal



Gambar 4.4 Objek Berputar Maju

5. Objek berputar mundur pada titik pusatnya menggunakan sumbu horizontal



Gambar 4.5 Objek Berputar Mundur

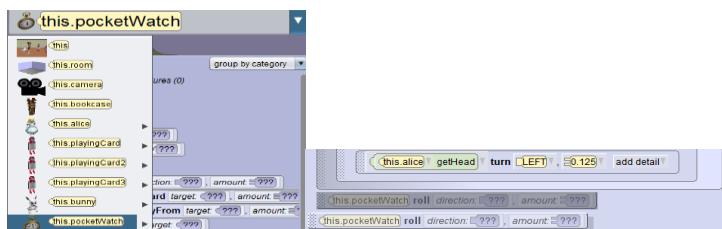
#### 4.6. Contoh Rotasi Sub-Bagian

1. Beberapa objek memiliki sub-bagian yang dapat dipindahkan untuk berbelok dan berputar.
2. Arloji saku memiliki jam dan menit yang dapat bergerak dan dapat berputar di titik tengah jam.



Gambar 4.6 Rotasi Sub-bagian

3. Kunci keberhasilan rotasi adalah **mengetahui titik pusat suatu objek**
4. Langkah-langkah memprogram objek untuk memutar adalah sebagai berikut:
  - a. Pilih objek yang akan diputar
  - b. Seret prosedur belokan atau berputar ke *code editor*
  - c. Tetapkan argumen arah
  - d. Tetapkan jumlah argumen (1.0 = satu putaran atau putaran penuh).



Gambar 4.7 Memutar Objek

Catatan: Garis indikator posisi hijau akan muncul untuk membantu pengguna menyelaraskan pernyataan pemrograman pada posisi yang diinginkan.

#### 4.7 Pernyataan kontrol

1. Pernyataan kontrol menentukan bagaimana pernyataan pemrograman dieksekusi dalam program

- myFirstMethod* dibuat dengan pernyataan kontrol urutan dalam perintah *Do in*. Di dalamnya, semua pernyataan pemrograman dieksekusi secara berurutan secara *default*.

Tabel 4.3 Pernyataan Kontrol

Prosedur	Deskripsi
Lakukan sesuai perintah	Jalankan pernyataan yang terkandung dalam pernyataan kontrol secara berurutan
Lakukan bersama	Jalankan pernyataan yang terkandung dalam pernyataan kontrol secara bersamaan
Hitung	Jalankan pernyataan yang terkandung dalam pernyataan kontrol beberapa kali
Sementara	Jalankan pernyataan yang terkandung dalam pernyataan kontrol saat kondisi yang ditentukan benar

#### 4.8 Pernyataan kontrol di *code editor*

- Pernyataan kontrol terletak di bagian bawah *editor code*
- Ini mewakili blok kode di mana pernyataan individu dikelompokkan
- Seret *icon* kontrol ke *code editor*, lalu tambahkan pernyataan ke dalamnya.



Gambar 4.8 Statement Kontrol Code Editor

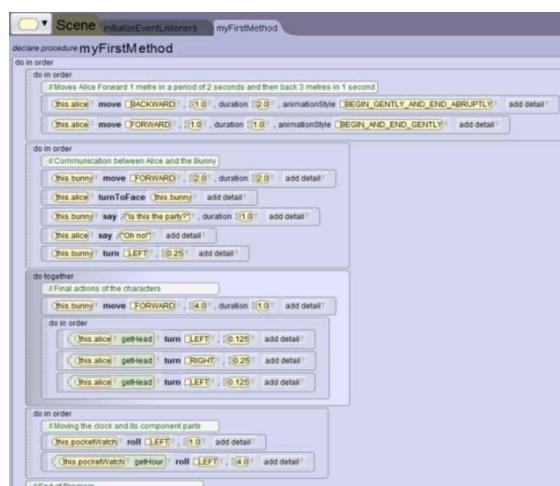
4. Untuk menambahkan prosedur ke pernyataan control dapat dilakukan dengan cara:
  - a. Seret pernyataan pemrograman baru atau yang ada ke dalam pernyataan kontrol
  - b. Dalam contoh yang diilustrasikan digambar 4.10, objek Alice akan bergerak maju, belok kiri, dan putar ke kanan secara berurutan.



Gambar 4.9 Pergerakan Objek

## 4.9 Membuat blok pemrograman

1. Meskipun *myFirstMethod* berisi *Do default* dalam urutan pernyataan kontrol, pengguna selalu dapat menambahkan *Do* dalam urutan ke area pemrograman.
2. Ini memungkinkan untuk menyimpan instruksi pemrograman bersama diblok yang terorganisir.
3. Blok dapat dengan mudah disalin ke *clipboard* atau dipindahkan dalam program besar.
4. *Nested Do* agar pernyataan kontrol tidak memengaruhi kinerja animasi, ini membuat pengeditan program lebih mudah bagi *programmer*.
  - a. Di sini, tiga pernyataan berurutan dan pernyataan yang dilakukan bersama bersarang di dalam pernyataan berurutan keempat.
  - b. *Nested Statement* menambahkan struktur visual ke program, seperti garis besar membawa struktur ke laporan.
  - c. *Nesting* adalah proses meletakkan satu benda ke dalam yang lain, seperti telur yang bersebelahan di dalam sarang.

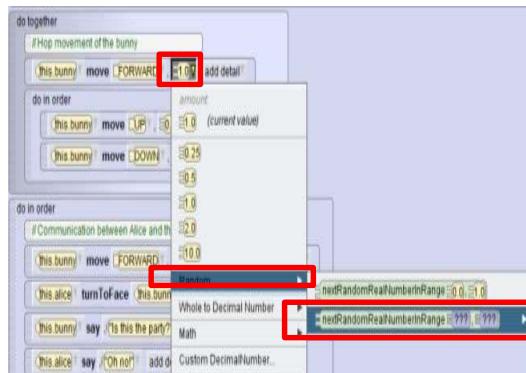


Gambar 4.10 Penggunaan *Nest Do*

#### **4.10 Angka acak**

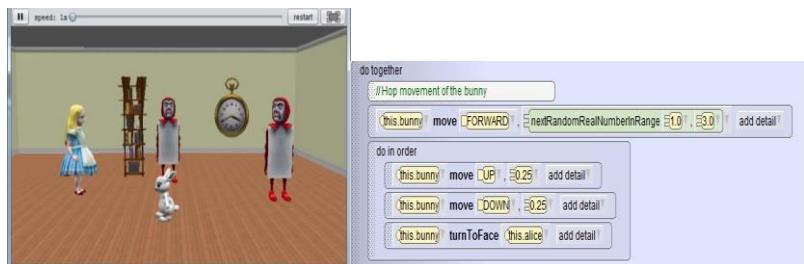
1. Angka acak adalah angka yang dihasilkan oleh komputer tanpa pola yang dapat diprediksi sesuai urutannya
2. Angka acak dihasilkan dalam rentang angka tertentu
3. Komputer mungkin memerlukan nomor acak untuk:
  - a. Keamanan: misalnya, kata sandi yang dibuat secara acak.
  - b. Simulasi: misalnya, pemodelan ilmu bumi (mis., Erosi seiring waktu).
4. Bilangan acak dapat dihasilkan dalam rentang angka yang luas, baik positif maupun negatif, serta dapat mencakup bilangan bulat dan bagian pecahan dari suatu bilangan.
5. Contoh angka acak:
  - a. 15674
  - b. -6934022
  - c. 0.371
  - d. -89.763
6. Angka acak dalam animasi, seperti:
  - a. Hewan tidak bergerak dalam garis geometris lurus
  - b. Ini sedikit mengubah arah saat berjalan, berenang, dan terbang.
  - c. Angka acak dapat digunakan dalam argumen jarak dari suatu metode sehingga objek bergerak dengan cara yang kurang dapat diprediksi, lebih seperti kehidupan.
  - d. Angka acak sering digunakan saat membuat *game* yang membutuhkan perilaku tak terduga
7. Langkah-langkah untuk menggunakan angka acak, yaitu:
  - a. Untuk argumen numerik apa pun, salah satu opsi adalah angka acak
  - b. Pilih argumen dari daftar *drop-down*
    - 1) Pilih *Random*
    - 2) Pilih *nextRandomRealNumberInRange*

- 3) Pilih *low and high values*
- Jalankan animasi
  - Alice 3 akan secara acak menghasilkan nilai dalam rentang yang dipilih ketika pernyataan dieksekusi



Gambar 4.11 Penggunaan Bilangan Acak

- Contoh animasi angka acak, yaitu kelinci bergerak maju secara acak naik dan turun sebelum berbalik ke arah Alice pada saat *run-time*.



Gambar 4.12 Contoh Animasi Angka Acak

## 4.11 Terminologi

Istilah-istilah yang digunakan dalam pelajaran ini yaitu:

- Control statements*

2. *Nesting*
3. *Random numbers*
4. *Textual storyboard*

#### **4.12 Ringkasan**

Dalam pelajaran ini, kita seharusnya belajar bagaimana:

1. Mengkorelasikan pernyataan *storyboard* dengan tugas pelaksanaan program
2. Menambahkan pernyataan kontrol ke *Code editor*
3. Menggunakan angka acak untuk mengacak gerakan

## **BAB V**

### **MENDEKLARASIKAN PROSEDUR**

Pelajaran ini mencakup tujuan-tujuan berikut:

1. Bandingkan serta tentukan animasi dan skenario
2. Tulis papan cerita
3. Bagan alur *storyboard*
4. Menjelaskan pewarisan dan bagaimana ciri-ciri diturunkan dari *superclasses* ke *subclass*
5. Menjelaskan kapan harus mengimplementasikan abstraksi prosedural
6. Menunjukkan cara mendeklarasikan prosedur
7. Identifikasi dan gunakan teknik abstraksi prosedural untuk menyederhanakan pengembangan animasi

#### **5.1 Gerakan objek**

1. Animator profesional memulai proses mereka dengan mengembangkan skenario — atau cerita — yang memberi animasi sebuah tujuan.
2. Contoh:
  - a. Sebuah kisah yang mewakili konflik dan pemecahan.
  - b. Pelajaran untuk mengajarkan konsep matematika.
  - c. Suatu proses untuk mensimulasikan atau menunjukkan.
  - d. Game untuk hiburan atau pelatihan.

#### **5.2 Skenario dan animasi**

Menentukan skenario dan animasi untuk mewakili skenario adalah langkah pertama untuk memprogram animasi.



Tabel 5.1 Skenario dan Animasi

Tipe Skenario	Skenario	Animasi
<b>Story</b>	Seekor kucing membutuhkan bantuan untuk turun dari pohon	Seorang petugas pemadam kebakaran manjat pohon untuk menyelamatkan kucing itu
<b>Lesson</b>	Menghafal simbol kimia sulit	Sebuah permainan mencocokkan simbol kimia dengan definisi mereka
<b>Process</b>	Mobil memiliki ban kempes	Demonstrasi menunjukkan cara mengganti ban pada mobil virtual
<b>Game</b>	Pesawat terbang harus menghindari benda di jalurnya saat terbang melintasi langit	Game interaktif manuver pesawat di sekitar benda-beda di langit

### 5.3 Storyboard

1. *Storyboard* mengidentifikasi spesifikasi desain untuk skenario animasi seperti: bagaimana benda muncul, bergerak, berbicara, berinteraksi, dan sebagainya.

2. Setelah skenario ditentukan, pengguna dapat mulai mengembangkan *storyboard* animasi.
3. Dua jenis *storyboard* digunakan untuk merencanakan animasi:
  - a. **Visual:** Serangkaian gambar ilustrasi yang mewakili adegan utama animasi.
  - b. **Tekstual:** Daftar tindakan yang terperinci dan terurut yang dilakukan setiap objek di setiap adegan animasi.

#### 5.4 Format papan cerita

*Storyboard* dibuat dalam format berikut:

1. **Digambar dengan kertas dan pensil**
2. **Dibuat menggunakan alat digital** seperti program pengolah kata, program cat atau menggambar, atau presentasi.
3. **Dibuat menggunakan komentar di editor Alice 3 Code**

#### Papan Cerita Visual

Papan cerita visual membantu pembaca memahami:

1. Komponen adegan
2. Bagaimana adegan awal akan diatur
3. Objek bergerak dan tidak bergerak dalam sebuah adegan
4. Tindakan yang akan terjadi
5. Interaksi pengguna yang akan terjadi selama eksekusi animasi



Gambar 5.1 Storyboard Visual

Laki-laki dan perempuan duduk di bangku taman. Laki-laki itu berjalan pergi, meninggalkan teleponnya.

Perempuan ini menyadari dan dia berfikir “Laki-laki itu melupakan teleponnya!”

Perempuan ini berteriak “hey! Kamu melupakan telepon mu!” laki-laki itu berbalik dan berjalan ke bangku, dia berkata “Terima kasih”

### Papan Cerita Tekstual

1. Papan cerita tekstual membantu pembaca memahami tindakan yang akan terjadi selama animasi.
2. Objek bergerak dan tidak bergerak dapat dengan mudah diidentifikasi dalam pernyataan tindakan, tetapi deskripsi yang lebih rinci mungkin diperlukan jika *programmer* tambahan juga terlibat dalam menerapkan adegan apa pun. Algoritma adalah daftar tindakan untuk melakukan tugas atau menyelesaikan masalah. Dalam komputasi, *storyboard* teks merupakan algoritma.

### Contoh Papan Cerita Tekstual 1

Tindakan program berikut secara berurutan:

1. Laki-laki dan perempuan duduk di bangku taman

2. Laki-laki berdiri dan berjalan pergi, meninggalkan ponselnya di bangku taman.
3. Perempuan menoleh untuk melihat telepon
4. Perempuan itu berpikir, "Hei! Laki-laki itu lupa teleponnya!"
5. Perempuan berkata dengan keras, "Hei! Kamu lupa teleponmu!"
6. Laki-laki berhenti dan berbalik
7. Laki-laki berjalan kembali ke bangku taman dan berkata, "Oh! Terima kasih!"

#### Contoh Papan Cerita Tekstual 2

1. Contoh ini menunjukkan bagaimana *programmer* dapat mengembangkan *storyboard* dengan terlebih dahulu menulis komentar di *Code editor* program.
2. Setelah itu, *programmer* dapat mulai mengembangkan animasi langsung dari *storyboard*.



Gambar 5.2 *Storyboard* Tekstual

Tabel 5.2 Komponen Papan Cerita Tekstual

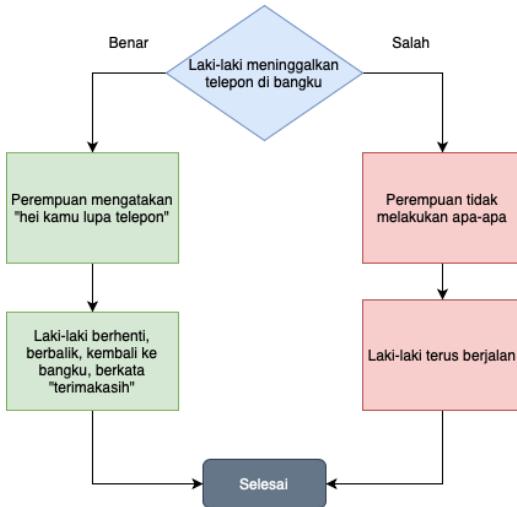
Komponen	Definisi	Contoh
----------	----------	--------



<i>Scene</i>	Tempat (atau “world” dalam Alice 3) tempat kisah bercerita	Taman, perpustakaan, sekolah, rumah, dan lain-lain.
<i>Objects</i>	Karakter bergerak atau tidak bergerak yang diprogram untuk bergerak dan bertindak	Hewan, mobil, manusia, pohon, dan lain-lain.
<i>Actions</i>	Instruksi untuk bagaimana setiap objek harus bertindak dalam adegan.	Berjalan 2 meter, belok kiri, katakan "Halo!"
<i>User Interactions</i>	Cara di mana pengguna melihat animasi dapat memanipulasi objek dalam animasi	Perintah <i>keyboard</i> atau klik <i>mouse</i> untuk membuat objek bergerak
<i>Design Specifications</i>	Bagaimana benda dan pemandangan harus terlihat dalam animasi	Ukuran, posisi, lokasi, warna, dan lain-lain.

### Flowcharting Storyboard

*Flowcharting storyboard* membantu *programmer* mengatur alur tindakan dan kondisi animasi.



## 5.5 Menggunakan *Storyboard* untuk mengatur program

1. *Storyboard* tekstual dapat digunakan untuk menghasilkan pernyataan komentar program dan mengatur pengembangan program
2. *Storyboard* juga dapat membantu *programmer* mengidentifikasi tindakan berulang untuk satu objek dan tindakan identik yang dapat dilakukan oleh banyak objek

## 5.6 Warisan

Warian berarti bahwa setiap objek *subclass* mewarisi metode dan properti *superclass*-nya. Ketika objek Dalmatian dibuat, ia mewarisi prosedur, fungsi, dan properti dari kelas berkaki empat dan *subkelas* Dalmatian yang dapat dilihat di *editor code*.



Gambar 5.3 Warisan

### 5.7 Warisan kelas

1. Sama seperti hewan di dunia nyata, objek di dunia pemrograman mewarisi karakteristik kelas mereka, termasuk semua metode kelas (prosedur dan fungsi).
2. Sebagai contoh, semua objek dalam kelas binatang berkaki empat di Alice mewarisi karakteristik empat kaki, kepala, tubuh, dan lain-lain.
3. Di dalam *superclass* berkaki empat ini, ada *subclass* untuk anjing, kucing, serigala, singa, sapi, dan lain-lain.
4. Setiap *subclass* menambahkan karakteristik yang lebih spesifik mengidentifikasi objek di dalamnya

### 5.8 Karakteristik yang diwarisi

1. Contoh pada Anjing Dalmatian
2. Karakteristik kelas Anjing (kelas induk atau "kelas super") meliputi empat kaki, dua mata, bulu, dan kemampuan menggonggong.

3. Karakteristik kelas ras Dalmatian (kelas anak atau "subclass," yang merupakan subset dari kelas anjing) meliputi bulu putih, bintik hitam, dan karakteristik lainnya.



Kelas Anjing (*Superclass*)

Kelas Dalmatian

Gambar 5.4 Pewarisan Pada Anjing Dalmatian

### 5.9 Membuat metode warisan

1. Selain metode yang telah ditentukan, *programmer* dapat membuat metode sendiri dan menampilkannya atau tersedia untuk objek *subclass* apa pun.
2. Metode yang diwariskan akan selalu ditampilkan di bagian atas daftar metode yang telah ditentukan setelah dibuat.

### 5.10 Mengidentifikasi perilaku berulang di *storyboard*

1. Memahami bahwa *subkelas milik superclasses* membantu seorang *programmer* mengidentifikasi perilaku berulang di *storyboard*.
2. Contoh:
  - a. Jika berencana untuk memprogram seekor anjing, kucing, dan singa berjalan, *programmer* harus memprogram perilaku berjalan berulang di *superclass* atau *di kelas berkaki empat*.
  - b. Hasilnya adalah bahwa semua *subclass* (anjing, kucing, singa) dapat menggunakan karakteristik berjalan yang diwariskan dan *programmer* tidak harus memprogram perilaku berjalan berulang untuk setiap objek secara individual.

## 5.11 Tab myFirstMethod

Tab *myFirstMethod* ditampilkan secara *default* ketika *code editor* dibuka



Gambar 5.5 Tab *myFirstMethod*

## 5.12 Kelas hierarki

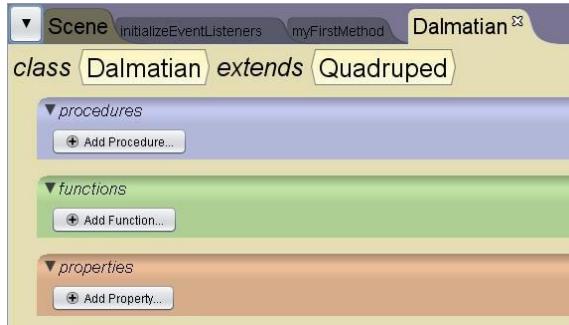
Klik menu *drop-down* kelas hierarki di sebelah kiri tab *myFirstMethod* (ditunjukkan oleh panah pengarah) untuk melihat daftar kelas dan subkelas dalam animasi Alice.



Gambar 5.6 Kelas Hirarki

## 5.13 Lihat metode kelas

Pilih *superclass* atau *subclass* untuk melihat prosedur, fungsi, dan properti yang ditentukan untuk kelas yang dipilih.



