BAB 2

LANDASAN TEORI

2.1 Rumah Baca Buku Sunda

Rumah Baca Buku Sunda atau yang disingkat RBBS mulai dibentuk sekitar Februari 2004. Awalnya RBBS merupakan perpustakaan pribadi milik Mamat Sasmita. Sebelumnya, Mamat Sasmita atau yang akrab dipanggil Uwak Sasmita merasa koleksi bukunya sudah semakin banyak dan rasanya sayang bila hanya dibaca sendiri. Ia pun memutuskan untuk menjadikan perpustakaan pribadinya sebagai rumah baca buku untuk umum. Koleksi buku di RBBS berjumlah sekitar 8000 buah dan 3000 buah diantaranya adalah karya sastra Sunda seperti novel, kumpulan cerita pendek, kumpulan esai, kumpulan puisi dan sajak, dan ada juga yang membahas tentang kebudayaan Sunda. Koleksi buku Sunda di RBBS tidak hanya buku-buku tentang sastra dan kebudayaan Sunda yang berbahasa Sunda saja. Ada juga buku-buku yang membahas hal sama tetapi disajikan dalam bahasa Indonesia atau bahasa lain seperti bahasa Inggris. Selain itu, ada pula buku-buku umum yang merupakan sastra Indonesia, sastra berbahasa asing seperti bahasa Inggris, dan sastra terjemahan. Buku-buku yang ada biasanya dibeli dari para penjual buku loak di Bandung seperti yang ada di daerah Palasari, Dewi Sartika, Cikapundung, Cibeunying, dan Jatayu. Selain membeli sendiri, koleksi buku juga didapatkan dari sumbangan pihak luar dan biasanya buku-buku bertemakan sastra serta kebudayaan.

Beberapa koleksi buku Sunda di RBBS:

- Kamoes Basa Sunda karangan R. Satjadibrata yang diterbitkan Bale Poestaka pada tahun 1948.
    - Kamus Sunda-Inggris yang disusun Jonathan Rigg, (anggota The Batavia Society of Arts and Sciences) dan diterbitkan Batavia Lange & Co pada tahun 1862.
    - Buku puisi Sawer Bahasa Sunda yang diterbitkan Departemen Pendidikan dan Kebudayaan.
    - Novel Sunda Sebelum Perang karangan Yus Rusyana.
    - Dipati Ukur Djilid 4 yang diterbitkan Daja Sunda Pusat.
    - Jagad Carita (Kandaga Carpon Dunya) karangan Hawé Setiawan.
    - Pasini Jangji di Muaraberes (Carita Mundinglaya) karangan Rohmat Tasdik Al Garuti.

2.2 Landasan Teori

2.2.1. String Matching

String dalam ilmu komputer dapat diartikan dengan sekuens dari karakter. Walaupun sering juga dianggap sebagai data abstrak yang menyimpan sekuens nilai data, atau biasanya berupa *bytes* yang mana merupakan elemen yang digunakan sebagai pembentuk karakter sesuai dengan *encoding* karakter yang disepakati seperti ASCII, ataupun EBCDIC [**6**]. Pencocokan *string* atau *string matching* adalah proses pencarian semua kemunculan *string* pendek *P[0..n-1]* yang disebut *pattern* di *string* yang lebih panjang *T[0..m-1]* yang disebut teks. Pencocokan *string* merupakan permasalahan paling sederhana dari semua permasalahan *string* lainnya, dan merupakan bagian dari pemrosesan data,

pengkompresian data, *lexical analysis*, dan temu balik informasi. Teknik untuk menyelesaikan permasalahan pencocokan *string* biasanya akan menghasilkan implikasi langsung ke aplikasi *string* lainnya [**7**].

