BAB III
ANALISISDAN PERANCANGAN SISTEM

Aplikasi simulasi kompresi algoritma Huffman Coding ini dirancang dengan menggunakan perangkat lunak Netbeans 7.2. Proses perancangan menggunakan pendekatan Object Oriented Analysis and Design (OOAD) dengan menggunakan notasi Unified Modeling Language atau UML yang mencakup *class diagram, use case diagram, sequence diagram, activity diagram* dan rancangan layar.

3.1 Analisis

Menurut Kamus Besar Bahasa Indonesia (KBBI), simulasi adalah metode pelatihan yang meragakan sesuatu dalam bentuk tiruan yang mirip dengan keadaan yang sesungguhnya, penggambaran suatu sistem atau proses dengan peragaan berupa model statistik atau pemeranan.

Pada aplikasi Simulasi Huffman Coding ini, Huffman Coding disimulasikan melalui penggambaran-penggambaran dari masing-masing proses dalam algoritma Huffman Coding. Penggambaran proses dilakukan secara berurutan.

3.1.1 Analisis Kebutuhan Proses

Dari hasil ringkasan analisis di atas dan teori pada bab sebelumnya maka penulis dapat mengambil garis besar beberapa kebutuhan proses aplikasi Simulasi Huffman Coding adalah sebagai berikut:

1. Proses menghitung frekuensi dan probabilitas kemunculan setiap karakter.
   Frekuensi kemunculan karakter adalah banyaknya karakter muncul dalam sebuah teks. Sedangkan untuk probabilitas dapat dihitung dengan menggunakan persamaan 3.1.

$$P = \frac{f}{length} \qquad \text{... (3.1)}$$

(David Salomon: 2006, 13)

   Dimana:

   P = probabilitas

f = frekuensi karakter

length = panjang teks/jumlah karakter

Misalnya dalam sebuah teks "*She sells seashells by the seashore*" didapat frekuensi dan probabilitas masing-masing karakter yang ditunjukan oleh Tabel 3.1 (huruf kecil dan huruf kapital dianggap beda karakter).

Tabel 3.1 Frekuensi dan Probabilitas Kemunculan Karakter.

Karakter	Frekuensi	Probabilitas
S	1	0.028
H	4	0.114
E	7	0.200
A	2	0.057
B	1	0.028
T	1	0.028
S	7	0.200
L	4	0.114
	5	0.142
O	1	0.028
R	1	0.028
Y	1	0.028

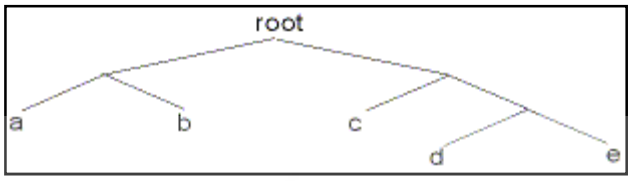
2. Proses pembuatan Huffman Tree.

   Proses pembuatan *tree* akan menggunakan algoritma *priority queue*, yaitu karakter atau simbol akan diurutkan berdasarkan frekuensi kemunculannya. Karakter dengan frekuensi lebih kecil diberi prioritas lebih besar dibandingkan dengan karakter dengan frekuensi lebih besar. Dengan menggunakan *priority queue*, algoritma untuk membangun Huffman Tree adalah:

   1. Ambil 2 buah karakter dengan frekuensi terkecil.
   2. Buat sebuah parent node untuk kedua karakter tersebut. Nilai *parent node* ini adalah jumlah frekuensi dari kedua *node*.
   3. Hapus 2 karakter dari priority *queue*.
   4. Ulangi langkah 1 sampai 3 hingga semua karakter dihapus dari *priority queue*.

3. Proses *encoding*.

   Encoding dilakukan dengan menelusuri Huffman Tree yang telah dibuat sebelumnya. Penelusuran dimulai dari karakter yang akan di-encode, tambahkan nol setiap bergerak ke kiri dan 1 setiap bergerak ke kanan, setelah mencapai *root*. Contoh: berdasarkan Huffman Tree yang ditunjukan oleh Gambar 3.1 maka prefix code untuk karakter d adalah 110.