Gambar 5.7 Metode Kelas

#### 5.14 Abstraksi prosedural

1. Abstraksi prosedural adalah proses melihat kode pemrograman, mengidentifikasi pernyataan pemrograman berulang, dan mengekstraksinya ke dalam metode mereka sendiri, sehingga membuat kode lebih mudah untuk dipahami dan digunakan kembali.
2. Tinjau kode yang ada atau papan cerita tekstual untuk mengidentifikasi dan merencanakan metode yang perlu dinyatakan dalam program.
3. Abstraksi prosedural mungkin perlu diimplementasikan jika suatu objek di Alice perlu melakukan suatu tindakan, tetapi tidak ada prosedur yang diwarisi yang menyelesaikan tindakan itu.
4. Identifikasi perilaku berulang dan buat satu metode untuk:
  - a. Menyederhanakan kode, membuatnya lebih mudah dibaca.
  - b. Mengizinkan banyak objek kelas menggunakan metode yang sama
  - c. Mengizinkan *subclass* dari *superclass* menggunakan metode ini.
5. Contoh abstraksi prosedural adalah penghapusan pernyataan pemrograman berulang atau panjang dari *myFirstMethod* dan menempatkannya ke dalam metode sendiri untuk membuat kode lebih mudah dibaca, dipahami, dan digunakan kembali oleh banyak objek dan beberapa kelas.
6. Contoh lain abstraksi prosedural:

- a. Satu atau lebih objek dapat melakukan gerakan berulang yang sama
- b. Animasi satu ikan berenang memerlukan penulisan banyak prosedur berulang yang kemudian harus diulang untuk setiap ikan
- c. Mengambil banyak ruang di dalam *myFirstMethod*



Gambar 5.8 Abstraksi Prosedural Pada Ikan

- d. Terkadang, prosedur yang diperlukan untuk melakukan suatu tindakan tidak tersedia secara *default*. Misalnya, burung perlu terbang tetapi prosedur terbang tidak tersedia untuk objek burung.



Gambar 5.9 Abstraksi Prosedural Pada Burung

7. Ketika abstraksi prosedural terjadi:
  - a. Abstraksi prosedural dapat terjadi sebelum atau setelah pernyataan pemrograman dibuat

- b. Namun, dengan mengembangkan *storyboard* terlebih dahulu, *programmer* dapat lebih mudah mengidentifikasi prosedur yang akan diperlukan sebelum pemrograman dimulai.

### 5.15 Menyatakan prosedur

1. Gerakan tidak memiliki prosedur standar, seperti burung terbang.
2. Gerakan perlu digunakan oleh banyak objek atau kelas, seperti semua hewan berkaki empat melompat-lompat.
3. Gerakan tunggal membutuhkan banyak pernyataan pemrograman, seperti orang yang menggerakkan bagian tubuh untuk berjalan.
4. Contoh Prosedur
  - a. Burung itu terbang, memutar bahunya dan bergerak maju secara bersamaan.
  - b. Gerakan berulang ini dapat diekstraksi ke dalam prosedur terbangnya sendiri.

*Lakukan bersama:*

*Burung belok kanan bahu ke belakang*

*Burung belok kiri bahu ke belakang*

*Burung bergerak maju  
bersama*



*Lakukan*

*Lakukan bersama:  
terbang*

*Burung*

*Burung belok kanan ke depan  
bergerak maju*

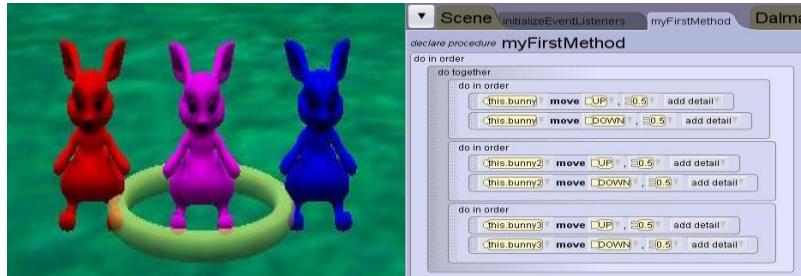
*Burung*

*Burung belok kiri ke depan*

*Burung bergerak maju*

## 5. Contoh lain prosedur

- a. Setiap kelinci bergerak ke atas dan ke bawah untuk mensimulasikan gerakan melompat
- b. Gerakan berulang yang digunakan oleh semua objek kelinci ini dapat diekstraksi ke dalam prosedur melompatnya sendiri



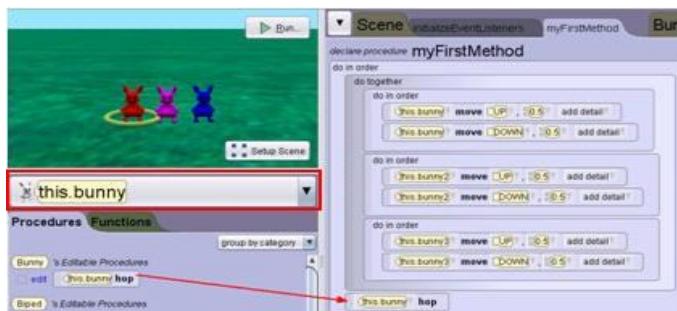
Gambar 5.10 Prosedur Pada Kelinci

6. Langkah-langkah untuk mendeklarasikan prosedur, yaitu:
  - a. Dari kelas hierarki, pilih kelas yang harus mewarisi prosedur.
  - b. Semua *subclass* kemudian akan mewarisi prosedur juga. (ada keterangan di bawah *superclass* mereka)
  - c. Klik tombol Tambahkan Prosedur ....



Gambar 5.11 Mendeklarasikan Prosedur

- d. Tulis kode untuk prosedur ini
  - e. Tambahkan prosedur baru ke *myFirstMethod* segera sehingga ketika ingin menguji animasi di *myFirstMethod*, programmer juga menguji animasi di prosedur baru.
7. Langkah-langkah untuk menambahkan prosedur ke tab *myFirstMethod* sebelum pemrograman, yaitu:
- a. Klik tab *myFirstMethod*
  - b. Pilih *instance* yang di mana programmer mengkoding prosedur dari menu *instance*
  - c. Di tab Prosedur, cari prosedur yang dinyatakan dan seret ke *myFirstMethod*.



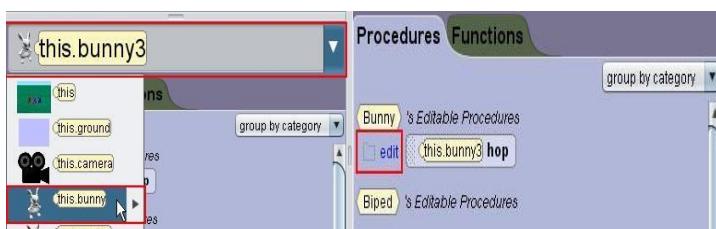
Gambar 5.12 Prosedur

- d. Kembali ke prosedur dengan mengklik *tab* di bagian atas *editor code* dengan nama prosedur



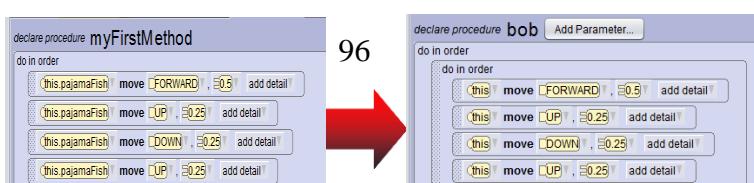
Gambar 5.13 Nama Prosedur

8. Langkah-langkah untuk mengakses dan mengedit prosedur yang dideklarasikan, yaitu:
- Di menu *instance*, pilih *instance* tempat prosedur dideklarasikan.
  - Klik *Edit* di sebelah kiri nama prosedur



Gambar 5.14 Mengakses dan Mengedit Prosedur yang Dideklarasikan

- Setelah selesai mengedit prosedur, klik tab *myFirstMethod* untuk kembali.
9. Mengidentifikasi peluang untuk prosedur yang dideklarasikan adalah sebagai berikut:
- Saat memprogram, gunakan teknik abstraksi prosedural.
  - Sebagai contoh, di *myFirstMethod*, ikan tersentak (bob) naik dan turun berulang kali.
  - Prosedur "bob" yang terpisah harus dinyatakan sebagai hasilnya.



Gambar 5.15 Pemisahan Prosedur

### 5.16 Abstraksi prosedural dan menggunakan *clipboard*

1. Setelah menulis banyak instruksi pemrograman di tab *myFirstMethod*, *programmer* dapat menentukan bahwa kode akan menjalankan program yang lebih baik jika berada dalam prosedur yang dinyatakan.
2. Untuk menghemat waktu, *programmer* dapat menyeret instruksi pemrograman ke icon *clipboard*.
3. Setelah membuat prosedur yang dideklarasikan, *programmer* dapat menyeret instruksi pemrograman dari *clipboard* ke dalam prosedur yang dinyatakan.

### 5.17 Menggunakan prosedur warisan

1. Prosedur yang dideklarasikan pada level *superclass* tersedia untuk objek lain di kelas itu
2. Misalnya, prosedur "bipedWave" yang dibuat untuk membuat *playingcard* melambaikan tangannya di udara dan dapat digunakan untuk membuat Alice melambaikan tangannya juga di udara.



Gambar 5.16 Prosedur Warisan

### 5.18 Deklarasikan prosedur di tingkat *superclass*

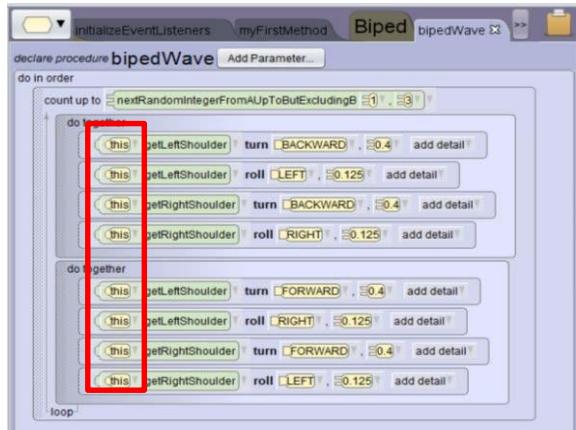
1. Deklarasikan prosedur "bipedWave" pada tingkat *superclass* *biped* sehingga *Playingcard* dan Alice bisa menggunakan prosedur.
2. Setiap *biped* lain yang mungkin ditambahkan kemudian juga akan memiliki akses ke prosedur ini



Gambar 5.17 Prosedur Tingkat *Superclass*

### 5.19 Pengidentifikasi objek "this"

1. Ketika prosedur yang dideklarasikan dibuat, pengidentifikasi objek "this" digunakan untuk menunjukkan bahwa *instance* yang memanggil prosedur adalah ini.



Gambar 5.18 Identifikasi Objek “this”

2. Contoh pengidentifikasi objek “this” adalah:
  - a. Jika Alice dipilih dalam menu contoh, maka
  - b. Prosedur *bipedWave* ditambahkan ke *myFirstMethod*; ini dalam prosedur yang dinyatakan mengacu pada Alice.



Gambar 5.19 Prosedur *bipedWave*

- c. Jika *playingCard* dipilih dalam menu *instance*, maka prosedur *bipedWave* ditambahkan ke *myFirstMethod*. Ini dalam prosedur yang dideklarasikan sekarang merujuk ke *playingCard*.
- d. Intinya, ini selalu merujuk pada *instance* kelas yang memanggil prosedur.

## 5.20 Terminologi

Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:

1. Algoritma
2. Prosedur yang diumumkan
3. Warisan
4. Abstraksi prosedural
5. Skenario
6. *Storyboard*

### **5.21 Ringkasan**

Dalam pelajaran ini, kita seharusnya belajar bagaimana:

1. Membandingkan serta menentukan animasi dan skenario
2. Menulis papan cerita
3. Bagan alur *Storyboard*
4. Menjelaskan pewarisan dan bagaimana ciri-ciri diturunkan dari *superclasses* ke *subclass*
5. Menjelaskan kapan harus mengimplementasikan abstraksi prosedural
6. Menunjukkan cara mendeklarasikan prosedur
7. Identifikasi dan menggunakan teknik abstraksi prosedural untuk menyederhanakan pengembangan animasi

## BAB VI

### CONTROL STATEMENTS

Pelajaran ini meliputi tujuan-tujuan sebagai berikut:

1. Menentukan beberapa pernyataan kontrol untuk mengontrol waktu animasi
2. Membuat animasi yang menggunakan pernyataan kontrol untuk mengontrol waktu animasi
3. Mengenali konstruksi pemrograman untuk menjalankan gerakan simultan

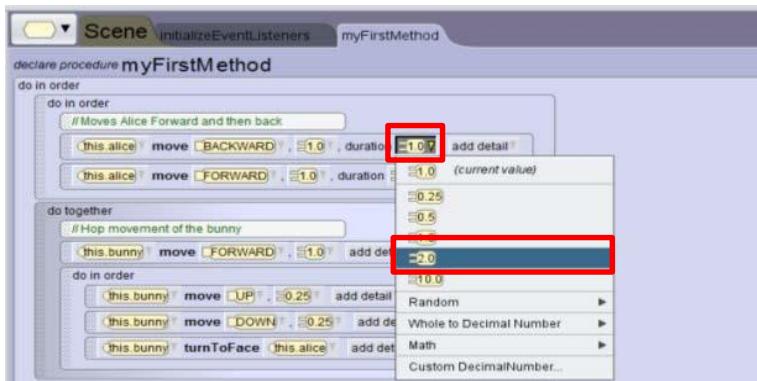
#### 6.1 Argumen

1. Program komputer **membutuhkan argumen** untuk memberi tahu cara **menerapkan prosedur**
2. Argumen prosedur dapat **di edit** atau didefinisikan lebih lanjut untuk mengontrol pergerakan dan waktu objek
3. Contoh **argumen Alice 3** meliputi:
  - a. Objek
  - b. Arah
  - c. Jumlah Arah
  - d. Durasi Waktu
4. Contoh argumen dalam suatu prosedur dapat dilihat pada gambar 6.1



Gambar 6.1 Argumen Dalam Prosedur

5. Langkah-langkah untuk mengubah argumen adalah sebagai berikut:
  - a. Di sebelah *argument's value*, klik panah untuk menampilkan menu nilai.
  - b. Pilih *new value*
  - c. Menu menunjukkan nilai saat ini, diikuti oleh nilai-nilai pra *set* untuk memilih dan diikuti oleh pilihan menu tambahan untuk menentukan pengacakan, perhitungan matematika, atau angka *decimal custom*.



Gambar 6.2 Mengubah Argumen

## 6.2 Argumen *placeholder*

1. Saat menambahkan prosedur ke kode, *programmer* harus memilih nilai untuk setiap argumen dalam prosedur.
2. Sering kali *programmer* dapat memilih nilai yang telah ditetapkan sebagai pengganti (nilai sementara) yang kemudian diubah selama siklus *edit*
3. Menggunakan nilai *placeholder* adalah pendekatan umum untuk membuat dan memperbaiki kinerja animasi

### 6.3 Pergerakan serentak

Untuk membuat gerakan simultan suatu objek, gunakan pernyataan kontrol *do together*.

Tabel 6.1 pernyataan kontrol *do together*

Pernyataan Kontrol	Deskripsi
<b><i>Do In Order</i></b>	<ul style="list-style-type: none"><li>• Pernyataan kontrol default di <i>code editor</i></li><li>• Menjalankan prosedur secara berurutan</li></ul>
<b><i>Do Together</i></b>	<ul style="list-style-type: none"><li>• Menjalankan prosedur secara bersamaan</li><li>• Digunakan untuk gerakan simultan seperti gerakan berjalan dan duduk</li></ul>

1. Contoh gerakan simultan adalah suatu gerakan yang dieksekusi bersama bisa sesederhana secara bersamaan mengangkat kedua lengan dari objek *biped* dari posisi gantung ke posisi lengan lurus keatas.



Gambar 6.3 Gerakan Simultan

2. Contoh gerakan simultan lainnya adalah gerakan berjalan yang membutuhkan gerakan simultan pinggul dan bahu. Untuk membuat gerakan berjalan dalam *biped*, gunakan:

- Serangkaian prosedur gerak, gulung, dan belok.
- Lakukan bersama mengontrol pernyataan

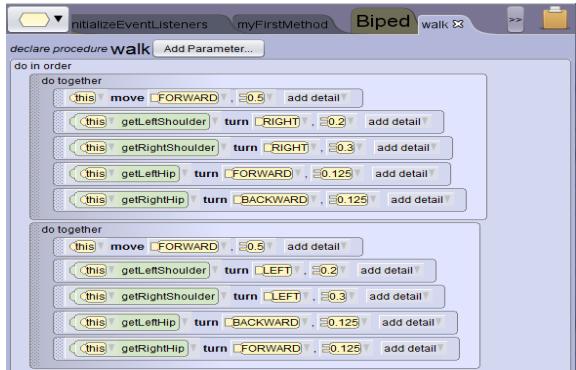
Pemrograman yang berbeda mungkin diperlukan untuk objek yang berbeda karena tidak ada dua objek yang berjalan dengan cara yang sama.

3. Contoh *textboard story* berjalan dapat dilihat pada tabel 6.2

Tabel 6.2 *Textboard Story* Berjalan

Perintah Instruksi	Instruksi Pemrograman
<i>Do Together</i>	Seluruh tubuh bergerak maju
	Bahu kiri berbelok ke kanan
	Bahu kanan belok kanan
	Pinggul kiri berputar ke depan
	Pinggul kanan berbelok ke belakang
<i>Do Together</i>	Seluruh tubuh bergerak maju
	Bahu kiri belok kiri
	Bahu kanan belok kiri
	Pinggul kiri berbelok ke belakang
	Pinggul kanan berputar ke depan

4. Contoh gerakan berjalan sederhana dapat dilihat pada kode yang tertara digambar 6.4



Gambar 6.4 Code Gerakan Berjalan

#### 6.4. Prosedur mengimbangi satu sama lain

1. Kesalahan umum adalah memasukkan dua prosedur yang membantalkan satu sama lain dalam konstruksi *Do Together*
2. Misalnya, jika *programmer* menyertakan prosedur naik 1 meter, diikuti dengan prosedur turun 1 meter dalam *Do Together*, tidak ada yang akan terjadi.
3. Maka, prosedur membantalkan satu sama lain.



Gambar 6.5 Prosedur Mengimbangi Satu Sama Lain

#### 6.5 Prosedur *setVehicle*

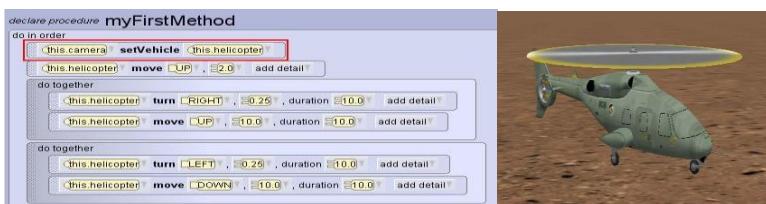
1. Prosedur *setVehicle* menggunakan konsep objek pengendara dan objek kendaraan

2. Objek pengendara dipilih ketika prosedur setVehicle digunakan untuk menentukan kendaraan pengendara
3. Kemudian, ketika objek kendaraan diprogram untuk bergerak, objek pengendara akan secara otomatis bergerak dengannya.
4. Contoh:
  - a. Seseorang mengendarai unta atau kuda
  - b. Kamera mengikuti helikopter untuk memotret pemandangan dari sudut pandang helikopter
5. Penggambaran prosedur *setVehicle* pertama:
  - a. Anak itu diposisikan di atas unta di *Scene editor*
  - b. Kemudian, unta diatur sebagai kendaraan anak dalam *Code editor*.
  - c. Ketika unta bergerak, anak tetap di atas dan bergerak dengan unta.



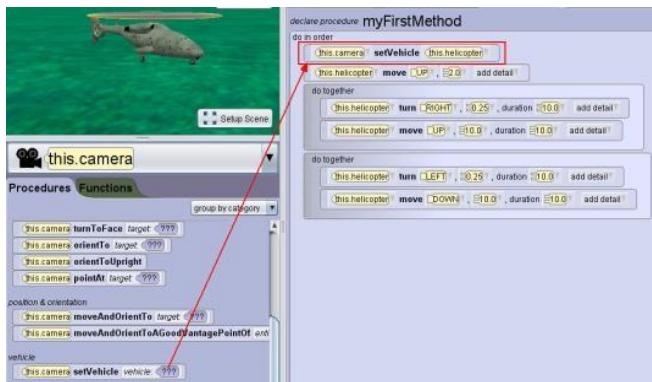
Gambar 6.6 Prosedur *setVehicle* 1

6. Penggambaran prosedur *setVehicle* kedua:
  - a. Helikopter ditetapkan sebagai kendaraan kamera dalam *Code editor*
  - b. Ketika helikopter bergerak, kamera memfilmkan pemandangan dari sudut pandang helikopter.



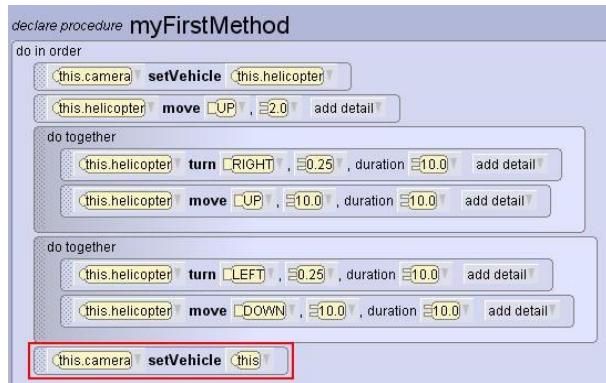
Gambar 6.7 Prosedur *setVehicle* 2

7. Langkah-langkah menggunakan prosedur *setVehicle*, yaitu:
  - a. Tentukan objek kendaraan dan objek pengendara
  - b. Di *Code editor*, pilih objek pengendara dari menu *Instance*.
  - c. Dari *Procedures tab*, seret prosedur *setVehicle* ke *Code editor*.
  - d. Dalam *procedure*, pilih objek kendaraan dari menu.