2.2.2. ASCII (American Standard Code for Information Interchange)

Kode Standar Amerika untuk Pertukaran Informasi atau ASCII (American Standard Code for Information Interchange) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Ia selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan biner sebanyak 8 bit. Dimulai dari 0000 0000 hingga 1111 1111. Total kombinasi yang dihasilkan sebanyak 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan Desimal**.**

Tabel 2-1 ASCII CODE

No	Decimal	Hex	Binary	Value
1	32	20	100000	(space)
2	33	21	100001	!
3	34	22	100010	"
4	35	23	100011	#
5	36	24	100100	$
6	37	25	100101	%
7	38	26	100110	&
8	39	27	100111	'
9	40	28	101000	(
10	41	29	101001	)

No	Decimal	Hex	Binary	Value
11	42	02A	101010	*
12	43	02B	101011	+
13	44	02C	101100	,
14	45	02D	101101	-
15	46	02E	101110	.
16	47	02F	101111	/
17	48	30	110000	0
18	49	31	110001	1
19	50	32	110010	2
20	51	33	110011	3
21	52	34	110100	4
22	53	35	110101	5
23	54	36	110110	6
24	55	37	110111	7
25	56	38	111000	8
26	57	39	111001	9
27	58	03A	111010	:
28	59	03B	111011	;
29	60	03C	111100	<
30	61	03D	111101	=
31	62	03E	111110	>
32	63	03F	111111	?
33	64	40	1000000	@
34	65	41	1000001	A
35	66	42	1000010	B

No	Decimal	Hex	Binary	Value
36	67	43	1000011	C
37	68	44	1000100	D
38	69	45	1000101	E
39	70	46	1000110	F
40	71	47	1000111	G
41	72	48	1001000	H
42	73	49	1001001	I
43	74	04A	1001010	J
44	75	04B	1001011	K
45	76	04C	1001100	L
46	77	04D	1001101	M
47	78	04E	1001110	N
48	79	04F	1001111	O
49	80	50	1010000	P
50	81	51	1010001	Q
51	82	52	1010010	R
52	83	53	1010011	S
53	84	54	1010100	T
54	85	55	1010101	U
55	86	56	1010110	V
56	87	57	1010111	W
57	88	58	1011000	X
58	89	59	1011001	Y
59	90	05A	1011010	Z
60	91	05B	1011011	[

No	Decimal	Hex	Binary	Value
61	92	05C	1011100	\
62	93	05D	1011101	]
63	94	05E	1011110	^
64	95	05F	1011111	_
65	96	60	1100000	`
66	97	61	1100001	a
67	98	62	1100010	b
68	99	63	1100011	c
69	100	64	1100100	d
70	101	65	1100101	e
71	102	66	1100110	f
72	103	67	1100111	g
73	104	68	1101000	h
74	105	69	1101001	i
75	106	06A	1101010	j
76	107	06B	1101011	k
77	108	06C	1101100	l
78	109	06D	1101101	m
79	110	06E	1101110	n
80	111	06F	1101111	o
81	112	70	1110000	p
82	113	71	1110001	q
83	114	72	1110010	r
84	115	73	1110011	s
85	116	74	1110100	t

No	Decimal	Hex	Binary	Value
86	117	75	1110101	u
87	118	76	1110110	v
88	119	77	1110111	w
89	120	78	1111000	x
90	121	79	1111001	y
91	122	07A	1111010	z
92	123	07B	1111011	{
93	124	07C	1111100	|
94	125	07D	1111101	}
95	126	07E	1111110	~
96	127	07F	1111111	DEL

2.2.3. Algoritma

Algoritma merupakan urutan langkah-langkah dalam menentukan suatu masalah. Algoritma juga dapat didefinisikan dengan deretan langkah komputasi yang mentransformasikan masukan menjadi keluaran

2.2.4. Algoritma String Matching

Algoritma pencocokan string dapat dibedakan atas dua cara pembacaan :

1) dari kiri ke kanan

Alogoritma pencarian dengan teknik ini sangat banyak. Hampir sebagian besar algoritma pencarian menggunakan cara pembacaan teks dari kiri ke kanan.

2) dari kanan ke kiri

Dalam Algoritma ini, terdapat algoritma Boyer-Moore yang dianggap merupakan salah satu algoritma yang utama dan algoritma standar dalam pencocokan string. Sampai saat ini terdapat banyak algoritma pencocokan string, ada sekitar 35

algoritma yang bisa digunakan, baik merupkan algoritma yang diciptakan dari awal maupun berupa pengembangan dari algoritma yang sudah ada. [**1**].

