“VISUAL NOVEL MAKER” UNTUK APLIKASI TELEPON GENGGAM MENGGUNAKAN TEKNOLOGI JAVA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

PANDU BAGUS BASKORO
NIM. 0310630102

KEMENTERIAN PENDIDIKAN NASIONAL
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2010

“VISUAL NOVEL MAKER” UNTUK APLIKASI TELEPON GENGGAM MENGGUNAKAN TEKNOLOGI JAVA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

.

Disusun oleh:

PANDU BAGUS BASKORO
NIM. 0310630102

Telah diperiksa dan disetujui oleh
DOSEN PEMBIMBING**:**

M. Aswin Ir., MT.	**Himawat A. ST., MSc.**
NIP. 19640626 199002 1 001	NIP. 19801018 200801 1 003

"VISUAL NOVEL MAKER" UNTUK APLIKASI TELEPON GENGGAM MENGGUNAKAN TEKNOLOGI JAVA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
PANDU BAGUS BASKORO
NIM. 0310630102

Skripsi ini telah diuji dan dinyatakan lulus pada
Tanggal 4 Agustus 2010

DOSEN PENGUJI

Adharul Muttaqin, ST. MT.
NIP. 19760121 200501 1 001

Ali Mustafa, ST. MT.
NIP. 19710601 200003 1 001

Bambang Siswoyo, Ir., MT.
NIP. 19621211 198802 1 001

Mengetahui
Ketua Jurusan Teknik Elektro

Rudy Yuwono, ST., M.Sc.
NIP. 19710615 199802 1 003

KATA PENGANTAR

Puji Syukur kehadirat Allah SWT karena dengan berkat rahmat dan karunia serta ridho-Nya penyusunan skripsi ini dengan judul “”*Visual Novel Maker*” untuk Aplikasi Telepon Genggam Menggunakan Teknologi Java” dapat diselesaikan. Penulis menyadari bahwa kajian ini tidak akan mencapai titik akhir penyelesaian tanpa bantuan berbagai pihak, karenanya penulis mengucapkan terima kasih kepada :

1. Bapak, ibu dan keluarga yang telah memberikan dukungan yang tak ternilai selama ini.
2. Rudy Yuwono, ST, MSc. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
3. M. Azis Muslim, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
4. M. Aswin Ir., MT. selaku dosen pembimbing pada penyusunan skripsi ini.
5. Himawat AS. ST., MSc. selaku dosen pembimbing pada penyusunan skripsi ini.
6. Waru Djuriatno, ST. MT. selaku KKDK jurusan Informatika.
7. Bapak dan Ibu dosen serta karyawan Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
8. Teman, sahabat, dan saudaraku yang telah bersama menjalani hari dengan suka dan duka. Terima kasih atas pelajaran hidup yang sangat berharga, dan semua kenangan tidak akan bisa terlupakan.
9. Serta semua pihak yang tak dapat disebutkan satu persatu yang telah turut membantu baik secara langsung maupun tidak langsung dalam penyelesaian skripsi ini.

Tiada gading yang tak retak, tersadar bahwa skripsi ini sangat jauh dari kesempurnaan. Karenanya, segala kritik dan saran yang sifatnya membangun dari pembaca tentang isi skripsi ini akan diterima dengan senang hati. Akhir kata, penulis berharap, semoga skripsi ini dapat bermanfaat bagi semua pihak yang membutuhkan.

Malang, Agustus 2010,

Penyusun

ii

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Pandu Bagus Baskoro. 2010. : "Visual Novel Maker" untuk aplikasi Telepon Genggam Menggunakan Teknologi Java. Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing : M. Aswin, Ir., MT. dan Himawat, ST., M.Sc.

Perkembangan teknologi *handphone* sekarang ini sangat pesat terutama *mobile game*. Oleh karena itu dibutuhkan system yang dapat membantu pengembang untuk mempermudah pembuatan game tersebut. Pada tugas akhir ini, akan dibuat aplikasi game maker dengan genre *visual novel* menggunakan bahasa Java yang dapat menggenerasikan script menjadi script J2me yang nantinya dengan menggunakan Java compiler dapat dijalankan pada perangkat *handphone*.