Gambar 3.1 contoh Huffman Tree

4. Proses untuk menghitung hasil kompresi.

   Setelah masing-masing *prefix code* untuk tiap karakter ditemukan. Tiap karakter dalam file diwakili oleh *prefix code* kemudian disusun kembali seperti file awal dan dihitung jumlah bit dalam file tersebut setelah dikompresi. Rasio kompresi dapat dihitung dengan menggunakan persamaan 2.1.

5. Proses dekompresi.

   Proses dekompresi dilakukan dengan menelusuri Huffman Tree berdasarkan nilai bit pada *prefix code* untuk mencari karakter. Penelusuran dimulai dari *root*, mulai dengan bit pertama, jika bit bernilai 0 maka penelusuran diarahkan ke kiri dan sebaliknya jika bit bernilai 1 maka penelusuran diarahkan ke arah kanan. Jika penelusuran mencapai sebuah *leaf* maka kita telah men-*decode* sebuah karakter, dan penelusuran kembali dimulai dari *root*. Begitu seterusnya hingga semua bit habis.

3.1.2 Analisis Kebutuhan Masukan

Aplikasi simulasi algoritma Huffman Coding ini kompresi dilakukan terhadap sebuah teks yang valid. Kriteria teks *valid* untuk aplikasi ini adalah:

1. Teks yang dapat diterima oleh sistem adalah sekumpulan karakter dapat berupa huruf kecil, huruf kapital, angka, dan spasi serta simbol lain seperti

dalam himpunan karakter dan simbol berikut:" !\"#$%&`()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~".

2. Panjang teks lebih dari nol.

Agar proses simulasi berjalan secara dinamis, maka teks yang dijadikan sebagai parameter kompresi diinput oleh *user*. Sehingga memungkinkan munculnya variasi-variasi *output* pada saat aplikasi berjalan. Jika *user* memasukan teks atau karakter yang tidak *valid*, proses simulasi tidak dapat dimulai.

3.1.3 Analisis Kebutuhan Keluaran

Keluaran dari aplikasi simulasi Huffman coding adalah berupa tampilan proses atau hasil proses dari urutan algoritma kompresi dan dekompresi Huffman Coding. Dalam hal ini penulis memanfaatkan komponen Java Swing yaitu JPanel. Kebanyakan komponen Swing mempunyai tampilan tersendiri, tapi komponen JPanel tampilan awalnya polos sehingga cocok digunakan sebagai bidang gambar. Untuk mengatur tampilan komponen Swing dapat menggunakan objek *Graphics2D*. Dengan memanggil metode-motode pada objek Graphics2D kita dapat mengatur tampilan JPanel. Contoh: kita dapat menggambar sebuah garis dengan memanggil operasi *drawLine*(int x1, int y1, int x2, int y2). Metode-metode lain dalam objek *Graphics2D* dapat dilihat di Tabel 2.2.

Untuk memperjelas proses simulasi pada tahap encoding dan decoding, maka gambaran proses penelesuran Huffman Tree akan ditampilkan secara animasi. Dengan mengimplementasikan kelas *Runnable* dan memanfaatkan metode *run*(), *start*(), *stop*(), dan *wait*() kita dapat mengatur tampilan Jpanel secara dinamis untuk menciptakan efek bergerak (animasi).

Proses-proses yang ditampilkan dalam aplikasi Simulasi Huffman Coding adalah sebagai berikut:

1. Frekuensi Karakter ditampilkan dalam bentuk grafik.
2. Huffman Tree.
3. Proses *encoding*.

4. Hasil kompresi berupa: teks awal, ukuran teks awal (bit), teks hasil kompresi (prefix code), ukuran teks hasil kompresi (bit) dan rasio kompresi.
5. Proses dekompres/*decoding*.