2.2.4.1 Algoritma Karp-Rabin

Algoritma Rabin-Karp diperkenalkan pertama kali oleh Michael O. Rabin dan Richard M. Karp pada tahun 1987. Algoritma ini menggunakan tabel *array* dan metode *Hashing* dalam pengoperasiannya. Metode *hashing* ini digunakan terutama untuk meningkatkan kecepatan pencarian dengan meningkatkan pengujian kesetaraan dalam teks. Pada faktanya, fungsi *hash* menyimpan bentuk *string* dalam bentuk lain yaitu enumerasi sehingga suatu *string* tertentu akan memiliki nilai enumerasinya sendiri-sendiri (unik). Karena suatu *string* hanya memiliki sebuah nilai enumerasi maka hal inilah yang digunakan oleh algoritma Rabin-Karp untuk mempercepat pencarian *string* dalam tabel *hash*. Dengan menggunakan metode seperti ini, akan terdapat kebocoran pada pencarian dalam teks yang panjang, karena pada teks yang panjang akan terjadi penomoran *string* yang sama meskipun *string* yang dituju berbeda. Sehingga dibutuhkan verifikasi lebih lanjut terhadap isi *string* tersebut. Hal ini sebenarnya dapat memakan waktu yang cukup lama apabila terjadi pada *substring* yang panjang. Namun fungsi *hash* yang baik akan menjamin kekurang seperti ini jarang terjadi, sehingga rata-rata waktu pencarian rata-rata menggunakan metode ini relatif baik.

Rolling hash adalah fungsi hash dengan basis. Basis biasanya adalah bilangan prima. Berikut ini adalah penggunaan fungsi hash dengan basis. Misalnya kembali menggunakan contoh sebelumnya yaitu dengan menggunakan string ”strategi” sebagai sumber string dan ”rat” sebagai string yang dicari.

1. Dengan menggunakan a= 2 sebagai basis lalu tentukan panjang sumber string disini ”strategi” n = 8. Sedangkan untuk pattern yang dicari ”rat” panjangnya m = 3.
2. Selanjutnya ubah pattern yang dicari dengan menggunakan fungsi Rolling hash. Dengan persamaan :

$$\mathbf{H = C_0 * a^{m-1} + C_1 * a^{m-2} + ... + C_{[m-1]} * a^0}$$

H = nilai hash

C = nilai ASCII karakter

a = nilai basis

m = banyaknya karakter

Jadi nilai hash dari ”rat” adalah 766 didapat dari :

$H = 114*2^2 + 97*2^1 + 116*2^0$

$= 456 + 194 + 116$

$= 766$

(nilai ASCII r = 114, a = 97 , t = 116)

Percobaan ke 1 :

Hash(rat) = 766

Hash (y[0..2]) = 806

s	t	r	a	t	e	g	i
r	a	t					

Nilai hash dari indeks ke 0 samapai dengan 2 tidak cocok dengan nilai hash target maka dilakukan pergeseran.

Percobaan ke 2 :

Hash(rat) = 766

Hash (y[1..3]) = 789

s	t	r	a	t	e	g	i
	r	a	t				

Tidak seperti menggunakan fungsi hash biasa pada percobaan ke dua menggunakan fungsi rolling hash tidak mengalami kecocokan nilai hash. Dengan 798 didapat dari :

$H = 116*2^2 + 114*2^1 + 97*2^0$

$= 464 + 228 + 97$

$= 789$

Maka kembali dilakukan pergeseran.

Percobaan ke 3 :

Hash(rat) = 766

Hash (y[2..4]) = 766

s	t	r	a	t	e	g	i
		r	a	t			

Terjadi kecocokan nilai hash maka algoritma menandai lokasi penemuan dan t melanjutkan pencarian sampai karakter pada sumber string habis.

Percobaan ke 4 :

Hash(rat) = 766

Hash (y[3..5]) = 721

s	t	r	a	t	e	g	i
			r	a	t		

Percobaan ke 5 :

Hash(rat) = 766

Hash (y[3..5]) = 769

s	t	r	a	T	e	g	i
				R	a	t	

Percobaan ke 6 :

Hash(rat) = 766

Hash (y[3..5]) = 715

s	t	r	a	t	e	g	i
					r	a	t

2.2.4.2 Algoritma Zhu-Takaoka

Algoritma Zhu-Takaoka merupakan algoritma pencocokan string (String Matching) yang dipublikasikan oleh Zhu Rui Feng dan Tadao Takaoka pada tahun 1986. Dalam makalahnya, Zhu dan Takaoka menyebut algoritma pencocokan

string ini sebagai BM‟ Algorithm (Boyer-Moore‟ Algorithm). BM‟ Algorithm merupakan algoritma modifikasi dari algoritma pencocokan string Boyer-Moore Algorithm yang dibuat oleh Boyer R.S dan Moore J.S.