Gambar 6.8 Menggunakan Prosedur *setVehicle*

8. Langkah-langkah untuk menghentikan prosedur *setVehicle*, yaitu:
  - a. Jika *programmer* ingin objek pengendara turun dari objek kendaraan, seret prosedur *setVehicle* lain ke *Code editor* pada titik pengendara harus turun dari kendaraan.
  - b. Atur kendaraan untuk ini, yang mengatur kendaraan pengendara kembali ke tempat kejadian.



Gambar 6.9 Menghentikan Prosedur *setVehicle*

## 6.6 Terminologi

Istilah-Istilah yang digunakan dalam pelajaran ini, yaitu:

1. *Arguments*
2. *Do together control statement*
3. *Do in order control statement*

## 6.7 Ringkasan

Dalam pelajaran ini, kita seharusnya belajar bagaimana:

1. Menentukan beberapa pernyataan kontrol untuk mengontrol waktu animasi
2. Membuat animasi yang menggunakan pernyataan kontrol untuk mengontrol waktu animasi
3. Mengenali konstruksi pemrograman untuk menjalankan gerakan simultan

## BAB VII

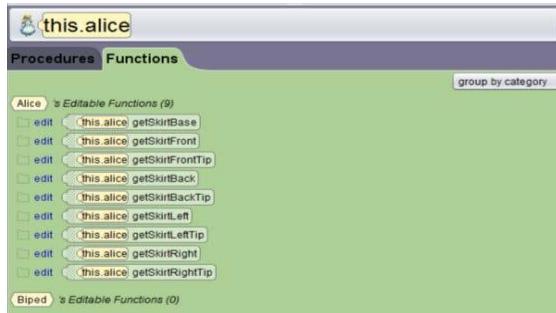
## FUNCTIONS

Pelajaran ini mencakup tujuan-tujuan berikut:

Menggunakan Fungsi untuk Mengontrol Pergerakan Berdasarkan Nilai Kembali

### 7.1 Fungsi

1. Digunakan untuk mengajukan pertanyaan tentang properti suatu objek
2. Mirip dengan prosedur kecuali bahwa fungsi mengembalikan nilai dari jenis tertentu
3. Dapat digunakan untuk menghitung nilai
4. Fungsi menjawab pertanyaan tentang suatu objek, seperti tinggi, lebar, kedalaman, dan bahkan jaraknya ke objek lain.
5. Fungsi menjawab pertanyaan dengan tepat
6. Fungsi menyediakan jawaban yang tepat untuk pertanyaan, seperti:
  - a. Berapa jarak antara Dalmatian dan kelinci?
  - b. Berapa tinggi kartu bermain?
  - c. Berapa lebar *pocketWatch*?
7. Fungsi *boolean* mengembalikan nilai benar atau salah
8. Sebagai contoh, jika fungsi *isFacing* dipanggil untuk menentukan apakah objek Alice menghadap objek kelinci, nilai benar atau salah akan dikembalikan.



Gambar 7.1 Functions

## 7.2 Tab Fungsi di panel metode

1. Pilih objek dari menu *Instance* dan kemudian lihat fungsinya
2. Di sinilah *programmer* dapat menemukan informasi tentang sambungan yang khusus untuk jenis objek yang telah ditambahkan ke animasi.

## 7.3 Fungsi Memecahkan Masalah Jarak

1. Misalkan *programmer* ingin memindahkan Dalmatian langsung ke kelinci tanpa harus secara manual menentukan nilai dengan menebak jarak antara Dalmatian dan kelinci.



Gambar 7.2 Memindahkan Objek

2. *Programmer* bisa menebak jarak dengan menentukan nilai *placeholder* dan menguji pergerakan sampai mendekati hasil akhir yang diinginkan, tetapi cara yang lebih efisien adalah menggunakan fungsi dalam menentukan jarak yang tepat untuk bergerak.



Gambar 7.3 Menentukan Jarak

3. Gunakan fungsi *getDistanceTo* sebagai bagian dari prosedur pemindahan untuk menyelesaikan masalah jarak ini. Langkah-langkah untuk menggunakan fungsi *getDistanceTo* adalah:
  - a. Tentukan objek bergerak dan objek target
  - b. Pada *editor Code*, pilih objek bergerak dari menu *Instance*.
  - c. Seret prosedur pindah ke *editor Code*
  - d. Pilih arah dan argumen *placeholder* untuk jarak (argumen jarak akan dimodifikasi pada langkah berikutnya)



Gambar 7.4 Arah dan Placeholder Jarak

- e. Dari tab *Functions*, seret *getDistanceTo* ke nilai jarak yang disorot.



Gambar 7.5 *getDistanceTo* ke Nilai Jarak

f. Pilih objek target

## **BAB VIII**

### **IF DAN WHILE**

Pelajaran ini mencakupi beberapa tujuan-tujuan sebagai berikut:

1. Menjelaskan metode, kelas, dan contoh
2. Menjelaskan skenario di mana IF control structure akan digunakan
3. Menjelaskan skenario di mana WHILE control structure akan digunakan
4. Mengenali sintaks untuk suatu metode, kelas, fungsi, dan prosedur
5. Menjelaskan input dan output

#### **8.1 Alice 3**

Tabel 8.1 **Perbandingan Alice 3 dan Java**

<b>Alice 3</b>	<b>Java</b>
Lingkungan pemrograman 3D yang menggunakan representasi visual untuk bahasa pemrograman.	Bahasa pemrograman dengan sintaks yang bisa diedit menggunakan Integrated Development Environment (IDE).
Digunakan untuk membuat animasi atau interaktif game sambil bekerja dengan kontruksi pemrograman.	Digunakan untuk membuat aplikasi yang berjalan di platform apa pun, termasuk web, menggunakan sintaks Java.
Seret dan lepas antarmuka yang dirancang untuk mengurangi kesalahan sintaks dan membuatnya lebih mudah untuk mempelajari cara memprogram.	IDE membantu Anda dalam memodelkan benda-benda didunia nyata, yang memungkinkan untuk digunakan kembali juga perawatan yang lebih mudah.

## 8.2 Prosedur dalam Alice 3

- Dalam Alice 3, prosedur adalah bagian dari kode yang mengirimkan pesan ke objek yang meminta untuk melakukan suatu tindakan.
- Prosedur tidak mengembalikan nilai.
- Seperangkat prosedur tersedia untuk setiap kelas.

## 8.3 Metode panel di Alice 3

- Panel metode mencantumkan semua prosedur objek dan fungsi.
- Panel tersebut berisi semua prosedur bawaan dan yang ditentukan pengguna dan fungsi yang tersedia untuk setiap objek dalam animasi Anda.



Gambar 8.1 Metode Panel

## 8.4 Mendeklarasikan prosedur dalam Alice 3

Anda dapat mendeklarasikan (membuat) prosedur Anda sendiri di Alice 3.



Gambar 8.2 Pendeklarasian Prosedur

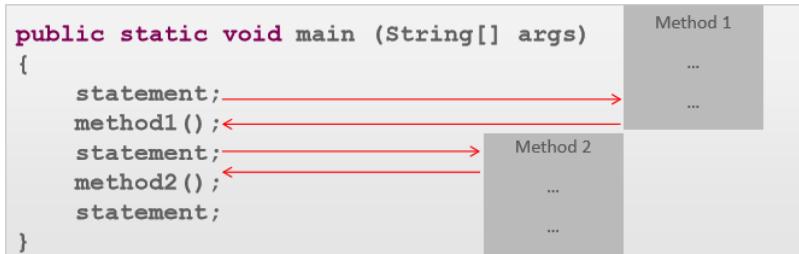
## 8.5 Metode di Java

Metode di Java sama dengan prosedur di Alice 3. Metode adalah bagian dari kode yang mengirim pesan ke objek yang memintanya untuk melakukan suatu tindakan. Metode:

- Tergolong ke kelas.
- Disebut berdasarkan nama.
- Dapat dipanggil kapan saja dalam suatu program menggunakan nama metode.
- Ketika nama metode ditemui dalam suatu program, itu dijalankan.

## 8.6 Metode dalam contoh Java

Ketika metode selesai, di eksekusi kembali ke area kode program dari mana ia dipanggil, dan program berlanjut ke baris kode berikutnya.



Gambar 8.3 Metode dalam Java

## 8.7 Keputusan untuk setiap metode

Ada tiga keputusan yang harus diambil untuk metode apa pun:

- a. **Apa yang harus dilakukan metode ini.**
- b. **Input apa yang dibutuhkan oleh metode ini.**
- c. **Apa jawaban yang diberikan metode ini.**



Sintaks Java untuk suatu metode:

```
[modifiers] dataType methodName(parameterList) {  
    <methodBody>  
    return result;  
}
```

Gambar 8.4 Sintaks Java

## 8.8 Properti metode

Tabel 8.2 Properti Metode

Properti Metode	Deskripsi
Modifiers	Ini adalah opsional dan dapat bersifat publik, pribadi, dilindungi, atau dibiarkan kosong.
datatype	Tipe data, seperti int.
methodName	Nama dari metode Anda
parameterList	nama parameter yang dipisahkan dengan komatipec datanya; setiap parameter terdaftar dengan tipe data pertama, lalu namanya.
methodBody	Set pernyataan yang melakukan tugas.
return	Kata kunci yang mengirim nilai hasil kembali kekode yang disebut metode.

## 8.9 Metode Findmax()

```
public class TestFindMax {  
    /** Main method */  
    public static void main(String[] args) {  
        int i = 5;  
        int j = 2;  
        int k = findMax(i, j);  
        System.out.println("The maximum between " + i + " and " + j +  
                           " is " + k);  
    }//end method main  
    /** Return the max between two numbers */  
    public static int findMax(int num1, int num2) {  
        int result;  
        if (num1 > num2)  
            result = num1;  
        else  
            result = num2;  
        //endif  
        return result;  
    }//end method findMax
```

findMax Method

Gambar 8.5 Metode FindMax

## 8.10 Kelas di Alice 3

Dalmatian adalah seekor anjing. Ketika objek Dalmatian ditambahkan ke sebuah adegan, ia memiliki sifat-sifat kelas Dalmatian seperti: empat kaki, dua telinga, kulit putih dan hitam berbintik, dan kemampuan berjalan.

*"Kelas adalah spesifikasi, seperti cetak biru atau pola, tentang cara membangun sebuah objek."*



Gambar 8.6 Kelas di Alice

## 8.11 Kelas di Java

Grup pertama:

- a. Opsional; mengacu pada visibilitas dari objek lain.
- b. sarana publik terlihat di mana-mana.
- c. Defaultnya adalah paket atau terlihat dalam paket saat ini saja.

Kelompok kedua:

- a. Opsional; mendefinisikan apakah kelas dapat diwarisi atau diperpanjang oleh kelas lain.
- b. Kelas abstrak harus diperluas dan kelas akhir tidak akan pernah diperpanjang oleh warisan.
- c. Default menunjukkan bahwa kelas mungkin atau mungkin tidak diperpanjang atas kebijakan programmer.

Class\_name adalah nama kelas.

Opsi ketiga dari ekstensi terkait dengan warisan.

Opsi implementasi keempat terkait dengan interface.

```
["public"] ["abstract"|"final"] "class" Class_name
    ["extends" object_name] ["implements" interface_name]
    "{"
        // properties declarations
        // behavior declarations
    }"
```

Gambar 8.7 Kelas di Java

## 8.12 Kode untuk membuat kelas cat (kucing) di Java

```
class Cat{  
    int catAge;  
    String catName;  
    public Cat(String name){  
        catName = name;  
    } //end of constructor  
  
    public void setAge(int age){  
        catAge = age;  
    } //end method setAge  
  
    public int getAge(){  
        return catAge;  
    } //end method getAge  
  
    public static void main(String []args){  
        Cat myCat = new Cat("Garfield");  
        myCat.setAge(6);  
        System.out.println("Cat age: " + myCat.getAge());  
    } //end method main  
} //end class Cat
```

Gambar 8.8 Source code Membuat Kelas Cat

## 8.13 Contoh di Alice 3

Saat Anda menambahkan objek ke sebuah adegan, ini menciptakan sebuah contoh kelas.

*"Sebuah objek adalah turunan dari kelas."*



Gambar 8.9 Hasil Source Code

## 8.14 Instances di Java

Dalam metode utama, turunan myCat dari kelas kucing yang dibuat.

```
class Cat{  
    int catAge;  
    String catName;  
    ...  
    ...  
    ...  
    public static void main(String []args){  
        Cat myCat = new Cat("Garfield");  
        myCat.setAge(6);  
        System.out.println("Cat age: " + myCat.getAge());  
    } //end method main  
} //end class Cat
```

Gambar 8.10 Instances di Java

## 8.15 Membuat instance dari class

Sebuah instance dari kelas dibuat menggunakan operator baru diikuti oleh nama kelas.

```
class Cat{  
    int catAge;  
    String catName;  
    ...  
    ...  
    ...  
    public static void main(String []args){  
        Cat myCat = new Cat("Garfield"); Create new instance  
        myCat.setAge(6);  
        System.out.println("Cat age: " + myCat.getAge());  
    } //end method main  
} //end class Cat
```

Gambar 8. 11 Instance dari Java

## 8.16 Control structure

- Control Structure memungkinkan Anda mengubah urutan caranya pernyataan dalam program Anda dieksekusi.
- Alice dan Java memungkinkan untuk jenis dari Control Structure.

Tabel 8.3 Control Structure

Tipe	Deskripsi	Contoh
------	-----------	--------

Keputusan Struktur Kontrol	Memungkinkan Anda memilih bagian kode tertentu yang akan dieksekusi.	jika ... lalu ... Lain
Pengulangan Control Structure	Memungkinkan Anda menjalankan bagian kode tertentu dalam beberapa kali.	sementara lingkaran

## 8.17 IF control structure

IF Control Structure adalah pernyataan yang memungkinkan Anda memilih dan menjalankan blok kode tertentu saat dilewati bagian lain.

Struktur ini memiliki bentuk seperti berikut.

```
if (boolean_expression) {
    doSomething();
}
else {
    doSomethingElse();
}//endif
```

Gambar 8.12 Source code IF control Structure

Sebagai contoh:

- Jika seorang siswa menerima skor 90% atau lebih pada nilai tes mereka, maka beri mereka "A".
- Jika seorang siswa menerima skor yang lebih besar atau sama dengan hingga 80% dan kurang dari 90%, maka beri mereka "B".
- Jika seorang siswa menerima skor yang lebih besar atau sama dengan hingga 65% dan kurang dari 80%, maka beri mereka "C".
- Jika seorang siswa menerima kurang dari 65%, maka beri mereka "F".

```

public class Grade {
    public static void main( String[] args ){
        //variable declaration section
        double grade = 89.0;

        if( grade >= 90 ){
            System.out.println( "A" );
        }
        else if( (grade < 90) && (grade >= 80) ){
            System.out.println("B" );
        }
        else if( (grade < 80) && (grade >= 65)){
            System.out.println("C" );
        }
        else{
            System.out.println("F" );
        }
    } //endif
} //end method main
} //end class Grade

```

Gambar 8.13 Source code contoh

## 8.18 WHILE control structure

- WHILE Control Structure, atau while loop, adalah pernyataan Java atau blok pernyataan yang memungkinkan Anda untuk menjalankan blok kode tertentu berulang kali selama beberapa kondisi terpenuhi.
- Kondisi diuji sebelum setiap loop.
- while loop format:

```

while( boolean_expression ){
    statement1;
    statement2;
    .
}

```

Gambar 8.14 Source code While control Structure

### Contoh While Control Structure

```

class WhileDemo {
    public static void main(String[] args){
        //variable declaration section
        int count = 1;

        while (count < 11) {
            System.out.println("Count is: " + count);
            count++;
        } //end while
    } //end method main
} //end class WhileDemo

```

Gambar 8.15 Contoh While

## 8.19 Input dan output

- Program Java berfungsi pada banyak platform.
- Mereka dapat berupa program sederhana yang dijalankan dari baris perintah, atau mereka dapat memiliki interface pengguna grafis yang kompleks.
- Saat belajar pemrograman, Anda akan membuat program yang menggunakan baris perintah khusus untuk input dan outputnya.
- Dengan lebih banyak pengalaman, Anda dapat membangun interface pengguna grafis dan belajar tentang perpustakaan yang diperlukan untuk membangun mereka.

## 8.20 Kode input dan output

- Cetak ke contoh layar.

```
System.out.println ("Hello World!");
```

- System.out.println() adalah bagian dari kode yang tersedia secara otomatis dan ada disetiap program Java.

Contoh **input pengguna menggunakan package java.util.Scanner.**

```
import java.util.Scanner;
public class ReadString {
    public static void main (String[] args) {
        System.out.print("Enter your name: ");
        Scanner in = new Scanner(System.in);
        String userName = null;
        userName = in.nextLine();
        System.out.println("Thanks for the name, " + userName);
    }//end method main
}//end class ReadString
```

Gambar 8.16 Source code Input dan Output

## 8.21 Terminologi

Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:

- Class
- Control Structure
- IF Control Structure
- Mesin Virtual (Instance)
- Metode

f. WHILE Control Structure

## **8.22 Ringkasan**

Dalam pelajaran ini, Anda seharusnya belajar bagaimana:

- a. Menjelaskan metode, kelas, dan contoh
- b. Jelaskan skenario di mana IF Control Structure akan digunakan
- c. Menjelaskan skenario di mana WHILE Control Structure akan digunakan
- d. Mengenali sintaks untuk metode, kelas, fungsi, dan prosedur
- e. Menjelaskan input dan output

## BAB IX

# FUNGSI

Pelajaran ini mencakup tujuan berikut:

Menggunakan fungsi untuk mengontrol pergerakan berdasarkan nilai balik

### 9.1 Fungsi

- a. Digunakan untuk mengajukan pertanyaan tentang property suatu objek.
- b. Mirip dengan prosedur kecuali bahwa mereka mengembalikan nilai dari jenis tertentu.
- c. Dapat digunakan untuk menghitung nilai.

*“Fungsi menjawab pertanyaan tentang suatu objek, seperti tinggi, lebar, kedalaman, dan bahkan jaraknya ke objek lain.”*

### 9.2 Fungsi tepat menjawab pertanyaan

Fungsi memberikan jawaban yang tepat untuk pertanyaan, seperti:

- a. Berapa jarak antara Dalmatian dan kelinci?
- b. Berapa tinggi Kartu yang dimainkan?
- c. Berapa lebar Jam saku? boolean function returns either a true or false value.

Sebagai contoh, jika fungsi `is Facing` dipanggil untuk menentukan apakah objek Alice menghadap objek kelinci, nilai benar atau salah akan dikembalikan.

### 9.3 Tab fungsi

- a. Tab Functions berada di panel metode.
- b. Pilih objek dari menu Instance, dan kemudian lihat fungsinya.

Gambar 9.1 Tab Fungsi

Di sinilah Anda dapat menemukan informasi tentang sambungan yang khusus



untuk jenis objek yang telah Anda tambahkan ke animasi Anda.

#### 9.4 Fungsi memecahkan masalah jarak

- a. Misalkan kita ingin memindahkan Dalmatian langsung ke kelinci tanpa harus secara manual menentukan, melalui coba-coba, jarak antara Dalmatian dan Kelinci.
- b. Kita bisa menebak jarak dengan menentukan nilai placeholder dan menguji pergerakan sampai kita mendekati hasil akhir yang diinginkan, tetapi cara yang lebih efisien adalah menggunakan fungsi untuk menentukan jarak yang tepat untuk bergerak.



Gambar 9.2 Hasil Fungsi Jarak

### 9.5 Gunakan fungsi get distance to

Gunakan fungsi get Distance To sebagai bagian dari prosedur pemindahan untuk menyelesaikan macalah jarak ini



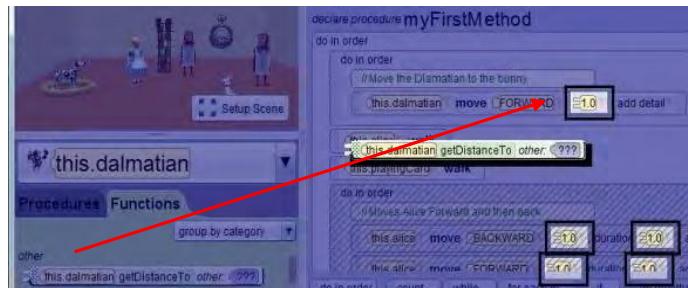
Gambar 9.3 Fungsi Get Distance To



Gambar 9.4 Hasil Get Distance To

### 9.6 Langkah-langkah untuk menggunakan fungsi get distance to

- Menentukan objek bergerak dan objek target.
- Pada editor Kode, pilih objek bergerak dari menu Instance.
- Seret prosedur pindah ke editor Kode.
- Pilih arah dan argumen placeholder untuk jarak (argumen jarak akan dimodifikasi pada langkah berikutnya).
- Dari tab Functions, seret ubin getDistanceTo ke nilai jarak yang disorot



Gambar 9.5 Langkah – Langkah Get Distance to (a)

f. Pilih objek target



Gambar 9.6 Langkah – Langkah Get Distance to (b)

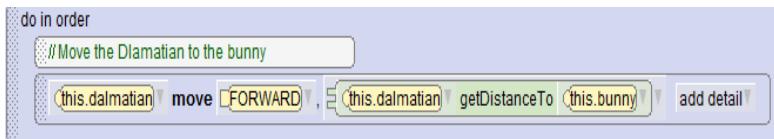


Gambar 9.7 Langkah – Langkah Get Distance to (c)

## 9.7 Uji fungsinya

- Dalam contoh di bawah ini, **Dalmatian bergerak ke pusat Kelinci** saat run-time.