3.1.4 Analisis Kebutuhan Perangkat Pendukung

3.1.4.1 Perangkat Lunak

Untuk membantu penulis dalam perancangan dan pembuatan aplikasi simulasi Huffman Coding ini, penulis menggunakan perangkat lunak diantaranya:

1. Operating sistem Windows 7 64bit
2. Java Development Kit versi 1.7.2
3. Java Runtime Environment versi 1.7.2
4. Netbeans IDE versi 7.2
5. Visual Paradigm for UML Community Edition versi 8.3

3.1.4.2 Perangkat Keras

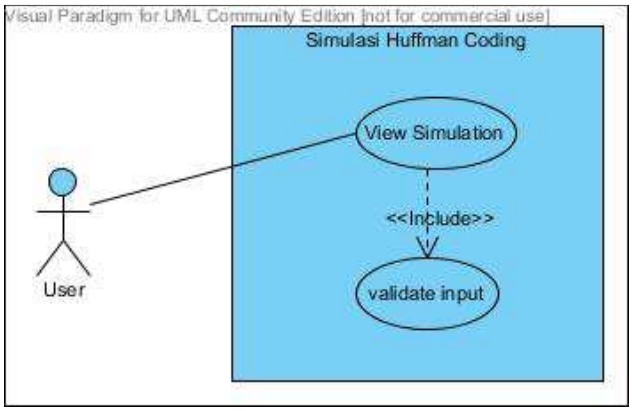
Untuk dapat menjalankan perangkat lunak diatas tentu saja dibutuhkan sistem komputer. Untuk spesifikasi komputer atau perangkat keras yang penulis gunakan dalam perancangan dan pembuatan aplikasi simulasi algoritma Huffman Coding ini adalah sebagai berikut:

1. Prosesor Intel Dual Core 2.1 Ghz.
2. Memori DDR3 8gb.
3. Layar 14 inchi dengan resolusi 1366x768 pixel.

Tentu saja spesifikasi di atas tidak mengikat, untuk spesifikasi minimum perangkat-perangkat lunak di atas dapat dilihat pada website masing-masing pengembang.

3.1.5 Use Case Diagram

Gambar 3.2 menggambarkan interaksi antara pengguna dengan aplikasi dan apa saja yang user dapat lakukan atau terima dari aplikasi. Untuk dapat melihat simulasi, user harus terlebih dahulu teks yang akan dikompresi.



Gambar 3.2 Use case diagram

Tabel 3.2 Use case Description View Simulation

Use case name		**View Simulation**
Goal in context		User dapat melihat proses simulasi.
Pre-condition		Aplikasi berjalan, dan user telah memasukan teks yang akan dikompresi.
Successful end condition		User dapat melihat proses simulasi.
Failed End Condition		Proses simulasi tidak dapat dimulai.
Primary Actors		User (pengguna).
Secondary Actors		None.
Trigger		User memulai proses simulasi
Included Cases		Validate Input
Main Flow	Step	Action
	1	User menjalankan aplikasi
	2	User memasukan teks yang akan dikompresi
	3	User menekan tombol start
	4	Aplikasi memeriksa masukan dari user
	Include::Validate Input	
	5	Aplikasi menampilkan proses simulasi
Extension	Step	Branching Action
	4.1	User tidak memasukan teks atau teks tidak valid
	4.2	Proses simulasi tidak dapat dimulai. Sistem menampilkan pesan kesalahan.