Algoritma BM‟ (Algoritma Zhu-Takaoka) yang merupakan modifikasi dari Algoritma BM mempunyai ciri-ciri yang sama dalam proses pencarian string. Ciri-ciri tersebut yaitu adanya tahap Preprocessing, Right-to-left scan, Bad-character rule, dan Good-suffix rule. Perbedaan antara Algoritma Boyer-Moore dan Algoritma Zhu-Takaoka yaitu terletak pada tahap penentuan bad character rule. Dalam Boyer-Moore, bad character hanya terdiri array satu dimensi, sedangkan dalam Zhu-Takaoka dimodifikasi menjadi array dua dimensi. Berikut karakteristik dari Algoritma Zhu-Takaoka.

1) Preprocessing

Prepocessing dalam algoritma Zhu-Takaoka meliputi pencarian nilai pergeseran karakter (good-suffix shift) dan pergeseran karakter jika karakter tidak cocok (bad-character shift). Nilai good-suffix shift ditetukan dalam good-suffix prepocessing sedangkan nilai bad-character shift ditentukan dalam bad-characterprepocessing. Prepocessing dilakukan sebelum proses inti dari pencarian pattern dalam suatu text. (Pratama, 2008)

2) Right-to-Left Scan Rule

Proses inti pencarian Algoritma Zhu-Takaoka yaitu dilakukan dengan teknik Right-to-left scan rule. Teknik ini yaitu melakukan perbandingan antara pattern yang dicari dengan target text secara terbalik yaitu bergerak dari kanan ke kiri. Perbandingan pattern dengan target text dimulai dengan membandingkan karakter

terakhir dari pattern (karakter paling kanan) dengan target text paling kanan. Apabila ada kecocokan maka perbandingan akan dilanjutkan dengan bergerak ke kiri sampai karakter pertama dari pattern. Sedangkan apabila terjadi ketidakcocokan maka akan dilakukan pergeseran, besarnya pergesaran yang dilakukan ditentukan oleh dua fungsi pergeseran yaitu bad-character shift dan good-suffix shift.

3) Bad-Charcter Shift Rule

Aturan bad-character shift dibutuhkan untuk menghindari pengulangan perbandingan yang gagal dari suatu karakter dalam target text dengan pattern. Besarnya pergeseran yang dilakukan dalam aturan bad-character shift disimpan dalam bentuk tabel array dua dimensi, tabel ini terdiri dari beberapa kolom yaitu kolom karakter dan kolom shift yang menunjukkan besarnya pergeseran yang harus dilakukan.

4) Good Suffix Shift Rule

Aturan good-suffix shift dibuat untuk menangani kasus dimana terdapat pengulangan karakter pada pattern. Contoh dibawah ini akan menjelaskan bagaimana aturan bad-character shift gagal dalam menangani adanya perulangan bagian dalam pattern

Langkah – langkah pencarian :

Proses inti pencarian Algoritma Zhu-Takaoka yaitu dilakukan dengan teknik *Right-to-left scan rule.* Teknik ini membandingkan pattern yang dicari dengan sumber teks dimulai dari kanan ke kiri. Berikut contoh pencocokan string dengan algoritma Zhu-Takaoka :