- b. Ini bisa dibaca dengan keras sebagai "tentukan jarak dari pusat Dalmatian ke pusat Kelinci dan kemudian gerakkan Dalmatian ke depan jumlah itu."



Gambar 9.8 Uji Fungsi

- c. Klik tombol Run untuk menguji pernyataan pemrograman.
- d. Dalmatian bergerak ke tengah Kelinci.
- e. Ini karena **fungsi getDistanceTo menghitung jarak antara pusat dari kedua objek**.
- f. Fungsi menghitung jarak dari pusat Dalmatian ke pusat kelinci, dan memindahkan objek menggunakan nilai itu



Gambar 9.9 Hasil Uji Fungsi

## 9.8 Hindari tabrakan

Anda dapat meningkatkan panggilan fungsi menggunakan penambahan operator matematika (+), (-) pengurangan, (\*) perkalian, dan (/) divisi. Sebagai contoh, **Anda dapat mengurangi jarak suatu benda akan bergerak untuk menghindari tabrakan.**

## 9.9 Menggunakan operator matematika

- a. Suatu fungsi menentukan jarak antara Dalmatian dan Kelinci.
- b. Untuk mengurangi nilai yang dikembalikan oleh fungsi get Distance, operator pengurangan mengurangi nilai yang ditentukan.
- c. Nilai yang ditentukan ditentukan dengan memanggil fungsi get Width dan membagi nilai itu menjadi dua.



X = Dapatkan jarak dari Dalmatian ke  
Bunny



$$Z = X - (Y / 2)$$

Y = Dapatkan lebar

## 9.10 Periksa perhitungan matematika

Mari kita periksa perhitungan matematika  $Z = X - (Y / 2)$ :

- a. Z menunjukkan jarak total yang akan dipindahkan oleh Dalmatian.
- b. X mewakili jarak antara Dalmatian dan Kelinci.
- c. Y mewakili lebar Kelinci.
- d. Y / 2 mewakili lebar Kelinci dibagi 2.
- e. () mewakili urutan prioritas.

## 9.11 Tip operator matematika

- a. Mengapa kita membagi lebar Kelinci dalam perhitungan kita?
- b. Karena kami ingin animasi muncul seolah Dalmatian bergerak ke ujung Kelinci.
- c. Jika kita menggunakan seluruh lebar Kelinci, Dalmatian akan berhenti lebih jauh dari Kelinci daripada yang diinginkan



X = Dapatkan jarak dari Dalmatian  
ke Bunny

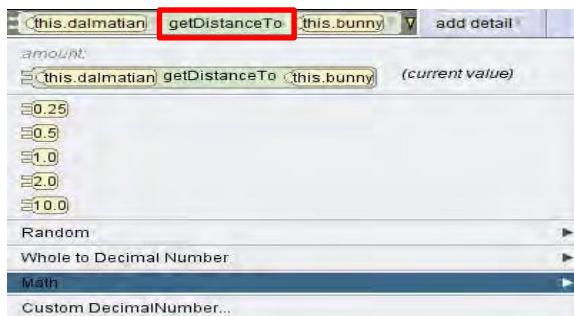


$$Z = X - (Y / 2)$$

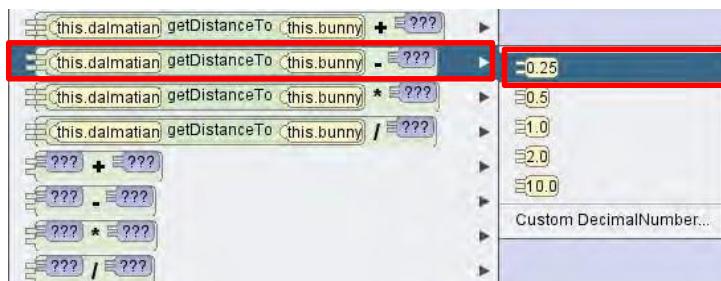
Y = Dapatkan lebar

## 9.12 Langkah-langkah menggunakan operator matematika untuk menghindari tabrakan

- Klik nama fungsi antara dua objek.
- Pilih Matematika.
- Pilih opsi pengurangan getDistanceTo.
- Pilih jumlah jarak tetap.



Gambar 9.10 Langkah-langkah menghindari tabrakan (a)



Gambar 9.11 Langkah-langkah menghindari tabrakan (b)

- Jalankan animasi untuk menguji bagaimana objek bergerak saat run-time.



Gambar 9.12 Hasil menghindari Tabrakan

- f. Sesuaikan nilai tetap untuk mendapatkan hasil yang diinginkan

The screenshot shows a Scratch script editor. On the left, there's a script for "this dalmatian" with a "move FORWARD" block followed by a "distance" block with a value of "0.25". To the right is a numeric slider with values ranging from -10.0 to 10.0. On the right side of the editor, there's a preview window showing the Dalmatian and the rabbit again, but now the Dalmatian is walking forward without jumping.

Gambar 9.13 Hasil Mengindari Tabrakan dengan waktu yang berbeda

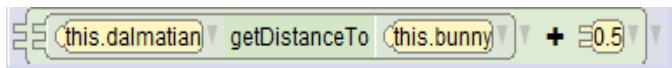
### 9.13 Memahami menu matematika contoh

- Gambar berikut menampilkan operator matematika (+ - \* /) yang membutuhkan satu atau dua argumen.
- Setiap opsi akan menyediakan satu atau dua menu cascading untuk menentukan nilai argumen.

The screenshot shows a Scratch script editor with a dropdown menu for mathematical operators. The menu is highlighted with a red border and contains options for addition (+), subtraction (-), multiplication (\*), and division (/). Below the menu, there are additional mathematical functions listed: min, max, absolute value, round, ceiling, floor, sqrt, pow, sin, cos, tan, asin, acos, atan, atan2, PI, and exp, log, E.

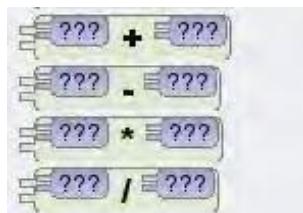
Gambar 9.14 Menu Matematika

- c. Gambar berikut menampilkan operator tambahan (+) yang membutuhkan argumen tunggal.



Gambar 9.15 Operator Tambahan (+) pada Menu Matematika

- d. Gambar berikut menampilkan operator matematika yang membutuhkan dua argumen.



Gambar 9. 16 Operator Matematika 2 Argumen

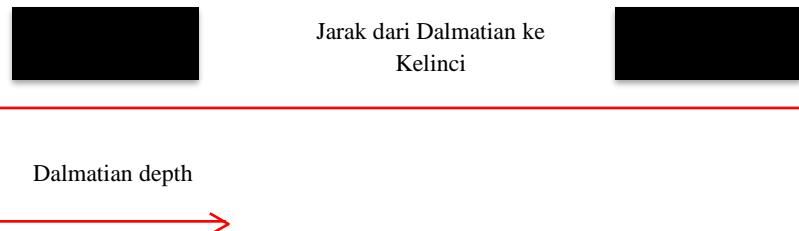
- g. Ingat, **Anda bisa memilih nilai placeholder untuk argumen.**  
h. **Nilai Placeholder selalu dapat didefinisikan.**

#### 9.14 Hapus depth objek dari fungsi

- a. Cara tepat lainnya untuk menghindari tabrakan adalah dengan menghilangkan depth (length) dari objek yang bergerak dari function.  
b. Dalam contoh di bawah ini, Dalmatian akan pindah jarak ke Kelinci, mengurangi depth Dalmatian.



Gambar 9.17 Menghapus Object DEPTH

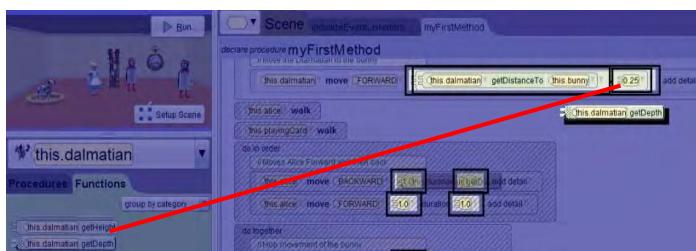


### 9.15 Depth diukur dari pusat objek

- Ketika nilai jarak dihitung, diukur dari pusat satu objek ke pusat objek lain.
- Hal yang sama berlaku untuk perhitungan matematika.
- Ketika depth Dalmatian dikurangi dari Kelinci, itu sebenarnya dikurangi dari pusat Kelinci.

### 9.16 Langkah-langkah untuk menghapus dept dari fungsi

- di tab functions, seret fungsi getdepth objek yang bergerak ke nilai jarak yang ditunjuk.
- jalankan animasi untuk menguji bagaimana objek bergerak saat run-time.
- sesuaikan dengan perhitungan matematika tambahan jika perlu.



Gambar 9.18 Menghapus DEPTH

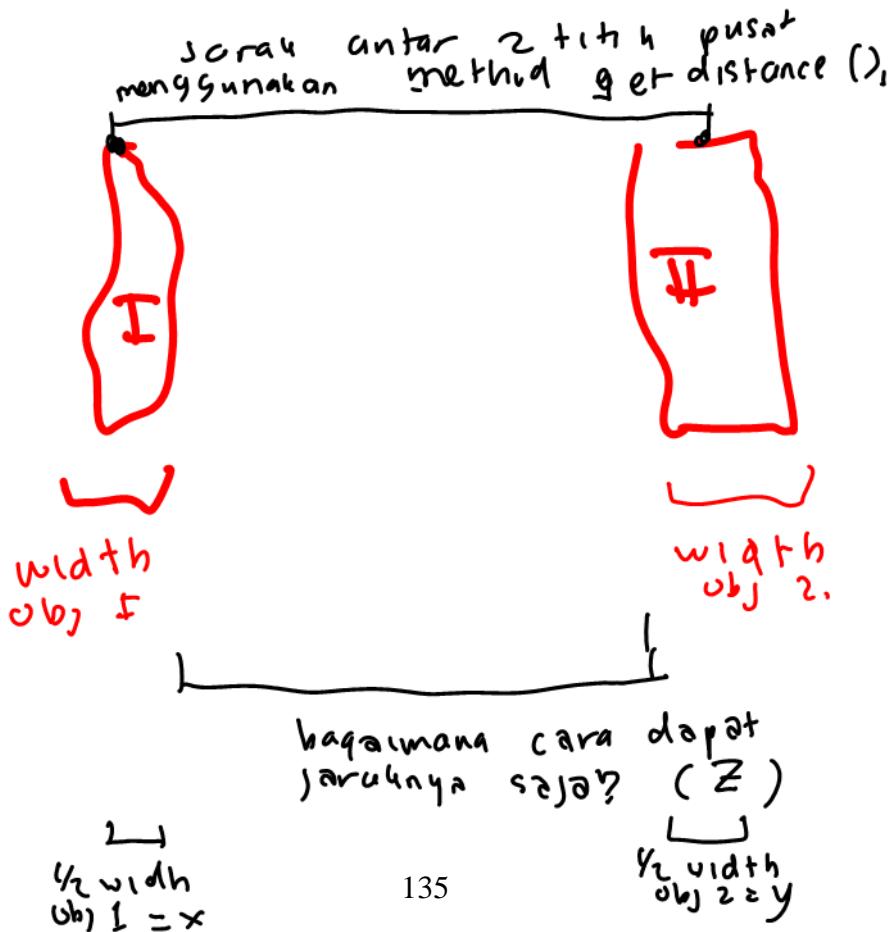
### 9.17 Terminologi

Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:

- Fungsi-fungsi
- Operator Matematika

### 9.18 Ringkasan

Dalam pelajaran ini, Anda belajar bagaimana menggunakan fungsi untuk mengontrol gerakan berdasarkan nilai pengembalian.



## BAB X

### STRUKTUR KONTROL IF dan WHILE

Pelajaran ini mencakup tujuan berikut:

1. Gunakan struktur kontrol IF untuk melaksanakan pelaksanaan instruksi
2. Gunakan struktur kontrol WHILE untuk membuat loop bersyarat untuk perilaku berulang

#### 10.1 Struktur Control

- a. Struktur kontrol adalah pernyataan yang telah ditentukan sebelumnya yang menentukan urutan instruksi pemrograman dieksekusi.
- b. Anda harus akrab dengan lakukan bersama-sama dan dalam rangka kontrol struktur dari topik sebelumnya.



Gambar 10.1 Struktur Control

#### 10.2 Struktur kontrol tersedia di Alice 3

- Struktur kontrol yang tersedia yang telah ditentukan meliputi:
- a. Lakukan dalam rangka
  - b. Menghitung
  - c. Jika
  - d. Untuk setiap

- e. Sementara
- f. Lakukan bersama
- g. Masing-masing bersama-sama

### 10.3 Tampilan struktur control

Anda dapat menyeret struktur kontrol ke myFirstMethod sebelum atau setelah membuat instruksi pemrograman yang akan dimasukkan dalam struktur kontrol.



Gambar 10.2 Tampilan Struktur Kontrol

### 10.4 Contoh struktur kontrol

Misalnya, jika Anda membuat gerakan dan memutar instruksi untuk objek, dan kemudian memutuskan bahwa tindakan harus dijalankan secara bersamaan, Anda bisa menyisipkan struktur kontrol Do together dan posisikan ulang gerakan dan putar instruksi dalam struktur kontrol.

Atau, Anda dapat mengantisipasi bahwa Anda akan memerlukan struktur kontrol Do together, masukkan struktur kontrol, dan kemudian buat dan posisikan instruksi pemrograman dalam struktur kontrol.

### 10.5 Struktur kontrol bersarang

- a. Struktur kontrol mungkin bersarang, yang berarti bahwa satu struktur terkandung di dalam yang lain.

- b. Misalnya, jika biped akan melambaikan tangan kiri mereka dan kemudian tangan kanan mereka saat bergerak maju, struktur kontrol bersarang akan diperlukan.

## 10.6 Contoh kode struktur kontrol bersarang



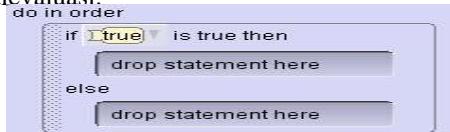
Gambar 10.3 Kode Struktur Kontrol Bersarang

## 10.7 Eksekusi bersyarat menggunakan struktur kontrol

- Struktur kontrol bersyarat memungkinkan Anda untuk mengontrol eksekusi berdasarkan suatu kondisi, atau keputusan dibuat.
- Pertimbangkan contoh-contoh ini:
  - Jika warna objek saat ini adalah biru, ubah warnanya menjadi oranye.
  - Jika jarak ke batu kurang dari 1 meter, bergerak maju 1/2 meter.
  - Jika objek opacity adalah 0, ubah opacity menjadi 1.

## 10.8 Struktur kontrol IF

- a. Struktur kontrol IF membutuhkan kondisi benar atau salah ketika struktur diseret ke dalam program.
- b. Kondisi awal ini adalah pengganti. Anda perlu menetapkan kondisi yang akan dievaluasi.



Gambar 10.4 Struktur Kontrol IF

## 10.9 Struktur control IF bagian

- a. Struktur kontrol IF memiliki dua bagian: bagian IF, dan bagian ELSE.
- b. Jika bagian IF dijalankan, bagian ELSE tidak pernah dieksekusi.
- c. Jika bagian ELSE dieksekusi, bagian IF tidak pernah dieksekusi.
- d. Bagian IF dan bagian ELSE dari pernyataan IF dapat berisi struktur kontrol IF bersarang lainnya.

## 10.10 Struktur Kontrol WHILE

- a. Pernyataan WHILE mengeksekusi instruksi berulang kali dalam satu lingkaran sementara suatu kondisi benar.
- b. Eksekusi bersyarat ini juga disebut “pengulangan”
- c. Kondisi WHILE:
  - 1) Bertindak seperti penjaga gerbang suatu event.
  - 2) Harus bernilai benar agar instruksi pengulangan dalam program dapat dijalankan
  - 3) Pengulangan akan keluar /berhenti saat kondisi bernilai salah

## 10.11 Struktur kontrol WHILE dan eksekusi perulangan

- a. Setelah semua instruksi pemrograman dalam satu loop dieksekusi, kondisi WHILE dievaluasi lagi untuk eksekusi berulang.

- b. Jika kondisi tetap bernilai benar, maka eksekusi akan terus dilakukan (perulangan).
- c. Jika kondisi bernilai salah, proses perulangan akan dilewati dan eksekusi akan dilanjutkan dengan pernyataan pemrograman selanjutnya mengikuti struktur kontrol WHILE.

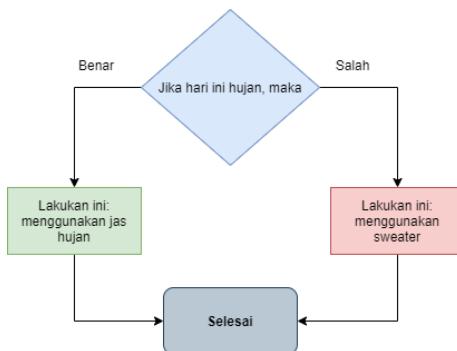
### 10.12 Menafsirkan struktur kontrol IF

- a. Struktur kontrol IF adalah keputusan berdasarkan pada suatu kondisi.
- b. Contoh kondisi:
  - 1) Jika hujan hari ini saya akan menggunakan jas hujan.
  - 2) Kalau tidak, saya akan menggunakan sweater.
- c. Struktur kontrol IF dapat diinterpretasikan menggunakan aliran proses.

 *“Aliran proses adalah representasi grafis dari model proses. Aliran proses menggunakan bentuk untuk mewakili tindakan dalam model.”*

### 10.13 Alur proses struktur kontrol IF

- a. Jika hari ini hujan, maka saya akan menggunakan jas hujan.
- b. Kalau tidak, saya akan menggunakan sweater



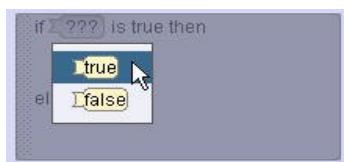
### 10.14 Contoh alur proses struktur kontrol IF

- a. Jika kucing bertabrakan dengan anjing, maka kucing akan mengatakan “maaf”
- b. Kalau tidak, kucing akan terus berjalan.



### 10.15 Langkah-langkah untuk memprogram struktur kontrol IF

- Masukkan gerakan awal yang terjadi sebelum struktur kontrol IF dijalankan.
- Seret dan letakkan struktur kontrol IF ke dalam editor Kode dan pilih kondisi sebenarnya sebagai pengganti selama pengaturan.
- Ganti kondisi yang sebenarnya dengan kondisi untuk mengevaluasi, seperti fungsi.
- Masukkan prosedur yang akan mengeksekusi jika kondisinya benar (IF) dan prosedur yang akan mengeksekusi jika kondisinya salah (ELSE).
- Jalankan animasi untuk menguji perilaku kondisional. Debug yang diperlukan.



Gambar 10.5 Struktur IF

### 10.16 Contoh struktur kontrol IF

- Jika kucing bertabrakan dengan anjing, maka kucing bergerak mundur dan berkata, "Maaf!"

- b. Kalau tidak, kucing akan terus bergerak maju.
- c. Fungsi `isCollidingWith` diseret ke placeholder kondisi sebenarnya.
- d. Fungsi ini memberi tahu kita jika satu objek bertabrakan dengan yang lain.



Gambar 10.6 Contoh Struktur Kontrol IF

### 10.17 Eksekusi bersyarat

Penggunaan struktur kontrol bersyarat memungkinkan dua jenis perulangan:

- a. Conditional loop: berhenti ketika kondisi benar.

contoh: Baling-baling helikopter berputar saat helikopter bergerak atau terbang. Jika helikopter berhenti, maka baling-baling berhenti berputar.

- b. Infinite loop: tidak pernah berhenti

Contoh: Jarum jam dan menit pada jam terus bergulir.



Gambar 10.7 Conditiona Loop

Gambar 10.8 Infinite Loop

### 10.18 Struktur kontrol WHILE

- a. Struktur kontrol WHILE melakukan conditional loop
- b. Ketika suatu kondisi benar, instruksi pemrograman dalam loop dieksekusi.

- c. Setelah kondisi tidak lagi benar, eksekusi program akan melewati kondisi WHILE dan melanjutkan dengan instruksi pemrograman yang mengikuti lperulangan WHILE.

“Struktur kontrol WHILE akan melakukan instruksi ketika suatu kondisi benar; jika tidak maka akan melewati instruksi.”

#### 10.19 Alur proses struktur kontrol WHILE

- The QUEEN bergerak maju, kecuali dia bertabrakan dengan PLAYING CARD.
- Jika The QUEEN bertabrakan dengan PLAYING CARD, ia berhenti dan berbalik menghadap kamera.