Gambar serta tulisan yang ingin digunakan diinput kedalam panel. Gambar dan Tulisan yang merupakan input tersebut kemudian akan digenerasikan menjadi script J2me yang sebelumnya telah dibuat sebagai *standard script*. Sistem akan merubah input berupa gambar dan tulisan pada aplikasi menjadi *script* j2me yang dapat dikompilasi menggunakan JDK menjadi aplikasi *handphone* berupa *game* bergenre *visual novel*.

Pengujian unit dan integrasi dilakukan terhadap beberapa bagian sistem untuk mengetahui unjuk kerja dan performa bagian-bagian sistem baik secara independen maupun dalam kesatuan fungsi tertentu. Pengujian performansi sistem juga dilakukan terhadap sistem secara keseluruhan untuk mengetahui kemampuan sistem secara utuh. Dari hasil pengujian yang dilakukan terhadap performa aplikasi menunjukkan bahwa Aplikasi "*Visual Novel Maker*" untuk Telepon Genggam dapat berfungsi dengan baik.

The first word that came was the word "Dream"
It came to me as I lay sleeping
And the darkness deep inside my heart
Was gently driven away

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi yang terdapat pada *handphone* sekarang ini sangat pesat. *Handphone* tidak hanya digunakan sebagai alat komunikasi suara saja, tetapi juga sebagai alat untuk memudahkan pengguna *handphone* dalam kehidupan sehari-hari. Fasilitas untuk pengaksesan internet,messenger, *e-mail, organizer, game* dan sebagainya yang dapat langsung dipakai melalui *handphone*, merupakan beberapa contoh fasilitas yang ditawarkan kepada pengguna *hanphone.*

Pada awal perkembangan aplikasi *mobile device* terutama *handphone*, masing-masing perusahaan pengembang membuat *platform* sendiri-sendiri, sehingga aplikasi yang dibuat hanya untuk sebuah produk atau merek tertentu saja. Hal ini tidaklah menguntungkan bagi perkembangan aplikasi itu sendiri karena aplikasi yang ada sangat bergantung pada merek *handphone*. Melihat perkembangan ini, selain dibutuhkan suatu rumusan standarisasi, diperlukan juga sebuah bahasa pemrograman yang memiliki kebebasan platform sehingga dapat menjembatani platform yang berbeda-beda tersebut. Secara spesifik Sun Microsystem mengeluarkan edisi khusus dari Java yaitu Java 2 *Micro Edition* (J2ME) untuk keperluan pengembangan aplikasi dibidang *mobile devices.*

Pada perkembangannya sekarang, kebanyakan vendor *handphone* telah menyertakan fasilitas "*Java Enable*" pada produknya. Adanya fitur "*Java Enable*" pada *handphone* memungkinkan penggunanya semakin mudah untuk menambahkan sendiri aplikasi-aplikasi berbasis J2ME. Kemampuan konektivitas J2ME dengan jaringan menggunakan koneksi *bluetooth* untuk mengakses informasi yang ada pada piranti lain yang terhubung, memungkinkan pertukaran data antara handphone dengan piranti-piranti lainnya, membuat perkembangan aplikasi J2ME sekarang ini semakin menarik.

Saat ini, *game* merupakan aplikasi yang masih terus berkembang dan sangat diminati pengguna sebagai salah satu fitur hiburan pada telepon genggam. Menggunakan J2ME, dimungkinkan untuk membuat suatu aplikasi yang dapat dijalankan pada telepon genggam yang berbeda baik merek dan tipenya.

Berawal dari kondisi yang telah dipaparkan di atas, maka dirumuskanlah rencana perancangan serta pembuatan aplikasi *visual novel maker* menggunakan basis Java SE. Aplikasi ini berupa sebuah program yang dapat mengcoding text, gambar, dan suara yang diinput menjadi sebuah script J2ME yang jika dikompolasikan menggunakan program JBuilder akan menghasilkan sebuah aplikasi untuk handphone yang memiliki fitur java enabled.