Pattern yang dicari : done

Sumber string : indonesiaindonesi

Dari hasil preproccessing maka dihasil kan tabel ztBc :

	a	d	e	i	N	o	s
a	4	3	4	4	4	4	4
d	4	3	4	4	4	2	4
e	4	3	4	4	4	4	4
i	4	3	4	4	4	4	4
n	4	3	4	4	4	4	4
o	4	3	4	4	1	4	4
s	4	3	4	4	4	4	4

bmGs:

i	0	1	2	3
X[i]	d	o	n	e
bmGs	4	4	4	1

Percobaan ke 1:

i n d o n e s i a i n d o n e s i a

d o n e

Dari hasil percobaan pertama terlihat pada karakter akhir pattern yang dicari yaitu karakter **e** sejajar dengan karakter **o** pada sumber string artinya pada percobaan pertama terjadi ketidakcocokan, maka dilakukan pergeseran sejauh dua karakter. Nilai pergeseran dua karakter ini diperoleh dari tabel ztBc dimana dicocokannya dua karakter akhri pada sumber string yang sejajar dengan karakter akhir pattern. Pada contoh ini diisi dengan karakter **d** dan **o**. maka cek pada tabel ztBc baris **d** dan kolom **o** maka bernilai dua.

Percobaan ke 2 :

i n d o n e s i a i n d o n e s i a

d o n e

Terlihat dari hasil percobaan ke dua pattern yang dicari ditemukan pada sumber string maka akan diberi tanda dan dilanjutkan pada pencocokan selanjutnya dengan pergeseran berdasarkan nilai dari tabel bmGs. Dari tabel bmGs karakter **d** berada pada array ke nol dan memiliki nilai pergeseran empat, maka pattern digeser sejauh 4 karakter.

Percobaan ke 3 :

i n D O N E s I a i n d o N e s i a

d O n e

Dari hasil percobaan ke tiga terlihat pada karakter akhir pattern yang dicari yaitu karakter **e** sejajar dengan karakter **i** pada sumber string artinya pada percobaan ketiga terjadi ketidakcocokan. Sama dengan sebelumnya nilai pergeseran diambil

dari dua karakter akhri pada sumber string yang sejajar dengan karakter akhir pattern yang kemudian dicocokan dengan tabel ztBc. Untuk percobaan ketiga diisi dengan karakter **a** dan **i** dan setelah dicocokan baris **a** dan kolom **i** pada tabel ztBc menghasilkan nilai pergeseran empat.

Percobaan ke 4 :

i n D O N E s i a i n d O n e s i a
d o N e

pada hasil percobaan ke empat terlihat pada karakter akhir pattern yang dicari yaitu karakter **e** sejajar dengan karakter **n** pada sumber string artinya pada percobaan empatpun terjadi ketidakcocokan. Dengan dua karakter akhir yang diambil yaitu **o** dan **n** maka nilai pergeseran dari ketidakcocokan adalah satu.

Percobaan ke 5 :

i n D O N E s i a i n d o n E s i a
d o n E

Terlihat dari hasil percobaan ke lima terjadi kecocokan maka pergeseran berdasarkan tabel bmGs. Dari tabel bmGs karakter **d** berada pada array ke nol dan memiliki nilai pergeseran empat, maka pattern digeser sejauh 4 karakter. Karena panjang sumber string sudah habis maka pencocokanpun dihentikan. Dari contoh diatas dapat ditarik kesimpulan dengan panjang sumber string 18 karakter dan

panjang pattern yang dicari empat dilakukan lima kali percobaan dan menghasilkan dua pola yang cocok.

i n D O N E s i a i n D O N E s i a

2.2.5. Metode Pembangunana Perangkat Lunak

Dalam proses pembangunan perangkat lunak digunakan beberapa model. Adapun model-model tersebut yaitu [5]:

2.2.5.1 Waterfall Model

Model proses *waterfall* merupakan dasar proses aktifitas pembangunan suatu perangkat lunak. Model proses *waterfall* mempunyai beberapa tahapan yang berurutan seperti yang dapat dilihat pada gambar berikut ini :

Setiap tahapan mempunyai arti tersendiri dalam proses pembangunan suatu perangkat lunak serta tahapan-tahapan tersebut harus dilakukan secara berurutan. Berikut ini merupakan penjelasan dari setiap tahapan yang ada dalam *waterfall* model

a. Requirements analysis and definition

Tahap ini adalah tahap untuk mengumpulkan kebutuhan secaralengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh program yang akan dibangun. Tahap ini harus dikerjakan secara lengkap untuk bisa menghasilkan desain yang lengkap.

b. System and software design

Tahap ini adalah tahap mendesain sistem dan perangkat lunak yang dikerjakan setelah kebutuhan selesai dikumpulkan secara lengkap.