#### 10.20 Langkah-langkah untuk memprogram struktur kontrol WHILE

- Seret dan taruh struktur kontrol WHILE ke dalam editor Kode dan pilih kondisi sebenarnya sebagai placeholder.
- Ganti placeholder kondisi sebenarnya dengan kondisi untuk mengevaluasi.
- Masukkan prosedur yang akan dieksekusi saat kondisinya benar.
- Masukkan prosedur yang dieksekusi setelah perulangan WHILE berhenti mengeksekusi.

## 10.21 Contoh kode struktur kontrol WHILE

- Sementara Ratu tidak bertabrakan dengan Kartu Bermain, Ratu bergerak maju berulang kali.
- Jika Ratu bertabrakan dengan Kartu Bermain, loop WHILE berhenti dan program melanjutkan dengan Intruksi Selanjutnya:
  - Dia berhenti dan berbalik menghadap kamera.



Gambar 10.9 Struktur Kontrol While



Gambar 10.10 Hasil Kontrol While

## 10.22 Langkah-langkah untuk menguji struktur kontrol WHILE

- Posisikan objek sedemikian rupa sehingga kondisi WHILE akan dievaluasi menjadi true.
- Perhatikan bahwa semua instruksi pemrograman dalam loop WHILE dijalankan.
- Pastikan bahwa eksekusi loop WHILE berhenti ketika kondisi sementara tidak lagi benar.



Gambar 10.11 Pengujian Kontrol While (a)



Gambar 10.12 Pengujian Kontrol While (b)



Gambar 10.13 Pengujian Kontrol While (c)

### 10.23 Terminologi

Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:

- Struktur Kontrol IF
- Aliran Proses
- Struktur Kontrol WHILE

### 10.24 Ringkasan

Dalam pelajaran ini, Anda seharusnya belajar bagaimana:

- Gunakan struktur kontrol IF untuk melakukan eksekusi instruksi

- b. Gunakan struktur kontrol WHILE untuk membuat loop bersyarat untuk perilaku berulang

## BAB XI

### EKSPRESI

Pelajaran ini mencakup tujuan-tujuan berikut:

1. Ciptakan ekspresi untuk mengerjakan operasi matematika
2. Terjemahkan ekspresi matematika

#### 11.1 Menggunakan ungkapan

- a. Pernyataan merupakan kombinasi nilai-nilai yang kapan di atur dengan benar, menghasilkan nilai akhir.
- b. Ekspresi biasanya digunakan di Alice 3 untuk memecahkan waktu dan masalah jarak dalam program anda.
- c. Contoh  $2 + 2 = 4$
- d. Dua nilai (2,2) dan operator (+) menghasilkan nilai akhir (4)

#### 11.2 Ekspresi di Alice 3

Ekspresi diciptakan di Alice 3 menggunakan berikut terdapat didalam matematika operator:

- a. Penjumlahan (+)
- b. Pengurangan (-)
- c. Perkalian (\*)
- d. Pembagian (/)

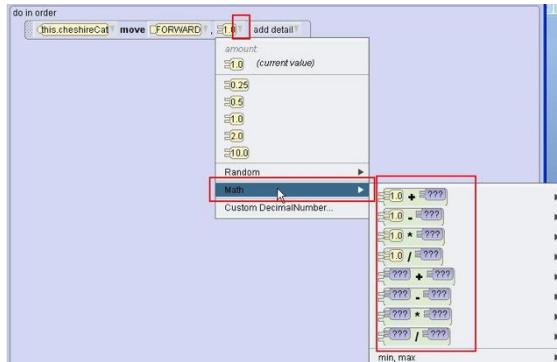
#### 11.3 Lokasi operator matematika

Operator matematika tersedia di Cascading Menus pilih nilai argument untuk:

- a. Jumlah dan durasi
- b. Fungsi jaga jarak

## 11.4 Pandangan ekspresi dengan argument jarak jauh

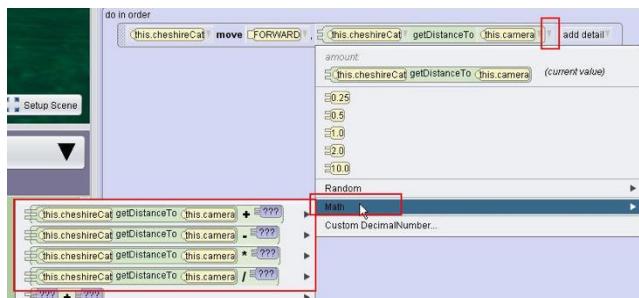
- Pilihlah opsi matematika untuk memandang matematika yang tersedia operator berdebat jarak jauh.
- Pilihlah dari dua rangkaian pernyataan matematika yang berbeda:
  - Set pertama memungkinkan anda menentukan nilai satu operator
  - Set kedua memungkinkan anda menentukan nilai – nilai kedua operator



Gambar 11.1 Ekspresi dengan argumen jarak jauh

## 11.5 Tampilan fungsi

Pilih opsi Matematika untuk melihat operator matematika yang tersedia untuk argumen fungsi `getDistanceTo`.



Gambar 11.2 Tampilan Argumen Fungsi

## 11.6 Masalah jarak

- Masalah: Objek PlayingCard bergerak ke tengah rak buku, bukan di dekat itu.
- Ini karena fungsi `getDistanceTo` menghitung jarak dari pusat objek orang ke pusat objek target (rak buku).
- Kita perlu mengurangi jarak objek PlayingCard bergerak sehingga tidak bertabrakan dengan rak buku.
- Perhitungan matematika digunakan untuk mengurangi jarak objek PlayingCard bergerak.

## 11.7 Langkah-langkah untuk membuat ekspresi

- Ringkas masalah waktu atau jarak dalam program Anda.
- Pertimbangkan ekspresi yang akan menyelesaikan masalah.
- Kode ekspresi.
- Uji dan debug ekspresi hingga animasi berfungsi sebagaimana yang dimaksud.



Gambar 11.3 Langkah Membuat Ekpresi

## 11.8 Langkah-langkah untuk memindahkan objek, jarak ke objek lainnya

- Seret prosedur pindah untuk objek ke editor Kode.
- Pilih maju dan nilai placeholder jarak.



Gambar 11.4 Langkah Memindahkan Jarak (a)

- Dari tab Functions, seret fungsi `getDistanceTo` ke placeholder argumen jarak.



Gambar 11.5 Langkah Memindahkan Jarak (b)

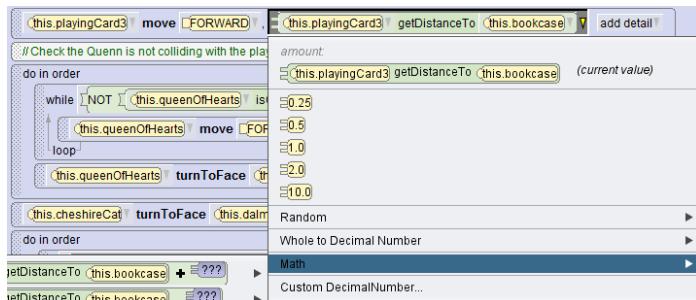
- d. Dari menu cascading, pilih objek target yang objek harus bergerak.



Gambar 11.6 Langkah Memindahkan Jarak dan Objek

### 11.9 Langkah-langkah untuk memodifikasi jarak menggunakan operator matematika

Dari bar getDistanceTo, klik panah paling luar untuk membuka menu nilai jarak, lalu pilih opsi Matematika.



Gambar 11.7 Langkah Memodifikasi Jarak

### 11.10 Langkah-langkah untuk merubah jarak menggunakan operator matematika

- Pilih getDistanceTo - ???
- Pilih nilai default untuk mengurangi jarak, atau pilih Custom DecimalNumber ... untuk memasukkan nilai.



Gambar 11.8 Langkah Merubah Jarak

- Uji dan jalankan ekspresi yang diperlukan

## 11.11 Mengubah nilai ekspresi

- Selama proses debug, kamu mungkin perlu untuk menyesuaikan nilai ekspresi
- Klik tanda panah disebelah nilai, dan pilih nilai default yang baru atau gunakan menu Custom DecimalNumber... untuk memasukkan nilai tertentu



Gambar 11.9 Megubah Nilai Ekspresi

## 11.12 Contoh nilai ekspresi

- Ekspresi ini mengurangi jarak pindah dari Playing Card sehingga tidak bertabrakan dengan rak buku
- Ini dicoba dan dijalankan beberapa kali hingga mencapai tujuan yang diinginkan
- Dengan nilai 0.25 Playing Card masih terlalu dekat dengan rak buku



Gambar 11.10 Langkah Nilai Ekspresi

- Dengan nilai 1.0 barulah jarak yang dihasilkan pas



Gambar 11.11 Hasil Nilai Ekspresi

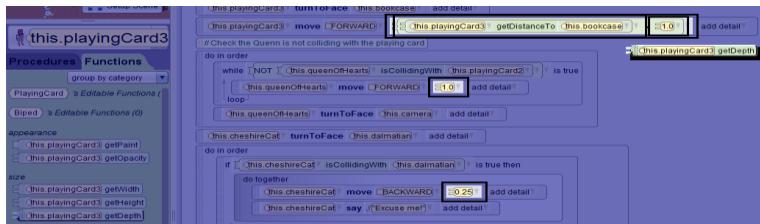
## 11.13 Kurangi kedalaman dari ekspresi



Mengurangi kedalaman objek target dari ekspresi adalah cara yang lebih tepat untuk memastikan bahwa objek bergerak mendarat langsung di dekat objek target tanpa melalui pusatnya.

## 11.14 Langkah – langkah untuk mengurangi kedalaman dari ekspresi

- Pilih objek target di menu contoh.
- Dari tab Functions, seret getDepth ke nilai jarak yang ada dalam ekspresi.



Gambar 11.12 Langkah Mengurangi Kedalaman Ekspresi

- Uji dan jalankan animasi, dan sesuaikan ekspresi seperlunya.

## 11.15 Menafsirkan ekspresi

- Untuk memahami pernyataan pemrograman yang mencakup ekspresi, Anda sering perlu menafsirkan ekspresi.
- Untuk menafsirkan ekspresi Anda dapat:
  - Baca dari kiri ke kanan.
  - Kenali instance yang ditentukan dalam ekspresi dan jelaskan apa yang dilakukan masing-masing.



Gambar 11.13 Langkah Menafsirkan Ekspresi

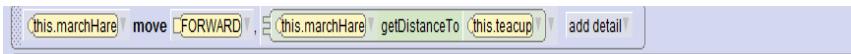
## 11.16 Membangun ekspresi contoh

- a. Tambahkan prosedur pindah dari marchHare dan gunakan a nilai placeholder.



Gambar 11.14 Langkah Membangun Ekspresi (a)

- b. Seret fungsi getDistanceTo dari marchHare dan pilih cangkir teh sebagai objek target



Gambar 11.15 Langkah Membangun Ekspresi (b)

- c. Klik pada panah luar ekspresi.



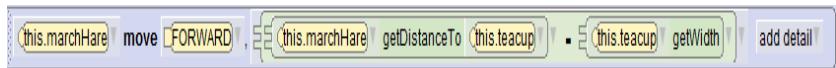
Gambar 11.16 Langkah Membangun Ekspresi (c)

- d. Pilih matematika dari daftar dan pilih kurang pilihan.



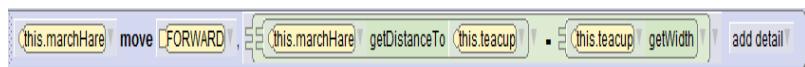
Gambar 11.17 Langkah Membangun Ekspresi (d)

- e. Ganti nilai placeholder dengan getWidth fungsi dari kelas cangkir teh



Gambar 11.18 Langkah Membangun Ekspresi (e)

- f. Klik pada panah dalam setelah getWidth dan pilih Matematika, lalu operator divisi dan pilih 2 sebagai nilai.



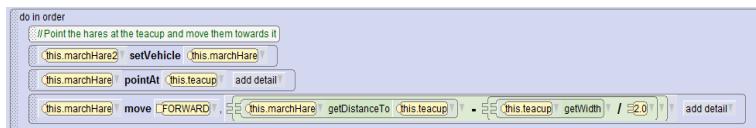
Gambar 11.19 Langkah Membangun Ekspresi (f)

- g. Ini melengkapi ekspresi yang seharusnya terlihat seperti ini:

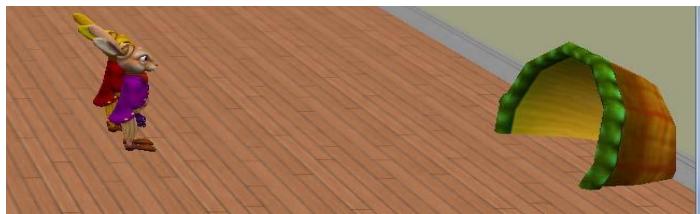


Gambar 11.20 Langkah Membangun Ekspresi (g)

- h. Periksa visual yang terkait dengan ungkapan ini.
- i. Kelinci bergerak menuju cangkir teh.
- j. Apakah Anda pikir mereka akan masuk ke dalam?



Gambar 11.21 Langkah Membangun Ekspresi (h)



Gambar 11.22 Hasil Ekspresi

### 11.17 Interpretasi ekspresi

Ungkapan ini memberi tahu kita hal berikut:

- a. MarchHare bergerak maju menuju cangkir teh.
- b. Jarak antara marchHare dan cangkir teh adalah ditentukan oleh fungsi getDistanceTo.
- c. Jarak yang ditempuh berkurang setengah dari lebar cangkir teh. Lebar cangkir teh ditentukan oleh getWidth fungsi.



Gambar 11.23 Langkah Interpretasi Ekspresi

### 11.18 Merumuskan ekspresi

- Untuk menafsirkan ekspresi, akan sangat membantu untuk menggambar atau tulis nilai yang Anda tahu sebelum merumuskan ekspresi.
- Contoh: -  $Z = X - (a / b)$ 
  - $Z$  = Jarak dipindahkan
  - $X$  = Jarak dari marchHare ke cangkir teh
  - $a = \text{Lebar cangkir teh} \cdot b = 2$  JF 2-9 Ekspresi

### 11.19 Contoh ekspresi jawaban

- Anda ditanya apakah Anda pikir marchares akan melakukannya pergi ke cangkir teh?
- Jawabannya adalah: Tidak, mereka berhenti di luar cangkir teh.
- Ini karena kami menggunakan ekspresi untuk memanipulasi jarak antar objek.



Gambar 11.24 Hasil Ekspresi Jawaban

### 11.20 Terminologi

Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:

- Ekspresi
- Operator matematika

### 11.21 Ringkasan

Dalam pelajaran ini, Anda seharusnya belajar bagaimana:

- a. Buat ekspresi untuk melakukan operasi matematika
- b. Menafsirkan ekspresi matematika.

## BAB XII

### VARIABEL

Materi ini meliputi beberapa tujuan:

1. Memahami variable
2. Memahami bagaimana kegunaan variable pada pemrograman
3. Memperhatikan kode Alice sebagai kode java di samping

#### 12.1 Variable

Terkadang, programmer perlu menyimpan informasi dan memakai informasi tersebut dalam pembuatan animasi atau game, contoh:

- a. Berapa kali sebuah prosedur harus dijalankan
- b. Property sebuah objek, seperti ukuran dan warna

*“Variabel adalah tempat pada memori dimana data dari sebuah tipe spesifik bisa disimpan untuk diambil dan digunakan lagi nanti oleh program anda. Tiap variable diberi nama yang unik utik mempermudan pencarian dan referensi. Sekali sebuah variable dideklarasi, itu dapat dipakai untuk menyimpan dan mengambil ulang data.”*

#### 12.2 Contoh variable

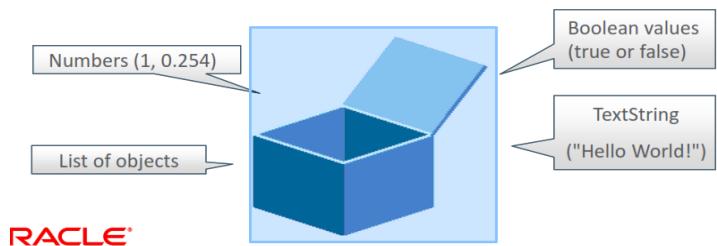
Berikut ini adalah nilai dan variable dari dalmation



Gambar 12.1 Nilai Variabel dalmation

### 12.3 Variable untuk penyimpanan data

Sebuah variable mirip dengan container yang menyimpan data spesifik. Variabel seperti wadah yang menyimpan tipe tertentu data untuk pengambilan nanti dan digunakan oleh program Anda. Deklarasi sebuah variable dengan menamainya dan memilih jenis data untuk menyimpannya lalu tentukan nilai default variable tersebut.



Gambar 12.2 Variabel Penyimpanan Data

### 12.4 Properti objek

Properti objek adalah variable yang menyimpan informasi tentang objek tersebut, seperti warna, lebar, tinggi, dan kedalaman.



Gambar 12.3 Properti Objek

## 12.5 Jenis data variabel di Alice 3

Tabel 12.1 Jenis Variabel

Tipe data	Deskripsi
<b>Angka desimal</b>	<ul style="list-style-type: none"> <li>Lakukan aritmatika dan tetapkan nilai argumen prosedur.</li> <li>Contoh: 0,1, 2,25, 98,6.</li> </ul>
<b>Seluruh Angka</b>	<ul style="list-style-type: none"> <li>Lakukan aritmatika dan tetapkan nilai argumen prosedur.</li> <li>Contoh: 1, 459, 30.</li> </ul>
<b>Boolean</b>	<ul style="list-style-type: none"> <li>Satu dari dua nilai: benar atau salah.</li> <li>Biasanya merupakan hasil tes yang membandingkan satu hal dengan hal lainnya.</li> </ul>
<b>Kelas</b>	<ul style="list-style-type: none"> <li>Kelas objek yang tersedia di animasi Anda.</li> <li>Contoh: Berkaki 2, Scene, Berkaki 4.</li> </ul>

<b>TextString</b>	<ul style="list-style-type: none"> <li>• String karakter seperti "halo" dan "selamat tinggal".</li> </ul>
<b>Lainnya</b>	<ul style="list-style-type: none"> <li>• Suara, warna, bentuk, dan nilai khusus lainnya.</li> </ul>

## 12.6 Mendeklarasikan variabel

Untuk mendeklarasikan (atau "membuat") suatu variabel adalah **memberikan nama variabel dan untuk menentukan jenis data yang akan berisi variabel**. Variabel dideklarasikan dalam editor Kode. Mereka berguna karena memungkinkan Anda untuk:

- a. Tetapkan nilai yang sama untuk beberapa prosedur, seperti jarak untuk bergerak.
- b. Secara bersamaan memperbarui nilai semua argumen dalam program yang mereferensikan variabel.
- c. Berikan informasi dari satu prosedur ke prosedur lainnya.
- d. Sederhanakan pernyataan pemrograman menggunakan banyak fungsi dan ekspresi matematika.

## 12.7 Inisialisasi variabel

Untuk menginisialisasi variabel berarti memberikan nilai. Variabel diinisialisasi dalam editor Kode pada saat yang sama ketika mereka dinyatakan; ini adalah nilai awalnya (ditugaskan pertama). Nilai variabel dapat diubah sesering yang Anda suka. **Kata "inisialisasi" berarti "tetapkan nilai"**.

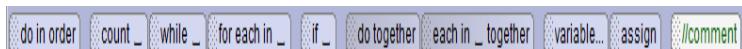
## 12.8 Mengubah nilai yang diinisialisasi

Ingat, nilai awal yang Anda tentukan untuk variabel dapat dianggap sebagai nilai placeholder, dan diubah di lain waktu. Anda dapat mengubah nilai

yang diinisialisasi dari suatu variabel menggunakan daftar drop-down. Nilai baru dari semua argumen yang menggunakan variabel akan berubah ketika nilai yang diinisialisasi diubah. Jika salah satu nilai default yang tercantum pada drop-down bukan yang Anda butuhkan, gunakan menu Opsi kustom untuk menentukan nilai lain.

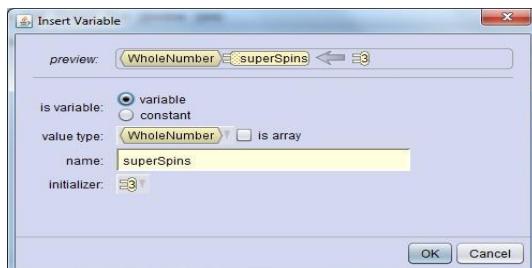
## 12.9 Langkah-langkah untuk mendeklarasikan variabel

- Seret bagian variable ke editor Kode.



Gambar 12.4 Langkah Mendeklarasikan Variabel

- Pilih jenis nilai dan beri nama variabel.
- Inisialisasi variabel (atur nilai pertama yang akan disimpan variabel) dan klik OK.
- Perhatikan preview variabel, di atas garis tipis, yang mana akan menampilkan pengaturan variabel.



Gambar 12.5 Tampilan Insert Variabel

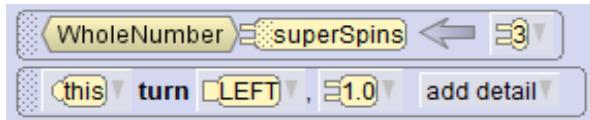
## 12.10 Contoh variabel

- Variabel "superSpins" dideklarasikan dan diinisialisasi ke seluruh nomor dengan nilai awal 3



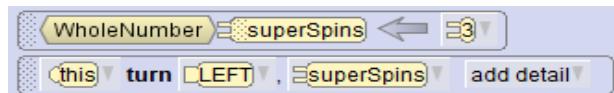
Gambar 12.6 Contoh Variabel (a)

- b. Gunakan prosedur " turn" untuk memutar karakter



Gambar 12.7 Contoh Variabel (b)

- c. Variabel ini diseret ke argumen jarak prosedur " turn" atau belokan.