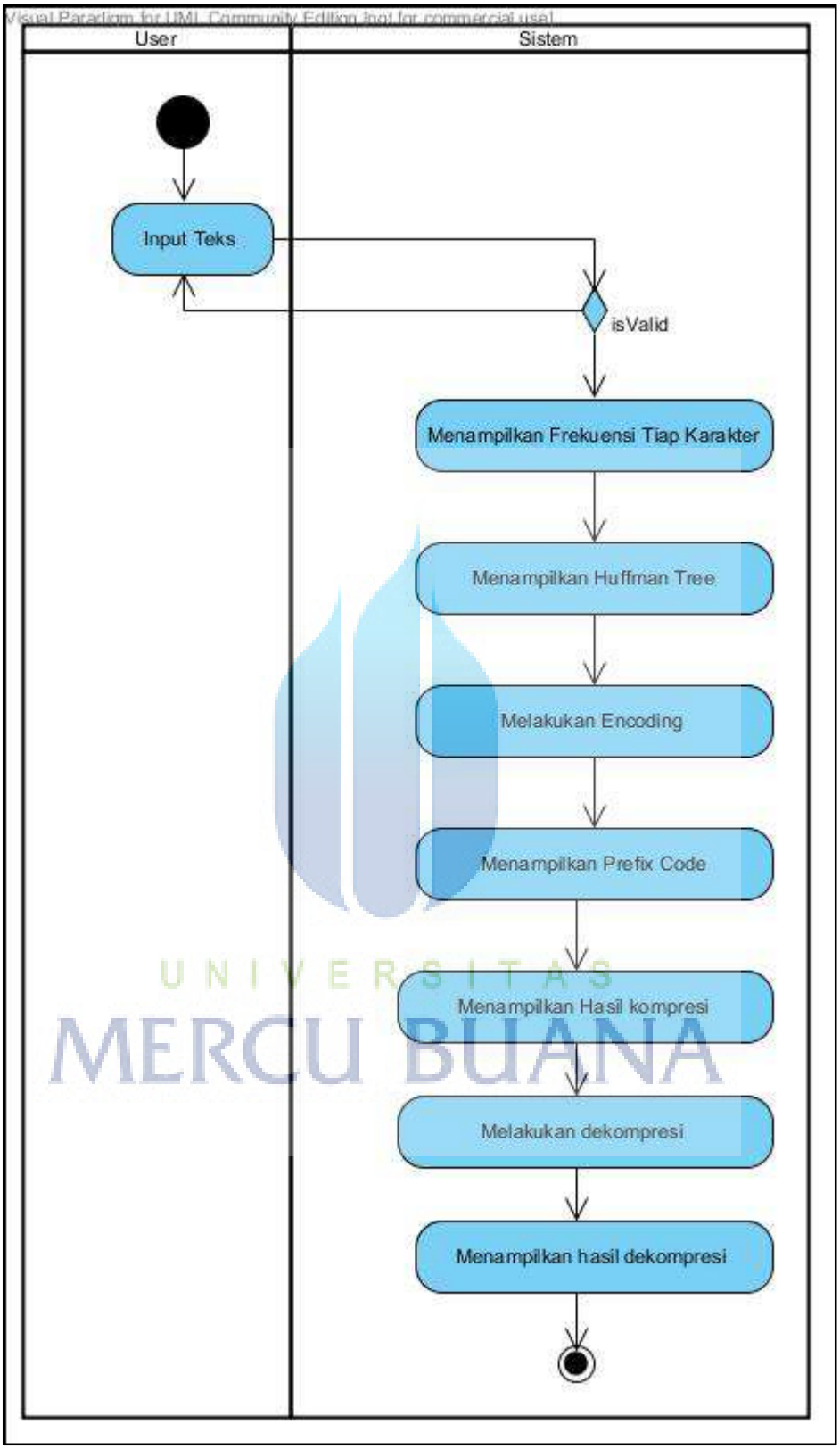
Tabel 3.3 Use case Description Validate Input

Use case name		Validate Input
Goal in Context		Melakukan pengecekan terhadap input sebelum proses simulasi dapat dimulai
Preconditions		User menjalankan aplikasi.
Succesful End Condition		Input user valid
Failed End Condition		Input user tidak valid
Primary Actor		User (pengguna)
Secondary Actor		None
Trigger		User memulai proses simulasi
Main Flow	Step	Action
	1	User memasukan teks
	2	Input user valid
Extensions	Step	Branching Action
	2.1	Input user tidak valid

3.2 Perancangan

3.2.1 Activity Diagram

Aplikasi simulasi Huffman Coding menampilkan proses kompresi dan dekompresi setelah user memasukan teks untuk dikompres. User hanya perlu menginput teks pada field yang tersedia kemudian menekan tombol Start, kemudian sistem akan memeriksa input, jika input valid maka proses simulasi dimulai, dan jika input tidak valid maka sistem akan meminta user untuk memasukan teks lagi. Setelah itu sistem akan memulai proses simulasi. Agar proses simulasi mudah diikuti oleh user, maka tiap akhir proses sistem akan *pause*, dan akan melanjutkan ke proses selanjutnya jika user melakukan klik pada layar sistem.