c. *Implementation and unit testing*

Tahap ini mendesain sistem dan program maka pada tahap ini desain diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudahditentukan.Program yang dibangun langsung diuji baik secara unit.

d. *Integration and system testing*

Pada tahap ini unit-unit program disatukan kemudiandiuji secara keseluruhan (system testing).

e. *Operation and maintenance*

Tahap ini program dioperasikan pada lingkungannya dan dilakukan pemeliharaan, seperti penyesuaian atau perubahan karenaadaptasi dengan situasi sebenarnya

2.2.6. Object Oriented Programing (OOP)

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunkan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dab perilaku dalam suatu entitas. Model berorientasi objek bermanfaat untuk memahami masalah, komunikasi dengan ahli aplikasi, pemodelan suatu organisasi, meyiapkan dokumentasi serta perancangan program dan basis data. Pertama-tama suatu model analisis dibuat untuk menggambarkan aspek dasar dari domain aplikasi, dimana model tersebut berisi objek yang

terdapat dalam domain aplikasi termasuk deskripsi dari keterangan objek dan perilakunya. Secara spesifik, pengertian berorientai objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karateristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme) dan *inheritance* (pewarisan).

1) Karateristik dari objek

   Identitas berarti bahwa data diukur mempunyai nilai tertentu yang membedakan entitas dan disebut objek. Suatu paragraf dari dokumen, suatu windows dari workstation, dan raja putih dari buah catur adalah contoh dari objek. Objek dapat kongkrit, seperti halnya arsip dalam sistem, atau konseptual seperti kebijakan penjadualan dalam multiprocessing pada sistem operasi. Setiap objek mempunyai sifat yang melekat pada identitasnya. Dua objek dapat berbeda walaupun bila semua atributnya identik Klasifikasi berarti bahwa suatu kegiatan mengumpulkan data (atribut) dan perilaku (operasi) yang mempunyai struktur data sama ke dalam satu grup yang disebut kelas. Paragraf, window, buah catur adalah contoh dari kelas. Kelas merupakan abstraksi yang menjelaskan sifat penting pada suatu aplikasi dan mengabaikan yang lain. Setiap kelas menunjukan suatu kumpulan infinite yang mungkin dari objek. Suatu objek dapat dikatakan sebagai instans dari kelas, setiap

instans dari kelas mempunyai nilai individu untuk setiap nama atribut dan operasi, tetapi memiliki bersama atribut dan operasi dengan instans lain dalam kelas

2) Karateristik Metodologi Berorientasi Objek

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karateristik utama yaitu:

1) Encapsulation

Encapsulation (pengkapsulan) merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas dalam bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain kecuali prosedur yang berada dalam objek itu sendiri.

2) Inheritance

Inheritance (pewarisan) adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi atribut dan metoda dari induknya langsung. Atribut dan metoda dari objek induk diturunkan kepada anak objek, demikian seterusnya. Pendefinisian objek dipergunakan untuk membangun suatu hirarki dari objek turunannya, sehingga tidak perlu membuat atribut dan metoda lagi pada anaknya, karena telah mewarisi sifat induknya.

3) Polymorphism

Polymorphism (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda.

Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.

2.2.7.JAVA

Pada 1991, sekelompok insinyur Sun dipimpin oleh Patrick Naughton dan James Gosling ingin merancang bahasa komputer untuk perangkat konsumer seperti *cable TV Box*. Karena perangkat tersebut tidak memiliki banyak memori, bahasa harus berukuran kecil dan mengandung kode yang liat. Juga karena manufaktur – manufaktur berbeda memilih *processor* yang berbeda pula, maka bahasa harus bebas dari manufaktur manapun. Proyek diberi nama kode ”Green”. Niklaus Wirth, pencipta bahasa Pascal telah merancang bahasa portabel yang menghasilkan *intermediate code* untuk mesin hipotesis. Mesin ini sering disebut dengan mesin maya *(virtual machine)*. Kode ini kemudian dapat digunakan di sembarang mesin yang memiliki *interpreter*. Proyek Green menggunakan mesin maya untuk mengatasi isu utama tentang netral terhadap arsitektur mesin. Karena orang – orang di proyek Green berbasis C++ dan bukan Pascal maka kebanyakan sintaks diambil dari C++, serta mengadopsi orientasi objek dan bukan prosedural. Mulanya bahasa yang diciptakan diberi nama ”Oak” oleh James Gosling yang mendapat inspirasi dari sebuah pohon yang berada pada seberang kantornya, namun dikarenakan nama Oak sendiri merupakan nama bahasa pemrograman yang telah ada sebelumnya, kemudian SUN menggantinya dengan JAVA. Nama JAVA sendiri terinspirasi pada saat mereka sedang menikmati secangkir kopi di sebuah kedai kopi yang kemudian dengan tidak sengaja salah satu dari mereka menyebutkan kata JAVA yang mengandung arti asal bijih kopi. Akhirnya mereka