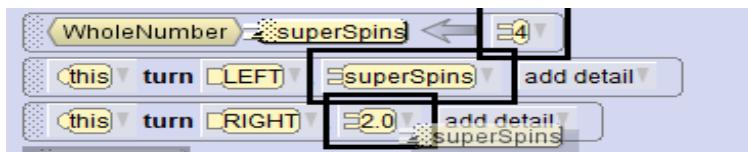


Gambar 12.8 Contoh Variabel (c)

- d. Jika nilai inisialisasi "superSpins" diubah menjadi 4, semua karakter akan berputar berdasarkan nilai variabel, yang sekarang 4.

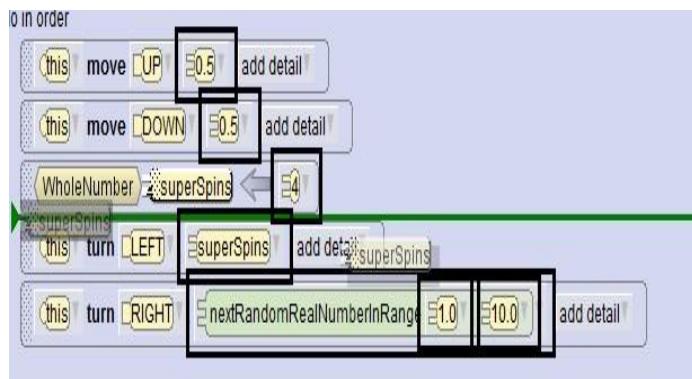
## 12.11 Menggunakan variabel dalam prosedur

- Untuk menggunakan variabel yang telah dideklarasikan dalam prosedur, seret ubin nama variabel ke nilai argumen prosedur.
- Argumen diganti dengan nilai inisialisasi variabel.
- Alice 3 membantu Anda memvisualisasikan lokasi Anda dapat menempatkan variabel dengan menggelapkan layar dan menyorot nilai-nilai yang dapat diganti oleh variabel.



Gambar 12.9 Langkah Menggunakan Variabel dalam Prosedur (a)

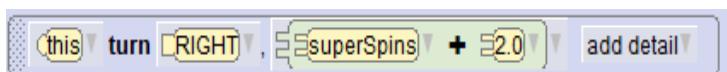
- d. Ketahuilah bahwa suatu variabel harus dideklarasikan dan diinisialisasi sebelum dapat dirujuk oleh pernyataan lain dalam kode Anda.
- e. Jika Anda mencoba mereferensikan variabel yang belum ada, program akan error saat dijalankan.
- f. Perhatian: Alice menyoroti semua argumen yang mungkin berpotensi merujuk variabel yang dipilih, termasuk argumen yang mendahului keberadaan variabel.



Gambar 12.10 Langkah Menggunakan Variabel dalam Prosedur (b)

### 12.12 Menggunakan variabel dalam perhitungan matematika

- a. Perhatikan bahwa variabel juga dapat digunakan dalam perhitungan matematika.
- b. Anda bisa menyeret variabel yang dideklarasikan ke nilai apa pun dalam ekspresi matematika.



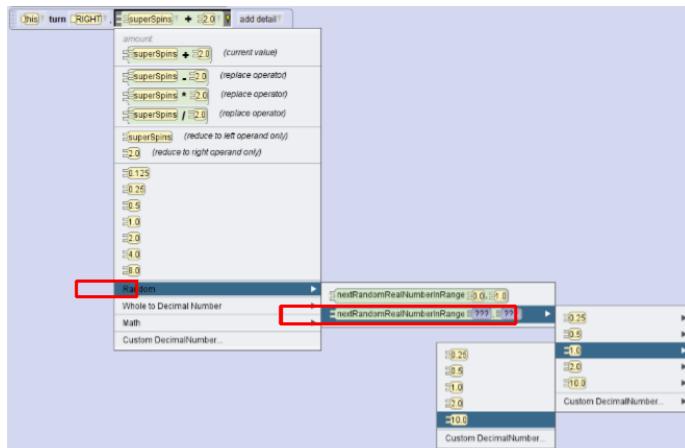
Gambar 12.11 Langkah Variabel dalam Perhitungan Matematika

## 12.13 Langkah-langkah untuk mengacak nilai yang diinisialisasi

- Klik panah bawah di sebelah nilai yang diinisialisasi.
- Pilih Acak dari daftar drop-down.
- Pilih opsi untuk mengacak menggunakan rentang yang telah ditentukan atau opsi untuk mengacak berdasarkan nilai yang Anda buat.
- Jika Anda memilih opsi untuk mengacak berdasarkan nilai yang Anda buat, pilih nilai awal dan akhir untuk rentang menggunakan menu cascading.
- Ingat, nilai argumen bisa selalu berubah.
- Pengacakan dari nilai variabel bisa menambah nilai ke animasi atau game dengan membuat perilaku yang acak.

## 12.14 Tampilan pengacakan sebuah nilai yang diinisialisasikan

Di bawah ini adalah cara untuk mengacak sebuah nilai yang diinisialisasi.



Gambar 12.12 Pengacakan Nilai yang diinisialisasikan

## 12.15 Menampilkan kode alice dalam bahasa java

- Untuk melihat kode yang sudah dibuat dengan lingkungan kode Java tradisional Alice mempunyai Java di opsi Side.

- b. Gunakan opsi menu WIndow, lalu preferensi dan kode Java untuk mengaktifkan Window.



Gambar 12.13 Menampilkan Kode Alice dalam bahasa Java (a)

- c. Ini memungkinkan Anda untuk melihat struktur programming tradisional seperti semi-colons(;) untuk mengakhiri statements dan brackets bergelombang untuk mengawali ({) dan () untuk mengakhiri statements.

```

declare procedure spins [ Add Parameter... ]
do in order
  [this move UP, 0.5 add detail]
  [this move DOWN, 0.5 add detail]
  [WholeNumber superSpins ← 4]
  [this turn LEFT, superSpins add detail]
  [this turn RIGHT, nextRandomRealNumberInRange 1.0, 10.0 add detail]

public void spins() {
    this.move( MoveDirection.UP, 0.5 );
    this.move( MoveDirection.DOWN, 0.5 );
    Integer superSpins = 4;
    this.turn( TurnDirection.LEFT, superSpins );
    this.turn( TurnDirection.RIGHT, RandomUtilities.nextDoubleInRange( 1.0, 10.0 ) );
}

```

Gambar 12.14 Menampilkan Kode Alice dalam bahasa Java (b)

- d. Melihat Kode Alice sebagai Java, Perubahan yang dibuat dalam kode Alice tercermin dalam kode java. Anda tidak dapat mengubah kode java secara langsung itu hanya representasi dari kode Alice.

```

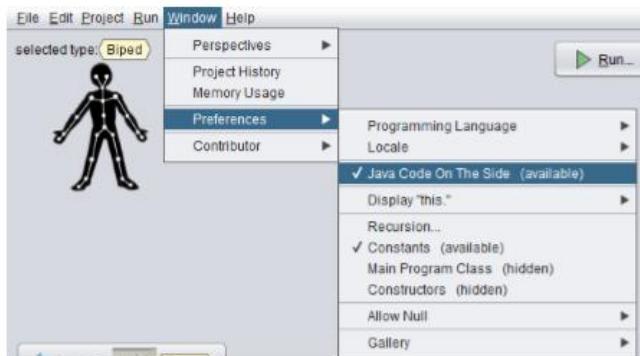
declare procedure spins [ Add Parameter... ]
do in order
  [this] move UP .05! add detail
  [this] move DOWN .05! add detail
  WholeNumber superSpins <-- [ ] 
  [this] turn LEFT .superSpins [ ] 
  [this] turn RIGHT .nextRand [ ] 
  Random
  Decimal to Whole Number
  Math
  Custom WholeNumber...

```

The screenshot shows the Alice software interface with the 'spins' procedure open. On the left, there's a tree view of the procedure steps. On the right, the Java code for the procedure is displayed. The Java code includes imports for MoveDirection, RandomUtilities, and Integer. It contains logic to move up and down, set a variable 'superSpins', turn left or right based on the value of 'superSpins', and handle random values.

Gambar 12.15 Menampilkan Kode Alice dalam bahasa Java (c)

- Untuk mematikan Kode Java pada opsi samping dan kembali ke antarmuka Alice saja pergi ke menu jendela, preferensi dan kemudian hapus centang pada Kode Java pada opsi samping.



Gambar 12.16 Menampilkan Kode Alice dalam bahasa Java (d)

## 12.16 Terminologi

**Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:**

- Variabel
- Properti objek
- Mendeklarasikan variable
- Inisialisasi variable

- e. Kode Java di samping

### **12.17 Ringkasan**

Dalam pelajaran ini, Anda seharusnya belajar bagaimana:

- a. Memahami variabel
- b. Memahami bagaimana variabel digunakan dalam pemrograman
- c. Melihat kode Alice sebagai Kode Java di samping

## **BAB XIII**

### **KONTROL PADA KEYBOARD**

Pelajaran ini mencakup tujuan-tujuan berikut:

1. Membuat urutan pembukaan
2. Menggunakan kontrol keyboard untuk memanipulasi animasi
3. Menyimpan file Kelas Anda
4. Menggunakan tab starter
5. Menambahkan file kelas yang ada ke animasi

#### **13.1 Penanganan acara**

Saat animasi diputar, program komputer sedang melakukan proses. Banyak program komputer meminta pengguna melakukan interaksi. **Program interaktif ini memungkinkan pengguna untuk mempengaruhi urutan tindakan yang terjadi dalam program.** Untuk memprogram jenis interaktivitas ini menjadi animasi, dengan membuat **Event Listener yang mencari dan merespons interaktivitas (input peristiwa pengguna)** dari pengguna. Ini sering disebut sebagai penanganan acara. (event handling)

#### **13.2 Apa itu acara?**

**Acara adalah segala tindakan yang dilakukan oleh pengguna yang dirancang untuk mempengaruhi pelaksanaan program selama bermain.**

**Acara dapat meliputi:**

- a. **Menekan tombol** apa saja pada **keyboard**
- b. **Mengklik tombol mouse**
- c. **Memindahkan joystick**
- d. **Menyentuh layar (pada perangkat yang mendukung sentuhan)**

### **13.3 Apa yang terjadi ketika suatu acara terjadi?**

Biasanya, **suatu peristiwa memicu** (menembak, atau menggerakkan) **pelaksanaan suatu prosedur atau fungsi**. Misalnya, ketika pengguna menekan tombol panah ke atas keyboard (acara), itu memicu metode yang membuat objek dalam animasi bergerak ke atas (penanganan acara metode).

### **13.4 Kontrol keyboard**

Memasukkan kontrol keyboard ke program memungkinkan pengguna untuk mengontrol satu atau lebih objek saat animasi sedang berlari. Pengguna dapat menekan tombol pada keyboard, atau klik mouse, untuk mengontrol aksi pemrograman selanjutnya. Dengan kontrol keyboard, Anda dapat:

- a. Membuat adegan di mana pengguna mengontrol objek itu berinteraksi dengan objek lain.
- b. Membuat animasi yang dijalankan secara kondisional, berdasarkan tekan tombol atau klik mouse.
- c. Membuat game di mana pengguna diminta untuk mengendalikan sebuah keberatan untuk memenangkan permainan.

### **13.5 Contoh kontrol keyboard**

- a. Di Alice 3, Anda dapat **menetapkan prosedur untuk kunci pada papan ketik**.
- b. Ketika penampil animasi mengklik perintah di papan ketik maka prosedur akan dieksekusi.
- c. Misalnya, **mengklik tombol panah kanan pada Keyboard mengubah teko ke kanan**.

### **13.6 Event listener**

**Event Listener** adalah prosedur dalam kelas Scene itu **dengarkan input keyboard saat animasi sedang berjalan**. Tombol keyboard dapat diprogram untuk:

- a. Memindahkan objek ke atas atau ke bawah saat tombol tertentu ditekan.

- b. Memindahkan objek ke depan, ke belakang, ke kiri, dan ke kanan menggunakan tombol panah.
- c. Membuat objek melakukan tindakan, seperti berbicara atau menghilang.

### 13.7 Jenis event listener

Ada empat jenis Event Listener yang tersedia di Alice 3:

- a. Aktivasi / Waktu Pemandangan
- b. Keyboard
- c. Mouse
- d. Posisi / Orientasi

### 13.8 Langkah-langkah untuk mengakses event listener

- a. Di editor Kode, klik tab Pemandangan.
- b. Klik tombol di sebelahnya untuk menginisialisasi Event Listeners dan pilih Edit.
- c. Perintah ini membuka tab initialize Event Listeners.

### 13.9 Tab event listener

Tab initialize EventListeners adalah tempat Anda dapat menambahkan Event Listener untuk kode Anda. Pendengar sceneActivated adalah tempat Anda dapat membuat animasi yang akan diputar di depan myFirstMethod dimulai. Ini dapat digunakan sebagai urutan pembuka untuk kode Anda.

### 13.10 Pendengar aktivasi adegan

Anda dapat membuat prosedur sendiri atau menggunakan salah satu dari prosedur bawaan untuk membuat urutan pembuka. Prosedur bernama "muncul" ini mengubah Alice dan membuatnya terlihat di adegan kami. Untuk mengaktifkan prosedur, seret dari Alice daftar prosedur dan letakkan di depan myFirstMethod hubungi pendengar sceneActivated. Uji urutan pembukaan Anda!

### 13.11 Listener keyboard

Pendengar keyboard:

- a. Ditemukan di menu tarik-turun Tambah Event Listener.
- b. Dengarkan, dan bereaksi terhadap, penekanan tombol keyboard yang Anda tentukan.

### 13.12 Jenis listener keyboard

Tabel 13.1 Jenis Pendengar Keyboard

Tipe data	Deskripsi
<code>addKeyPressListener ()</code>	Pendengar ini memungkinkan Anda memprogram prosedur untuk <b>tombol keyboard yang Anda tentukan.</b>
<code>addArrowKeyPressListener ()</code>	Pendengar ini memungkinkan Anda memprogram prosedur untuk <b>tombol panah yang Anda tentukan.</b>
<code>addNumberKeyPressListener ()</code>	Pendengar ini memungkinkan Anda memprogram prosedur untuk <b>tombol angka yang Anda tentukan.</b>
<code>addObjectMoverFor (???)</code>	Pendengar ini memungkinkan Anda memprogram pengguna <b>gerakan yang ditentukan untuk objek tertentu.</b>

### 13.13 Program keyboard event listener

Misalnya, kami akan memprogram Event Listener papan ketik untuk memerintahkan teko untuk bergerak naik dan turun menggunakan Tombol B

dan C, dan bergerak ke kiri, kanan, maju dan mundur menggunakan tombol panah. Bagaimanapun, kita berada di negeri ajaib!

### **13.14 Langkah-langkah untuk menambahkan event listener keyboard**

- a. Pilih daftar drop-down Tambah Event Listener.
- b. Pilih Keyboard.
- c. **Pilih addKeyPressListener.**

### **13.15 Langkah-langkah untuk memprogram struktur kontrol if**

- a. **Seret struktur kontrol IF ke addKeyPressListener.**
- b. Pilih kondisi sebenarnya.

### **13.16 Langkah-langkah untuk memilih kunci keyboard untuk diaktifkan gerak objek**

- a. **Seret isKey: ??? ubin ke kondisi sebenarnya.**
- b. Menu kunci muncul.
- c. Dari menu drop-down, **pilih tombol keyboard yang ingin Anda gunakan untuk mengontrol gerakan.**

### **13.17 Langkah-langkah untuk program gerakan diaktifkan oleh kunci tekan**

- a. Dari menu drop-down Instance, **pilih objek dikontrol** oleh tombol keyboard.
- b. **Seret prosedur yang harus diaktifkan oleh tombol keyboard ke dalam struktur kontrol IF dan pilih argumen.**
- c. Anda dapat menarik banyak prosedur dan struktur kontrol ke dalam struktur kontrol IF.
- d. Misalnya, ketika tombol B ditekan, teko bergerak dan kemudian berbalik

### **13.18 Program tindakan pendengar tambahan**

Untuk memprogram keyPressListerner untuk mendengarkan lebih dari satu tombol keyboard, tambahkan struktur kontrol IF tambahan ke struktur pendengar.

Ada dua cara untuk melakukan ini:

- a. Tambahkan serangkaian struktur kontrol IF satu demi satu dan selalu biarkan kondisi ELSE kosong.
- b. Sarang tambahan struktur kontrol IF dalam kondisi ELSE.

Kedua metode mengeksekusi dengan cara yang sama. Langkah-langkah berikut menggunakan metode kedua, bersarang IF kontrol struktur dalam kondisi ELSE, untuk menyimpan tampilan ruang.

### **13.19 Langkah-langkah untuk memprogram tindakan tambahan**

- a. Seret struktur kontrol IF ke dalam kondisi ELSE distruktur kontrol IF yang ada dan pilih yang benar kondisi.
- b. Seret isKey variabel ke argumen yang benar.
- c. Tentukan tombol keyboard untuk mendengarkan.
- d. Tentukan pernyataan pemrograman untuk dieksekusi.

### **13.20 Contoh instruksi pemrograman lengkap**

Di bawah ini adalah contoh teko yang diprogram untuk naik dan turun menggunakan tombol keyboard B dan C.

### **13.21 Langkah-langkah untuk memindahkan objek menggunakan tombol panah**

- a. Pilih menu tarik-turun Tambah Event Listener.
- b. Pilih Keyboard.
- c. Pilih addObjectMoverFor.
- d. Pilih entitas, atau objek, untuk dikendalikan.
- e. Membuat baris kode berikut

### **13.22 Langkah-langkah untuk menguji event listener**

- a. Jalankan animasi.
- b. Klik di dalam jendela animasi dengan kurSOR Anda.

- c. Gunakan tombol keyboard (ditentukan dalam addKeyPressListener) untuk membuat objek melakukan prosedur (bergerak ke atas dan ke bawah).
- d. Gunakan tombol panah pada keyboard Anda untuk memindahkan objek maju, mundur, kanan, dan kiri.

### 13.23 Menggunakan kelas yang ada dalam animasi lain

Dapat berguna untuk mentransfer kelas dari satu animasi ke yang lainnya. Jika Anda ingin membuat beberapa fitur animasi Alice maka Anda bisa menggunakan file kelas Alice yang ada dengan semua prosedur terkait. Ini akan mengurangi pekerjaan Anda karena Anda sudah memiliki kode tindakan untuk Alice. Ini akan mengurangi pengujian Anda karena Anda sudah melakukannya menguji bahwa kode tersebut berfungsi dalam animasi asli. Anda dapat menggunakan kelas dan prosedurnya sebagai dasar untuk yang baru animasi menambahkan prosedur tambahan jika diperlukan.

### 13.24 Menyimpan file kelas di Alice 3

- a. Menggunakan tombol daftar kelas, **pilih Alice dari daftar dan kemudian klik pada kelas Alice.**
- b. Klik tombol **Simpan ke File Kelas.**
- c. Simpan file di folder MyClasses untuk akses mudah saat menambahkannya ke animasi lain dan beri nama kelas dengan nama yang sesuai. 