Gambar 3.3 Activity Diagram

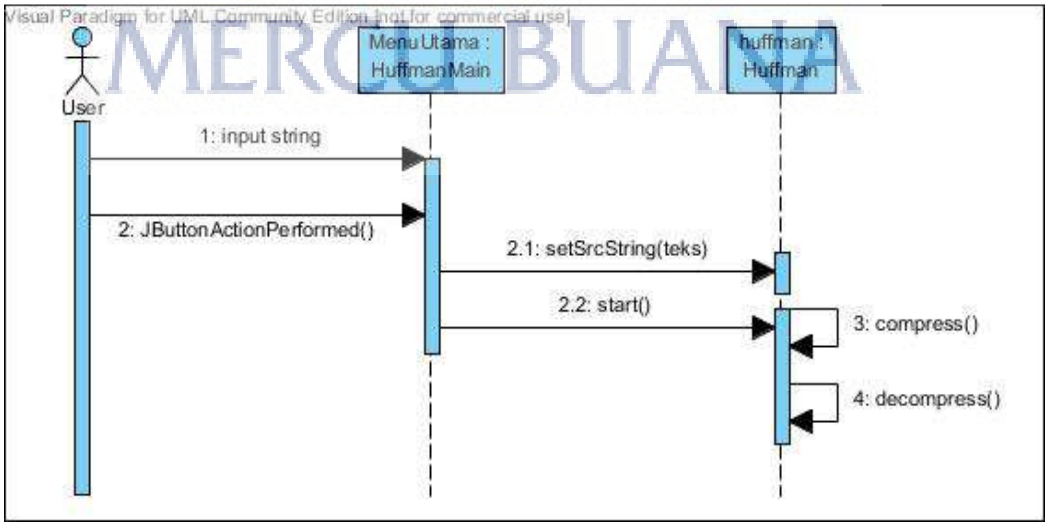
3.2.2 Sequence Diagram

Gambar 3.4 menunjukan sequence diagram aplikasi simulasi Huffman Coding. Identifikasi dan klasifikasi objek pada sequence diagram adalah:

1. User adalah pengguna yang berinteraksi dengan sistem.
2. MenuUtama adalah antarmuka grafis berfungsi sebagai *controller* dan *container* untuk objek huffman.
3. Huffman adalah objek yang bertugas untuk melakukan proses simulasi. Proses menampilkan frekuensi karakter, menampilkan Huffman Tree, menampilkan proses *encoding*, menampilkan *prefix code*, dan menampilkan hasil kompresi digabung dalam satu operasi yaitu *compress*(). Sedangkan proses menampilkan proses *decoding* dan menampilkan hasil *decoding* digabung dalam operasi *decompress*().

Dapat dilihat pada gambar 3.4 sekuensial untuk aplikasi simulasi algoritma Huffman Coding ini adalah:

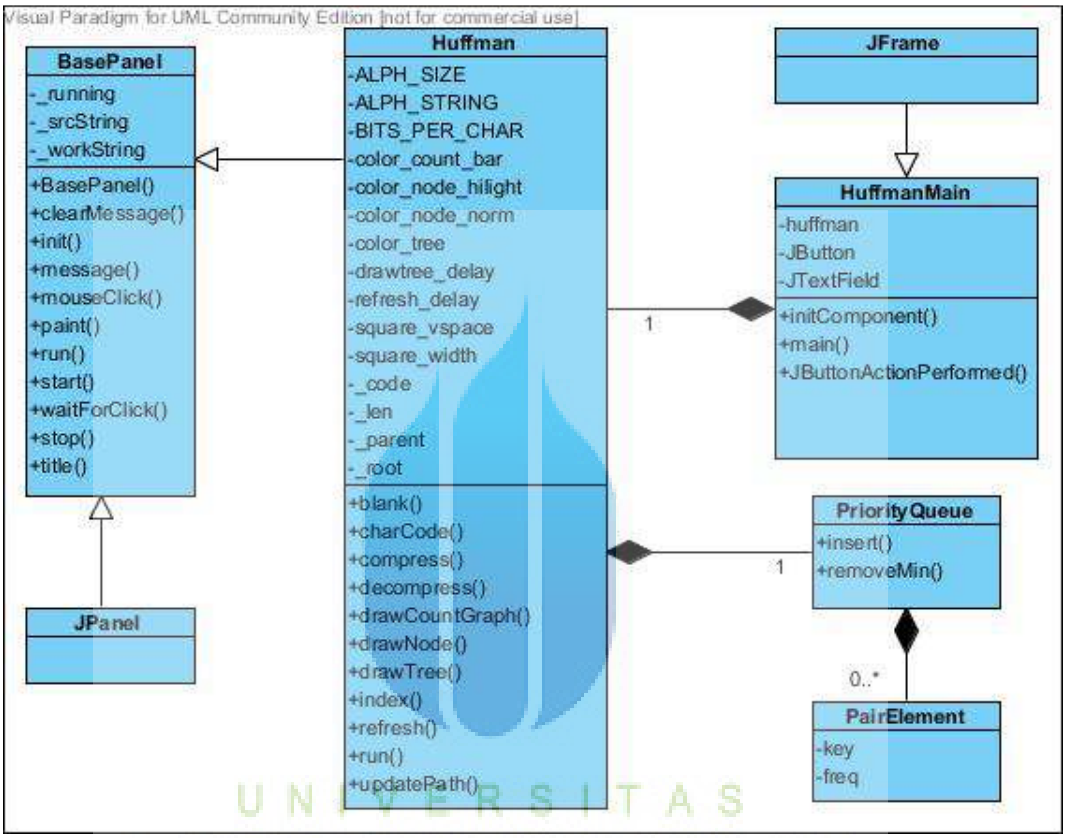
1. User terlebih dahulu memasukan teks yang akan dikompresi, kemudian menekan tombol start.
2. Teks kemudian diteruskan oleh MenuUtama ke objek huffman.
3. MenuUtama memanggil operasi start() untuk memulai proses simulasi.
4. Huffman memanggil operasi compress() kemudian decompress().