sepakat untuk memberikan nama bahasa pemrograman tersebut dengan nama Java. Berdasarkan white paper resmi dari SUN, Java memiliki karakteristik berikut :

1) Sederhana (Simple)

Bahasa pemrograman Java menggunakan Sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan pointer yang rumit dan multiple inheritance. Java juga menggunakan automatic memory allocation dan memory garbage collection.

2) Berorientasi objek (Object Oriented)

Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata kedalam objek dan melakukan interaksi antar objek-objek tersebut.

3) Terdistribusi (Distributed)

Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya libraries networking yang terintegrasi pada Java.

4) Interpreted

Program Java dijalankan menggunakan interpreter yaitu Java Virtual Machine (JVM). Hal ini menyebabkan source code Java yang telah dikompilasi menjadi Java bytecodes dapat dijalankan pada platform yang berbeda-beda.

5) Robust

Java mempuyai reliabilitas yang tinggi. Compiler pada Java mempunyai kemampuan mendeteksi error secara lebih teliti dibandingkan bahasa

pemrograman lain. Java mempunyai runtime-Exception handling untuk membantu mengatasi error pada pemrograman.

6) Secure

Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7) Architecture Neutral

Program Java merupakan platform independent. Program cukup mempunyai satu buah versi yang dapat dijalankan pada platform berbeda dengan Java Virtual Machine.

8) Portable

Source code maupun program Java dapat dengan mudah dibawa ke platform yang berbeda-beda tanpa harus dikompilasi ulang.

9) Performance

Performance pada Java sering dikatakan kurang tinggi. Namun performance Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan

2.2.8. NetBeans IDE 6.9.1

Platform NetBeans memungkinkan aplikasi dibangun dari sekumpulan komponen perangkat lunak moduler yang disebut ‘modul’. Sebuah modul adalah suatu arsip Java (Java archive) yang memuat kelas-kelas Java untuk berinetraksi dengan NetBeans Open API dan file manifestasi yang mengidentifikasinya sebagai modul. Aplikasi yang dibangun dengan modul-modul dapat dikembangkan dengan menambahkan modul-modul baru. Karena modul dapat

dikembangkan secara independen, aplikasi berbasis platform NetBeans dapat dengan mudah dikembangkan oleh pihak ketiga secara mudah dan powerful. Pengembangan NetBeans diawali dari Xelfi, sebuah proyek mahasiswa tahun 1997 di bawah bimbingan Fakultas Matematika dan Fisika Universitas Charles, Praha. Sebuah perusahaan kemudian dibentuk untuk proyek tersebut dan menghasilkan versi komersial NetBeans IDE hingga kemudian dibeli oleh Sun Microsystem pada tahun 1999. Sun kemudian menjadikan NetBeans open source pada bulan Juni tahun 2000. Sejak itu komunitas NetBeans terus berkembang.

2.2.9. WAMP Server

Wamp Server adalah paket web server yang bekerja secara pada localhost yang dibuat secara independen dan di instal pada sistem operasi Windows. WAMP adalah singkatan dari dari Windows and the principal components of the package: Apache, MySQL and PHP (or Perl or Python). Apache adalah Web server, MySQL adalah database, PHP adalah bahasa scripting yang dapat memanipulasi informasi yang dibuat di database dan menghasilkan halaman web dinamis konten setiap waktu diminta oleh browser. Program lain juga dapat dimasukkan dalam paket, seperti phpMyAdmin yang menyediakan antarmuka pengguna grafis untuk manajer database MySQL, atau bahasa scripting Python alternatif atau Perl. [**9**]