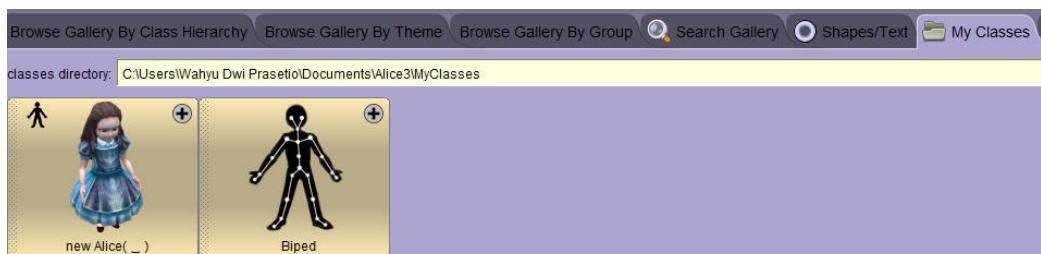
### 13.25 Menggunakan tab pemula untuk membuat dunia

- a. Alice 3 memiliki dunia pra-bangun yang dapat digunakan dengan cepat buat animasi yang lengkap dan menarik.
- b. Gunakan tab permulaan dari antarmuka proyek baru ke pilih duniamu

### 13.26 Menambahkan file kelas di Alice 3

- a. Buat proyek baru di Alice dan pergi ke editor adegan untuk mengakses galeri
- b. Pilih tab Kelas Saya dari galeri
- c. Ini memberi Anda akses ke folder MyClasses di mana kelas Anda harus diselamatkan
- d. Tambahkan kelas seperti yang Anda lakukan pada kelas lain dari galeri.
- e. Ketika Anda mengklik OK Anda **disajikan dengan daftar prosedur yang terkait dengan file kelas.**
- f. Menempatkan cursor di atas salib hijau akan memperluas kode untuk prosedur.
- g. **Mengklik selesai akan menambahkan objek Alice ke yang baru animasi.**
- h. Sekarang Anda dapat menggunakan karakter Alice dan prosedurnya dalam animasi baru Anda.
- i. Anda dapat menambahkan metode yang muncul ke yang baru sceneActivated listener sehingga Alice tampil di a cara konsisten di semua animasi Anda.

class yang sudah disimpan dapat digunakan via scene setup->myclass



## 13.27 Terminologi

Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:

- |                     |                       |
|---------------------|-----------------------|
| a. Event            | d. Kontrol keyboard   |
| b. Penanganan event | e. Pendengar keyboard |
| c. Event Listener   | f. File kelas         |

## 13.28 Ringkasan

Dalam pelajaran ini, Anda seharusnya belajar bagaimana:

- a. Membuat Urutan pembuka
- b. Menggunakan kontrol keyboard untuk memanipulasi animasi
- c. Menyimpan file Kelas Anda
- d. Menggunakan tab starter
- e. Menambahkan file kelas yang ada ke animasi

## BAB XIV

### MENGEMBANGKAN ANIMASI LENGKAP

Pelajaran ini mencakup tujuan sebagai berikut:

1. Gunakan dekomposisi fungsional untuk menulis skenario dan storyboard
2. Menyelesaikan animasi
3. Menguji animasi
4. Reposisi objek pada saat run-time
5. Upload animasi Anda
6. Rencanakan presentasi proyek animasi selesai

## 14.1 Animasi

**Animasi adalah urutan tindakan yang mensimulasikan gerakan.**

Gunakan Alice 3 untuk membangun tempat kejadian dan menulis urutan tindakan untuk animasi Anda, dan Alice 3 akan membuat animasi untuk Anda.

**“Rendering adalah proses di mana program perangkat lunak**

**mengkonversi kode Anda ke dalam animasi yang Anda lihat. Alice 3**

*merender animasi berdasarkan petunjuk yang diberikan oleh programmer.”*

## 14.2 Animasi membutuhkan perencanaan

- Animasi dapat menjadi kompleks untuk direncanakan dan dikebangkitan. Untuk mempermudah dan mengatur tugas yang kompleks ini, Anda dapat:
- Menggunakan proses metodis untuk mengidentifikasi dan memecahkan masalah yang timbul di sepanjang jalan.
  - memecahkan perkembangan menjadi langkah-langkah yang bisa dikelola.
- “Dekomposisi fungsional adalah proses metodis untuk mengidentifikasi masalah yang kompleks dan memecahnya menjadi langkah-langkah kecil yang lebih mudah untuk dikelola.”*

## 14.3 Contoh dekomposisi fungsional

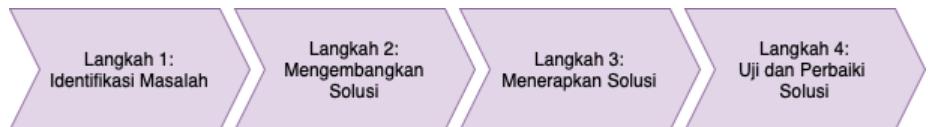
Memeriksa proses tingkat tinggi ini:

- Pertimbangkan sebuah konsep matematika yang memerlukan banyak langkah.
- Mengidentifikasi langkah-langkah tingkat tinggi untuk konsep matematika.
- Lebih lanjut memperbaiki dan menetapkan tugas-tugas tingkat rendah yang diperlukan untuk setiap langkah tingkat tinggi.

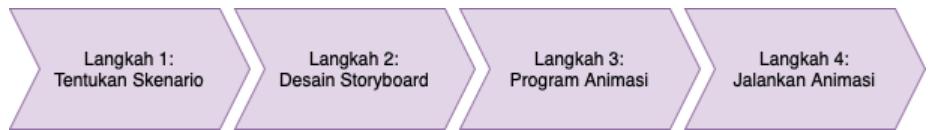
## 14.4 Proses pengembangan animasi

Proses untuk mengembangkan animasi mirip dengan proses pemecahan masalah. Bandingkan dua proses di bawah ini.

- Proses Pemecahan Masalah



- Proses pengembangan animasi



### **Langkah 1: Tentukan skenario**

- a. Animator profesional memulai dengan **mengembangkan skenario** — atau **cerita** — yang memberi animasi tujuan. Contoh:
- 1) Sebuah cerita yang menghadirkan konflik dan resolusi.
  - 2) Sebuah pelajaran yang mengajarkan konsep matematika.
  - 3) Sebuah simulasi yang menunjukkan proses.
  - 4) Sebuah game yang menghibur atau melatih.

Tabel 14.1 Contoh Skenario dan Animasi

Skenario Jenis	Skenario	animasi
Masalah dan solusi.	Seekor kucing <b>membutuhkan bantuan</b> untuk turun dari pohon.	Seorang petugas <b>pemadam kebakaran</b> <b>manjat pohon</b> untuk <b>menyelamatkan kucing</b> itu.
Mengajarkan konsep.	<b>Menghafal simbol kimia</b> sulit.	Sebuah <b>permainan</b> waktuunya <b>mencocokkan simbol kimia</b> dengan <b>definisi</b> mereka.
Simulasikan atau menunjukkan suatu proses.	Sebuah mobil memiliki <b>ban kempes</b> .	Sebuah cara <b>demonstrasi</b> <b>bagaimana</b> mengganti ban pada mobil virtual.

Bermain permainan.	sebuah Pesawat terbang harus menghindari benda di jalurnya saat terbang melintasi langit.	Sebuah permainan manuver pesawat interaktif di sekitar benda-benda di langit.
--------------------	---	---

### Langkah 2: Desain storyboard

Dua jenis storyboard sering digunakan untuk merencanakan animasi:

- Visual:** Serangkaian gambar ilustrasi yang mewakili adegan utama animasi.
- Tekstual:** Daftar tindakan yang terperinci dan terurut yang dilakukan setiap objek dalam setiap adegan animasi.

#### 14.5 Format storyboard

Kembangkan storyboard visual dan tekstual Anda menggunakan berbagai format.

Contoh:

- Gambar mereka di atas kertas dengan pensil.
- Buat mereka menggunakan alat digital seperti pengolah kata, perangkat lunak menggambar, atau program presentasi.
- Tulis storyboard tekstual dalam Alice 3 di editor Kode menggunakan komentar untuk mengatur langkah-langkah dalam program Anda.

#### 14.6 Storyboard visual

Storyboard visual membantu pembaca memahami:

- Komponen adegan.
- Bagaimana adegan awal akan diatur.
- Objek bergerak dan tidak bergerak dalam sebuah adegan.
- Tindakan yang akan terjadi.
- Interaksi pengguna yang terjadi selama eksekusi animasi.

## Contoh storyboard visual



Gambar 14.2 Storyboard

Laki-laki dan perempuan duduk di bangku taman. laki-laki itu berjalan pergi, meninggalkan teleponnya..

perempuan tersebut melihat telefon. Dia **berpikir**, "Hei! Laki-laki itu lupa membawa teleponnya!"

perempuan terserbut berteriak, "Hey! Kamu lupa membawa telefon mu!" laki-laki itu berbalik dan berjalan kembali ke bangku. Dia mengatakan, "Oh! Terima kasih!"

## 14.7 Storyboard tekstual

Sebuah **storyboard tekstual** membantu pembaca memahami tindakan yang akan terjadi selama animasi. Objek bergerak dan tidak bergerak dapat dengan mudah diidentifikasi dalam pernyataan tindakan, tetapi **deskripsi** yang lebih rinci mungkin diperlukan jika beberapa programmer terlibat dalam menerapkan setiap adegan.

*"Dalam komputasi, sebuah storyboard tekstual disebut algoritma: daftar tindakan untuk melakukan tugas atau memecahkan masalah."*

### Contoh 1 Storyboard tekstual

Program tindakan berikut secara berurutan:

- a. Laki-laki dan perempuan duduk di bangku taman.

- b. laki-laki itu berdiri dan berjalan pergi, meninggalkan teleponnya di bangku taman.
- c. perempuan tersebut berbalik untuk melihat telepon.
- d. perempuan itu berpikir, "Hei! laki-laki itu lupa teleponnya!"
- e. perempuan tersebut berteriak, "Hei! Kamu lupa teleponmu!"
- f. laki-laki itu berhenti dan berbalik.
- g. laki-laki tersebut berjalan kembali ke bangku taman dan berkata, "Oh!  
Terima kasih!"

### Contoh 2 Storyboard tekstual

Contoh ini menunjukkan bagaimana Anda dapat mengembangkan storyboard Anda dengan terlebih dahulu **menulis komentar di Code editor program Anda**. Setelah itu, **Anda dapat mulai untuk mengembangkan animasi langsung dari storyboard**.



```

Scene initializeEventListeners myFirstMethod
declare procedure [myFirstMethod v]
do in order
  [// Boy and girl sit on a park bench.]
  [// Boy stands up and walks away, leaving his mobile phone on the park bench.]
  [// Girl turns to look at the phone.]
  [// Girl thinks "Hey! You forgot your phone!"]
  [// Boy stops and turns around.]
  [// Boy walks back to the park bench and says, "Oh! Thank you!"]
  [// End of program.]
end
end

```

Gambar 14.3 Contoh stsoryboard Tekstual

### 14.8 Komponen storyboard

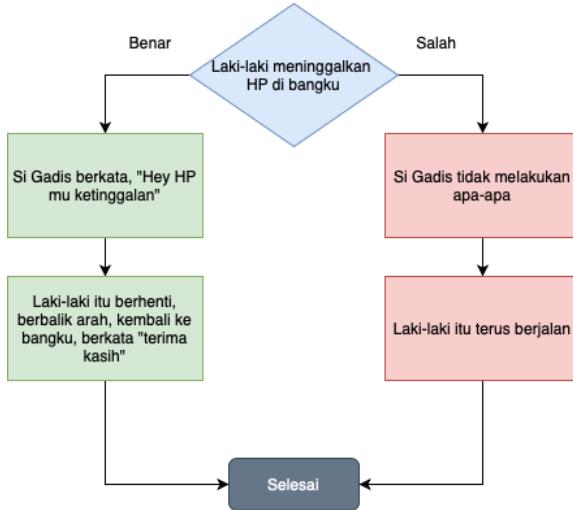
Komponen storyboard tekstual:

Tabel 14.2 Komponen Storyboard

Komponen	Definisi	Contoh
Tempat kejadian	Tempat (atau "dunia" di Alice 3) di mana cerita Anda terjadi.	Taman, perpustakaan, sekolah, rumah
benda	Karakter bergerak atau tidak bergerak yang Anda program untuk bergerak dan bertindak.	Hewan, mobil, orang, pohon-pohon
tindakan	Instruksi untuk bagaimana setiap objek harus bertindak dalam adegan.	Berjalan 2 meter, belok kiri, mengatakan "Hello!"
interaksi pengguna	Cara di mana pengguna melihat animasi dapat memanipulasi objek dalam animasi.	perintah keyboard atau klik mouse untuk membuat objek bergerak
Rancangan Spesifikasi	Bagaimana benda dan pemandangan harus terlihat dalam animasi.	Ukuran, posisi, lokasi, warna

#### 14.9 Alur proses storyboard

Diagram alur storyboard membantu Anda mengatur aliran tindakan dan kondisi dalam sebuah animasi.



#### 14.10 Bagaimana storyboard bermanfaat

Ada beberapa cara yang membantu storyboard dalam pengembangan program:

- Storyboard teknstual** dapat digunakan untuk **menghasilkan pernyataan komentar** Program dan **mengatur pengembangan program.**
- Storyboard** juga dapat **membantu programmer mengidentifikasi tindakan berulang-ulang**, atau tindakan yang mungkin dilakukan oleh lebih dari satu objek.

#### Langkah 3: Program animasi

- Setelah menyelesaikan storyboard, langkah selanjutnya adalah **memprogram animasi di Alice3.**
- Saat Anda memprogram animasi, lihat storyboard Anda untuk spesifikasi desain animasi.
- Kode yang ditulis dalam Alice 3 memberikan instruksi untuk tampilan animasi saat run-time.

## 14.11 Animasi checklist

Selama proses pengembangan animasi, menggunakan checklist ini untuk memastikan bahwa animasi Anda memenuhi semua prinsip-prinsip animasi.

### Checklist for Animation Completion

Area	Check	
Scenario	Did you clearly define the scenario?	
Storyboard	Did you think-through the animation by creating a storyboard?	
Textual storyboard	Did you think-through the programming code by creating a textual storyboard?	
Program	Did you complete the programming of the animation?	

Gambar 14.4 *Animasi Checklist*

## 14.12 Buat animasi lengkap

Membuat animasi lengkap membutuhkan pemahaman yang menyeluruh dan penerapan semua komponen yang telah Anda pelajari sejauh ini.

Tabel 14.3 Komponen dan fungsi dalam animasi

Adegan dari beberapa objek dari banyak kelas, termasuk properti dan bentuk	Kendaraan berkuda dengan prosedur setVehicle
Prosedur deklarasi	Fungsi
Prosedur pergerakan	IF dan struktur kontrol WHILE
Rotasi objek dan prosedur rotasi sub-bagian objek.	Angka acak

Gerakan simultan dengan struktur kontrol Do Together	Ekspresi matematika
Variabel	Kontrol keyboard

#### **Langkah 4: Jalankan animasi**

- Jalankan animasi untuk menguji apakah ia berfungsi dengan baik dan jalankan tindakan yang direncanakan di storyboard.
- Proses ini sering disebut sebagai pengujian dan debugging software.

*"Program dapat diuji dengan memasukkan nilai yang tidak diinginkan dalam bidang argumen metode dalam upaya untuk mencoba dan "memecahkan" kode. Ketika ada sesuatu yang rusak atau tidak berfungsi seperti yang Anda maksudkan dalam program perangkat lunak, itu sering disebut sebagai "bug". Debugging adalah proses menemukan dan menghilangkan bug dalam program perangkat lunak."*

#### **14.13 Tugas debugging**

- Tes dan debug animasi sesering yang sedang dikembangkan.
- Gunakan beberapa teknik debugging berikut:
- Sesuaikan arah, jarak, dan durasi objek bergerak.
- Sesuaikan perhitungan matematika yang membantu mempersempit jarak atau durasi benda bergerak.
- Saring instruksi dalam kode yang tidak berfungsi sebagaimana dimaksud.
- Atasi kesalahan yang dibuat oleh programmer.

#### **14.14 Uji elemen animasi anda**

- Menguji setiap elemen untuk membuktikan bahwa ia bekerja tanpa kesalahan.
- Ekspresi matematika menghitung seperti yang diharapkan.
- Benda bergerak dengan waktu halus.
- Struktur kontrol beroperasi seperti yang diharapkan.

e. Event Listener memicu respon yang benar.

*“Pastikan bahwa kode Anda mencakup komentar yang dengan jelas mengidentifikasi tujuan atau fungsionalitas blok pernyataan dalam program Anda. Mengacu pada komentar akan membantu Anda menguji dan men-debug program Anda.”*

#### 14.15 Input pengguna untuk mengubah posisi objek saat di run-time

- Untuk membuat program interaktif, objek Listener harus ditambahkan ke adegan.
- Prosedur `addDefaultModelManipulation` membuat objek Listener yang menargetkan klik mouse pada objek apa pun dalam adegan dan merespons dengan memungkinkan pengguna untuk menyeret objek itu di sekitar adegan saat animasi sedang berjalan.



#### 14.16 Langkah-langkah untuk menambahkan prosedur add default model manipulation

##### manipulation

- Buka tab Scene.
- Klik tombol edit di sebelah initializeEventListeners.
- Tarik prosedur `addDefaultModelManipulation` (Scene kelas) ke code editor initializeEventListeners.



7

Gambar 14.5 addDefaultModelManipulation

#### 14.17 Menggunakan prosedur add default model manipulation

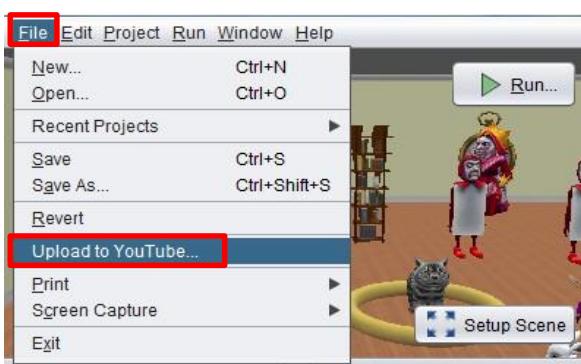
Prosedur ini memungkinkan Anda untuk memposisikan ulang objek saat dijalankan:

- Klik dan seret objek dengan kursor Anda untuk memindahkannya di sekitar tempat kejadian.
- Tekan tombol Control (Ctrl), lalu klik dan drag objek dengan kursor Anda untuk belok kanan dan kiri.
- Tekan tombol Shift, lalu klik dan seret objek untuk memindahkannya ke atas dan ke bawah.

#### 14.18 Meng-upload animasi anda

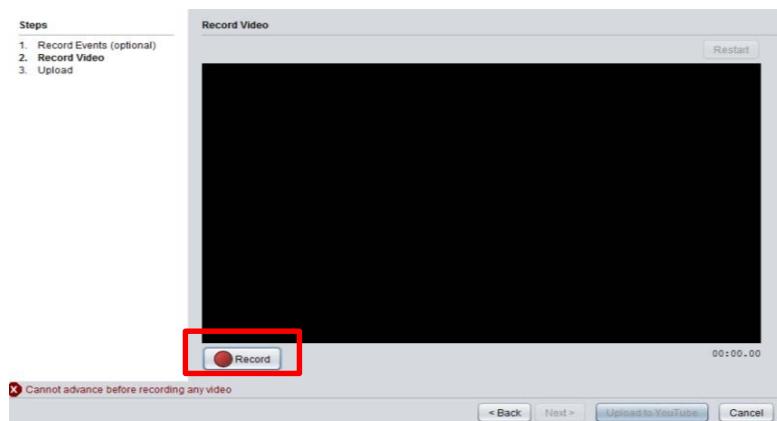
Setelah Anda menyelesaikan animasi Anda, Alice memiliki fasilitas untuk memungkinkan Anda untuk meng-upload file Anda langsung ke YouTube, yang Anda butuhkan adalah akun YouTube dan Anda dapat meng-upload langsung dari dalam Alice 3

Untuk mengakses fasilitas pergi ke File, Upload ke YouTube.



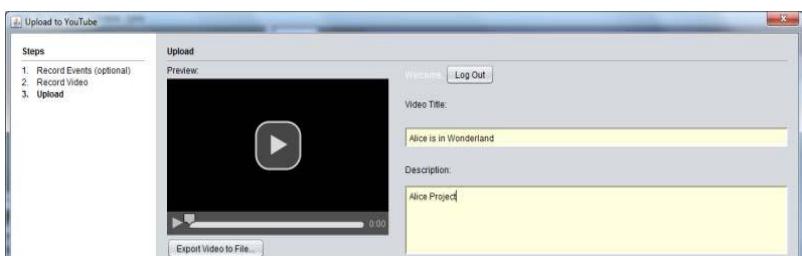
Gambar 14.6 Langkah Upload Animasi ke Youtube

- a. Antarmuka mengharuskan Anda untuk merekam animasi Anda ke dalam format (.webm) yang dapat di-upload ke YouTube.
- b. Untuk melakukan hal ini tekan tombol rekam.



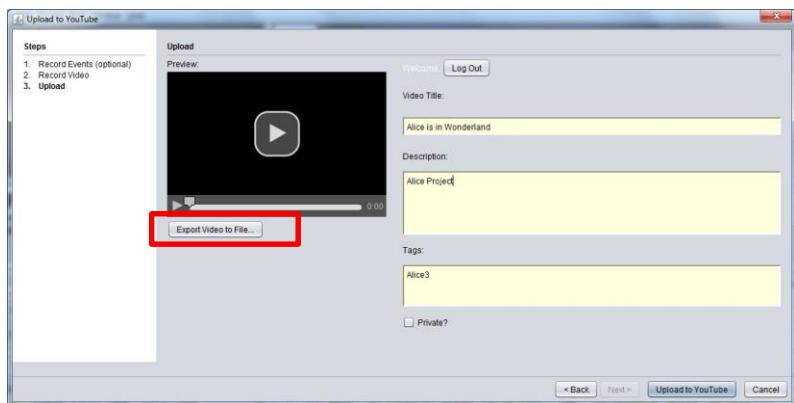
Gambar 14.7 Tampilan Record layar

- c. Tekan tombol stop ketika animasi Anda selesai atau Anda telah merekam bagian yang ingin Anda upload.
- d. Anda dapat meninjau video Anda sebelum masuk ke YouTube dengan rincian akun Anda dan menambahkan judul, deskripsi dan tag tambahan yang Anda inginkan.
- e. Kemudian klik upload ke tombol YouTube.



Gambar 14.8 Tampilan akhir record

- f. Jika Anda tidak ingin mengunggah video tetapi **Anda ingin membuat salinan file lokal maka klik tombol Eksport Video to File yang akan memungkinkan Anda menyimpan file secara lokal.**



Gambar 14.9 Eksport Video to File

#### 14.19 Presentasikan proyek animasi anda

Saatnya untuk mempresentasikan proyek animasi lengkap Anda. Berikut adalah beberapa **langkah yang harus diikuti ketika mengatur presentasi proyek animasi Anda:**

- a. Pastikan presentasi Anda diuji secara menyeluruh dan lengkap.