Gambar 3.4 Sequence Diagram

3.2.3 Class Diagram

Seperti yang dapat dilihat pada kelas diagram gambar 3.4 dalam aplikasi simulasi algoritma Huffman ini terdiri dari beberapa kelas, masing-masing kelas mempunyai atribut dan operasi yang berbeda.



Gambar 3.5 Class Diagram

1. Kelas BasePanel

   Kelas BasePanel diturunkan dari kelas JPanel. BasePanel memiliki beberapa atribut-atribut dan operasi-operasi umum diantaranya:

   a. Atribut srcString, atribut ini merupakan sebuah variabel yang menyimpan teks yang akan dikompresi.
   b. Operasi *setSrcString*(), operasi untuk menyimpan teks yang diinput ke dalam atribut *srcString*.
   c. Operasi *message*(), untuk menampilkan pesan pada panel.
   d. Operasi *clearMessage*(), untuk menghapus pesan pada bidang kerja.
   e. Operasi *title*(), untuk memberikan judul pada setiap proses.

f. Operasi *waitForClick*(), untuk mendeteksi masukan dari user yang berupa klik pada bidang kerja.

Beberapa operasi hasil implementasi kelas Runnable adalah:

a. start().

Operasi *start*() dipanggil untuk memulai proses.

b. stop().

Operasi *stop*() dipanggil untuk menghentikan proses.

c. run().

Ketika operasi *start*() dipanggil, operasi *run*() akan dijalankan secara konkuren.

2. Kelas Huffman

Kelas Huffman diturunkan dari kelas BasePanel, seperti halnya kelas BasePanel, kelas Huffman merupakan bidang gambar namun mempunyai fungsi khusus yaitu untuk proses simulasi Huffman. Dalam kelas huffman ini terdapat beberapa atribut dan operasi diantaranya:

a. Atribut BITS_PER_CHAR

Nilai ukuran bit untuk satu karakter, jika diasumsikan ukuran 1 karakter adalah 8 bit maka nilai awalnya adalah 8.

b. Atribut color_count_bar

Atribut warna untuk grafik frekuensi karakter.

c. Atribut ALPH_STRING

Variabel untuk menyimpan himpunan karakter atau simbol yang dapat diterima oleh sistem untuk dikompresi.

d. Atribut color_tree

Variabel untuk warna tampilan Huffman Tree.

e. Atribut refresh_delay

Nilai delay untuk animasi, dalam satuan milidetik.

f. Operasi *run*()

Operasi utama dalam kelas Huffman, operasi ini menjalan rangkaian proses simulasi dalam aplikasi Huffman Coding ini. Pada operasi ini dipanggil juga operasi *compress*() dan *decompress*().

g. Operasi *compress*()

Operasi untuk proses kompresi, dalam operasi ini juga memanggil operasi lain yaitu: *drawCountGraph*() untuk menampilkan frekuensi karakter, *drawTree*() untuk menampilkan HuffmanTree, proses encoding dan menampilkan hasil kompresi.

h. Operasi *decompress*()

Operasi untuk proses dekompresi dan menampilkan hasil dekompresi.