- b. Rencanakan untuk menunjukkan bagaimana Anda menggunakan setiap konsep yang dipelajari dalam kursus ini.
- c. Cari tahu berapa banyak waktu yang Anda miliki untuk presentasi Anda.
- d. Jika presentasi kelompok, rencanakan siapa yang akan melakukan bagian presentasi yang mana.

**1) Siapkan garis besar presentasi**

- a. Buat garis besar presentasi untuk merencanakan presentasi proyek animasi lengkap Anda.
- b. Slide berikut menunjukkan contoh garis besar presentasi.

**2) Garis besar presentasi : bagian 1**

**Bagian 1: Pendahuluan**

- a. Dapatkan perhatian pendengar Anda.
- b. Perkenalkan **tema proyek animasi**.
- c. **Pratinjau animasi Anda**.
- d. Tentukan kredibilitas Anda sebagai pembicara.
- e. Berikan alasan kepada audiens untuk mendengarkan presentasi

**3) Garis besar presentasi: bagian 2**

**Bagian 2: Presentasi proyek animasi**

- a. Atur presentasi dalam alur yang logis.
- b. Tampilkan semua kemampuan proyek animasi Anda.
- c. **Demonstrasikan** bagaimana setiap konsep kursus digunakan dalam **proyek animasi**.
- d. Alat bantu waktu untuk mendukung presentasi yang diucapkan.
- e. Sertakan transisi yang bermanfaat di antara gagasan.

**4) Garis besar presentasi: bagian 3**

**Bagian 3: Kesimpulan**

- a. **Ringkas animasi** dengan cara yang mudah diingat.
- b. Memotivasi audiens untuk merespons.

- c. Berikan penutupan.

**5) Berlatih presentasi anda**

Ketika berlatih presentasi Anda:

- a. Berlatih keras.
- b. Mengatur waktu bicara Anda; jika terlalu panjang atau pendek, merevisinya.
- c. Berlatih berdiri.
- d. Berlatih di depan seseorang.
- e. Rekam atau video pidato Anda. Jika Anda tidak memiliki akses ke peralatan rekaman, berlatih di depan cermin.
- f. Berlatih menggunakan alat bantu visual dan teknologi.

**6) Presentasi kelompok**

- a. Jika Anda akan membuat presentasi Anda dengan kelompok, berikut adalah beberapa langkah kelompok Anda dapat mengikuti untuk memastikan semua anggota kelompok memiliki bagian dalam menyiapkan dan menyampaikan presentasi.
- b. Masukkan semua anggota kelompok dalam semua langkah perencanaan presentasi.
- c. Bagilah tugas presentasi secara merata di antara anggota kelompok.
- d. Berlatih presentasi sebagai sebuah kelompok, dan saling memberikan umpan balik yang konstruktif.

**14.20 Terminologi**

Istilah kunci yang digunakan dalam pelajaran ini termasuk:

- a. Algoritma
- b. Animasi checklist
- c. Debugging
- d. Komentar
- e. Dekomposisi fungsional

- f. Rendering
- g. Skenario
- h. Storyboard textual
- i. Storyboard Visual

#### **14.21 Ringkasan**

Dalam pelajaran ini, Anda harus belajar bagaimana untuk:

- a. Gunakan dekomposisi fungsional untuk menulis **skenario** dan storyboard
- b. Menyelesaikan animasi
- c. Menguji animasi
- d. Ubah posisi objek pada saat run-time
- e. Upload animasi Anda
- f. Rencanakan presentasi proyek animasi yang lengkap.

## **BAB XV**

### **VARIABEL DAN TIPE DATA PADA JAVA**

Pelajaran ini mencakupi beberapa tujuan-tujuan sebagai berikut:

1. Menjelaskan variabel
2. Menjelaskan tipe-tipe sederhana dari java
3. Mendefinisikan operator aritmatika
4. Menjelaskan operator relasional dan operator logika
5. Menjelaskan operator penugasan

#### **15.1 Perbedaan Alice 3 dengan Java**

Tabel 15. 1 Perbedaan Alice dan Java

Alice 3	Java
Konsep dari Alice 3 berupra 3 Dimensi menggunakan representasi visual	Bahasa pemrograman: sintaks dapat diedit menggunakan IDE
Digunakan untuk membuat sebuah animasi atau game dengan menggunakan konstruksi pemrograman	Digunakan untuk membuat aplikasi tersebut bisa dijalankan di platform apa pun, termasuk web dengan menggunakan sintaks java
Tarik dan lepas interface yang sudah didesain untuk mengurangi kesalahan pada sintax dan membuat belajar pemrograman menjadi lebih mudah	IDE membantu kita untuk memodelkan objek secara nyata. Memungkinkan digunakan kembali dan pemeliharaan yang mudah

#### **15.2 Variabel pada Alice 3**

- a. Variabel dideklarasikan sesuai dengan keinginan kita

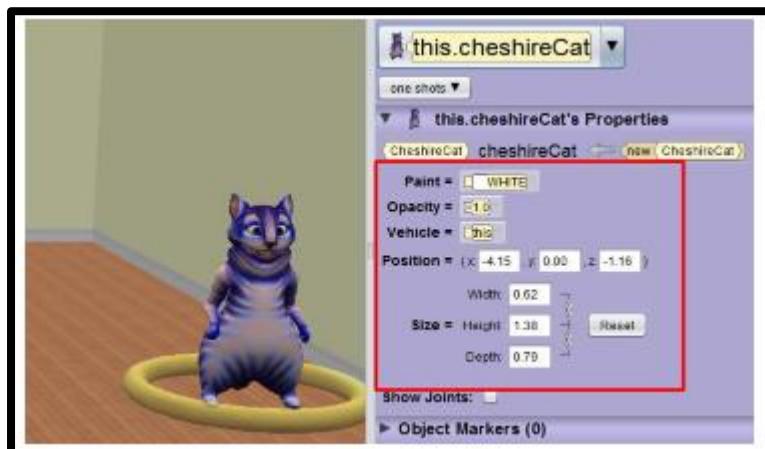
- b. Aplikasi ini menggunakan berbagai nilai yang terus berubah saat program sedang dijalankan
- c. Contohnya, dalam Alice 3 sebuah mobil diprogramkan untuk berguling berapa kali.
- d. Nilai yang dimasukkan oleh suatu pengguna mungkin berbeda-beda sesuai keinginan masing-masing

### 15.3 Mengatur variabel nilai

Mengatur variabel nilai yang dimasukan oleh pengguna yang berbeda-beda di Alice 3. Variabel adalah di mana tipe data tertentu dapat disimpan dan nantinya akan dipanggil untuk digunakan oleh program tersebut. Setiap variabel diberi nama yang uni agak mudah untuk ditemukan.

### 15.4 Properti objek

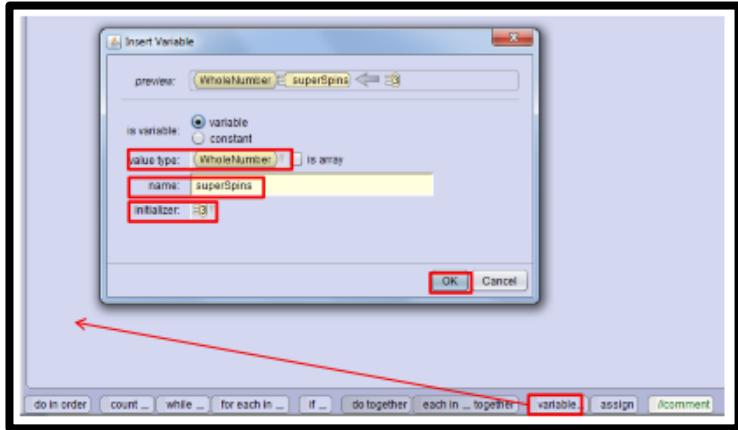
Properti objek adalah variabel yang menyimpan informasi tentang objek seperti warna, tinggi dan lebarnya.



Gambar 15.1 Properti Objek

### 15.5 Mendeklarasikan variabel dalam Alice 3

Tarik kotak variabel dan ke editor kode untuk mendeklarasikan a sebagai variabel baru.



Gambar 15.2 Deklarasi Variabel dalam Alice

### 15.6 Deklarasi variabel di Java

Di Java, kita mendeklarasikan suatu variabel kemudian kita dapat memanggil variabel tersebut di program ini.

```

public class Print{
    public static void main(String[] args){
        //variable declaration section
        int i=1;
        int j;

        while(i<=7){
            for(j=1; j<=i; j++)
                System.out.println("*");
            i=i+2;
            System.out.println();
        }
    }
}

```

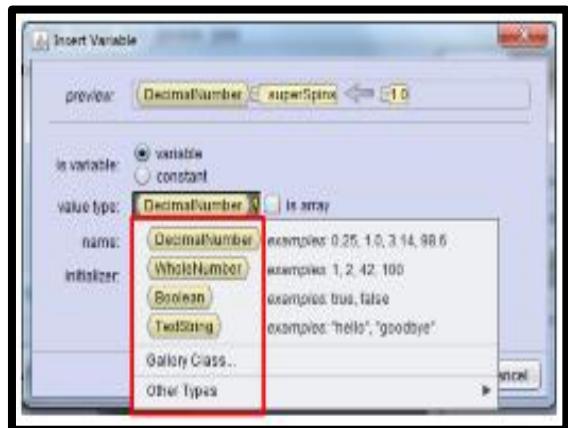
i and j are variables that are declared here.

i and j are called in the program body.

Gambar 15. 3 Deklarasi Variabel Java

### 15.7 Tipe data di Alice 3

Ketika kita ingin mendeklarasikan suatu variabel di Alice. Kita harus mendefinisikan tipe data. Suatu variabel tipe data mendefinisikan jenis-jenis nilai dan variabel dapat disimpan.



Gambar 15.4 Tipe Data di Alice

### 15.8 Variabel tipe data di Alice 3

Tabel 15.2 Variabel Tipe Data di Alice

Tipe Data	Deskripsi
Angka decimal	Menampilkan sebuah aritmatika dan mengatur nilai dari argumen Contoh: 0.1, 2.25, 98.6
Angka bulat	Menampilkan sebuah arimatika dan mengatur nilai dari argument Contoh: 1,459,30
boolean	Satu dari 2 nilai yaitu: benar atau salah Biasanya merupakan hasil tes yang membandingkan satu hal dengan hal lainnya
objek	Banyak objek di Alice 3 seperti kucing, anjing, orang, dan lain-lain
Kelas	Kelas objek dalam animasi yang kita buat Contoh: biped, scene, quadruped

### Teks String

String merupakan suatu variabel untuk mencetak huruf seperti “Halo”, “Selamat tinggal” dan lain-lain

## 15.9 Contoh tipe data di Java

```
class PrintText{  
    public static void main(String[] args){  
        //variable declaration section  
        byte aByte = -10;  
        int aNumber = 10;  
        char aChar = 'b';  
        boolean isBoolean = true;  
  
        //print variables alone  
        System.out.println(aByte);  
        System.out.println(aNumber);  
        //print variables with text  
        System.out.println("aChar = " + aChar);  
        System.out.println("The boolean var is:" + isBoolean);  
    } //end method main  
} //end class PrintText
```

This code defines variables using four different Java types.

Variables are declared and initialized.

Variables are printed.

Gambar 15.5 Tipe Data di Alice

## 15.10 Tipe data dasar pada Java

Tabel 15.3 Tipe Data Dasar pada Java

Tipe Data	ukuran	Contoh data	Dekripsi data
boolean	1 bit	Benar, salah	Menyimpan nilai benar atau salah
byte	1 byte (8bits)	12,128	Menyimpan bilangan bulat dari -127 sampai 128
char	2 bytes	'A' '5' '#'	Menyimpan karakter kode tunggal
short	2 bytes	6, -14, 2345	Menyimpan bilangan bulat dari -32,768 sampai 32,767

<b>int</b>	<b>4 bytes</b>	6, -14,2345	Menyimpan bilangan bulat dari $-2,147,483,648$ sampai $2,147,483,647$
<b>long</b>	<b>8 bytes</b>	3459111,2	Menyimpan bialangan bulat dari - $9,223,372,036,854,775,808$ to $9,223,372,036,854,775,807$
<b>Float</b>	<b>4 bytes</b>	3.145, .077	Menyimpan nilai decimal positif atau negative dari $1.403 \times 10^{-45}$ sampai $3.4028 \times 10^{38}$
<b>double</b>	<b>8 bytes</b>	0.0000456, 3.7	Menyimpan nilai decimal positif atau negative dari $4.9406 \times 10^{-324}$ sampai $1.7977 \times 10^{308}$

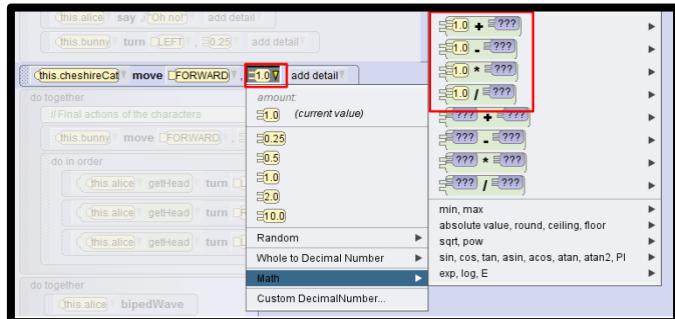
### 15.11 Operasi aritmatika di Alice 3

Menggunakan tanda tambah (+), kurang (-), Kali (\*) dan bagi (/) untuk membuat:

- Jumlah dan durasi argumen
- Fungsi jarak
- Nilai dari operasi arimatika dapat disimpan dalam variabel a
- Operasi Aritmatika melakukan operasi matematika dasar. contoh halnya menggunakan 2 -operasi matematika dan jalankan maka hasilnya sama dengan perhitungan matematika.

### 15.12 Operator aritmatika dalam argument jarak

Mengguanakan panah ke bawah di sebelah nilai argumen memberikan kamu akses menggunakan fungsi matematika.



Gambar 15. 6 Operator Aritmatika dalam argumen Jarak

### 15.13 Operasi aritmatika pada Java

Operator aritmatika memiliki kerja yang sama pada Alice dan java, tetapi cara mengaksesnya berbeda. Pada java, anda memasukan operator aritmatika langsung pada kode anda,

```
class BasicOperators {
    //using arithmetic operators
    public static void main(String[] args) {
        //variable declaration section
        int a = 1 + 1;
        int b = 3 * 3;
        int c = 1 + 8 / 4;
        int d = -2;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("d = " + d);
    }//end method main
}//end class BasicOperators
```

Gambar 15. 7 Operasi Aritmatika pada Java

### 15.14 Operator relasional

Operator operasional meliputi: = ≠ > < ≥ ≤

**↳**

Ekspresi dengan operator relasional menghasilkan nilai benar atau salah. Sebuah operator relasional adalah unit leksikal yang digunakan untuk mengekspresikan hubungan, seperti kesetaraan atau lebih besar dari, antara dua ekspresi. Dua ekspresi cocok dikombinasikan dengan operator relasional sering membentuk ekspresi relasional atau kondisi dalam bahasa pemrograman.

## 15.15 Operator relasional di Alice 3

Contoh berikut menunjukkan operator relasional digunakan untuk menguji jarak antara kucing Cheshire dan Dalmatian. Jika jarak dari kucing ke anjing kurang dari kedalaman anjing, kucing mengatakan "Hello anjing!". Jika jarak tidak kurang dari kucing bergerak maju 0,25 meter.



Gambar 15.8 Operator relasional di Alice

## 15.16 Operator relasi di Java

Tabel 15.4 Operator relasi di Java

Simbol	Nama Operator	Contoh
$==$	Sama dengan	$(A == B)$ adalah salah
$!=$	Tidak sama dengan	$(A != B)$ adalah benar
$>$	Lebih Besar dari	$(A > B)$ adalah salah
$\geq$	Lebih besar dari atau sama dengan	$(A \geq B)$ adalah salah
$<$	Kurang dari	$(A < B)$ adalah benar
$\leq$	Kurang dari atau sama dengan	$(A \leq B)$ adalah benar

## 15.17 Operator relational dalam contoh Java

Operator relasional di Jawa dan Alice bekerja dengan cara yang sama.

Di bawah ini adalah contoh kode Java yang terdapat operator relasional.

```
class Test {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 20;  
  
        System.out.println("a == b = " + (a == b));  
        System.out.println("a != b = " + (a != b));  
        System.out.println("a > b = " + (a > b));  
        System.out.println("a < b = " + (a < b));  
        System.out.println("b >= a = " + (b >= a));  
        System.out.println("b <= a = " + (b <= a));  
    } //end method main  
} //end class Test
```

Gambar 15.9 Contoh Operator Relational di Java

## 15.18 Operator logika dalam Alice 3

Operator logika adalah operator boolean **AND**, **OR** dan **NOT**. Ekspresi ditulis dengan operator logika menghasilkan benar atau salah.



Gambar 15.10 Operator Logika dalam Alice

## 15.19 Contoh operator logika dalam Alice 3

Contoh penggunaan a dan b operator logika untuk menggabungkan dua kondisi ke kondisi tunggal. Kedua kondisi ini harus benar untuk keseluruhan kondisi agar menjadi kenyataan. Jika tidak, itu akan menjadi salah.



Gambar 15.11 Contoh Operator Logika dalam Alice

### 15.20 Operator logika di Java

- Operator logika dalam Alice 3 dan Java sama, hanya cara aksesnya yang berbeda.
- Di Java, operator logika diketik menggunakan simbol-simbol.

Tabel 15.5 Operator Logika di Java

Simbol	Nama Operator
&&	"Operator Short-hubung" Conditional <b>DAN</b> atau
	"Operator Short-hubung" Conditional <b>OR</b> atau
!	<b>TIDAK</b>

## 15.21 Contoh operator logika di java

```
class BoolNotDemo {  
    public static void main(String[] args){  
        //mendeklarasikan variabel yang  
        //dipilih  
        int x = 2;  
        int y = 1;  
        boolean bl;  
        bl = !(x > y); // bl salah  
        System.out.println("x is not greater  
        than y:"+bl);  
        bl = !(y > x); // bl benar  
        System.out.println("y is not greater  
        than x:  
        "+bl); // akhir method utama  
    } //akhir kelas
```

Gambar 15.12 Operator Logika di Java

## 15.22 Operator tugas di Alice 3

- Operator penugasan mengubah nilai variabel.
- Dalam Alice 3 antarmuka, Anda menempatkan nilai ke properti dan variabel.
- Warna, opacity, posisi, kendaraan dan ukuran sifat ditugaskan nilai-nilai di Objek Properties.



Gambar 15.13 Operator Tugas di Alice

- d. Dalam contoh ini, variabel superSpins lokal ditugaskan nilai awal dari tiga.
- e. Jumlah kali spin karakter ditugaskan untuk superSpins.

### 15.23 Operator penugasan di java

Di Java, gunakan tanda sama dengan ("=") untuk menetapkan satu nilai ke nilai lain. Dalam contoh ini, nilai y adalah 5 dan nilai z adalah 25

```
class
AssignmentDemo{
public static void main(String[] args) {
// bagian deklarasi variabel
    int x=5;
    int y=10;
    int z=20;
    y = x;
    z = y + z;
    System.out.println("T
        he value of y is:"+ y);
    System.out.println("T
        . . . . .
```

Gambar 15.14 Operator Penugasan di Java

### 15.24 Menangani operasi standar di Java

Java menyediakan sintaks khusus untuk menangani operasi standar seperti  $z = y + z;$

Sintaksnya adalah  $z += y;$

Kode ini menyimpan beberapa penekanan tombol, tetapi masih memberikan z nilai y ditambah z.

### 15.25 Sintaks penugasan untuk operasi lain

Tabel 15.6 Sintaks Penugasan

Simbol	Contoh	Setara dengan
$+=$	$x += y$	$x = x + y;$

- =	x -= y	x = x - y;
* =	x *= y	x = x * y;
/ =	x /= y	x = x / y;

### 15.26 Contoh kode sintaks penugasan

```
class AssignmentDemo2{
    public static void main(String[] args) {
        // bagian deklarasi variabel
        int x=5;
        int y=10;
        x += y;
        System.out.println("The += result is:" + x);
        x -= y;
        System.out.println("The -= result is:" + x);
        x *= y;
        System.out.println("The *= result is:" + x);
        x /= y;
        System.out.println("The /= result is:" + x);
    } //Akhir dari main method
} //Akhir dari AssignmentDemo2
```

Gambar 15.15 Kode Sintaks Penugasan

### 15.27 Terminologi

Istilah-istilah kunci yang digunakan dalam pelajaran ini termasuk:

- Operator aritmatika
- Operator penugasan
- Tipe data
- Operator logika
- Operator relasional
- Variabel

### 15.28 Ringkasan

Dalam pelajaran ini, kamu seharusnya mempelajari bagaimana:

- a. Menjelaskan variabel
- b. Menjelaskan tipe-tipe sederhana Java
- c. Menentukan operator aritmatika
- d. Menjelaskan operator relasional dan logika
- e. Menjelaskan operator penugasan