3. Kelas HuffmanMain

Kelas HuffmanMain diturunkan dari kelas *Jframe*, berfungsi sebagai *container* untuk objek huffman, *jtextfield* dan *jbutton*.

4. Kelas PairElement

Kelas PairElement berfungsi sebagai objek yang menyimpan pasangan karakter dan frekuensinya. Atribut *key* untuk menyimpan index karakter dan *freq* untuk menyimpan nilai frequensinya.

5. Kelas PriorityQueue

Kelas *PriorityQueue* diturunkan dari kelas *Vector*, kelas ini berfungsi menyimpan objek PairElement yang urutkan berdasarkan frekuensinya. Ada dua operasi utama dalam kelas ini diantaranya:

a. Insert()

Memasukan objek *PairElement* kemudian diurutkan berdasarkan nilai frekuensi, dari yang paling kecil ke yang paling besar.

b. removeMin()

Mengeluarkan frekuensi paling kecil ke dalam sebuah objek.

3.2.4 Perancangan Antar Muka

Dapat dilihat pada Gambar 3.4 rancangan antar muka aplikasi simulasi algoritma Huffman Coding terdapat 3 komponen yaitu:

a. Textfield

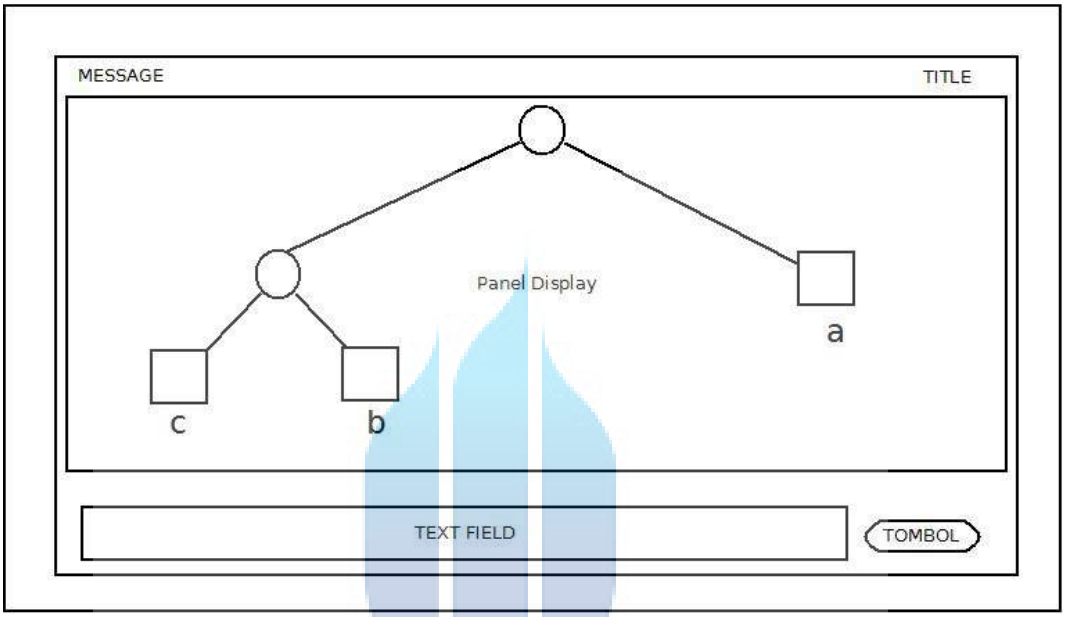
Textfield berfungsi sebagai tempat untuk memasukan paramater teks yang akan dikompresi. Untuk komponen *textfield* ini penulis memanfaatkan komponen Swing JTextField.

b. Tombol

Setelah memasukan teks proses simulasi dimulai setelah user menekan tombol ini. Ketika tombol ini ditekan proses simulasi terhadap teks dimulai. Untuk komponen tombol ini penulis memanfaatkan komponen Swing JButton.

c. Panel Display

Urutan proses simulasi ditampilkan melalui panel display ini. Panel Display ini merupakan representasi dari objek dari kelas huffman.



Gambar 3.6 Rancangan Antar Muka