Aplikasi yang akan dihasilkan berupa *visual novel*. Dalam aplikasi ini, pemain cukup mengikuti cerita yang disajikan dengan tampilan teks dan gambar seperti sedang membaca novel.

Membuat aplikasi visual novel maker dan mobile *game* dengan menggunakan bahasa Java akan memberikan beberapa keuntungan, diantaranya :

- *Platform independence*, game yang dibuat menggunakan Java dapat dijalankan pada berbagai macam *platform* yang telah ada.
- *Improved object-oriented platform*, Java di desain dengan konsep object-oriented yang memiliki keunggulan : *reusable*, enkapsulasi, polimorfisme, dan pewarisan.
- Java mudah dipelajari, karena beberapa fiturnya yang membantu programmer dalam menulis program (terutama untuk pengembangan game) yaitu *automatic garbage collection* serta penghapusan penggunaan *pointer*.
- Java juga menyediakan berbagai fasilitas lain yang sangat berguna bagi lingkungan pengembangan game. Fasilitas ini berupa pengolahan GUI (Graphical User Interface) dengan AWT atau Swing, *threads*, pengolahan input dari *keyboard*, memainkan *file* suara atau musik, serta tentu saja pengolahan grafis 2D.

1.2 Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan diatas, rumusan masalah difokuskan pada :

1. Merancang sebuah sistem "*visual novel*" berbasis J2ME sebagai dasar dari aplikasi yang akan dihasilkan menggunakan aplikasi *visual novel maker*.

2. Merancang aplikasi *Visual Novel Maker* yang dapat menggenerate gambar dan script yang di input, mengolahnya dan menjadikannya output berupa file .jar yang dapat dijalankan pada handphone.
3. Menerapkan dan menguji game secara langsung pada handphone.

1.3 Batasan Masalah

Dalam perencanaan dan pembuatan skripsi ini perlu dilakukan pembatasan masalah. Pembatasan masalah yang diajukan dalam penyusunan tugas akhir ini antara lain:

1. Pembahasan difokuskan kepada pembuatan aplikasi "visual novel maker".
2. Aplikasi *Game* yang dihasilkan berupa *visual novel*.
3. *Input* yang digunakan pada *visual novel* adalah *input* dari *keypad* yang terdapat pada handphone.
4. *File* gambar yang digunakan dalam pembuatan *visual novel* adalah file yang bertipe jpeg, gif (non animated) , png.
5. Software aplikasi (*visual novel maker* dan *visual novel*) dibuat menggunakan platform Java (*Java SE* dan *Java2 Micro Edition*) dengan tool IDE Netbeans 6.5.b.
6. Handphone yang digunakan mendukung platform J2ME dengan profile MIDP 2.0.
7. Pemain menggunakan telepon genggam yang proses instalasi aplikasi ke telepon genggam dilakukan secara manual melalui kabel data atau *bluetooth* atau *infrared* dengan cara instalasi yang telah diatur oleh masing-masing produsen telepon genggam.

1.4 Tujuan Penulisan

Tujuan dari tugas akhir ini adalah bagaimana merancang dan membuat aplikasi untuk mempermudah pengguna yang ingin membuat novel interaktif berupa *visual novel* yang menarik, murah, dan dapat dijalankan melalui telepon genggam tanpa harus terdedikasi pada pemrograman yang rumit.

1.5 Manfaat

Manfaat yang diharapkan dapat diperoleh melalui pengerjaan skripsi ini adalah :

1. Mengembangkan sebuah aplikasi baru dimana dapat membantu dan mempermudah untuk mengembangkan *mobile programming* dimana pengguna tidak perlu untuk berpikir rumit dan terdedikasi pada pemrograman.
2. Menjadikan media *handphone* sebagai sarana lain untuk berkreasi dan bercerita, dan secara tidak langsung juga mengajarkan pemrograman sederhana bagi pengguna aplikasi.
3. Menambah bekal pengetahuan dan wawasan baru serta dapat belajar memecahkan masalah dengan efisien dan berfikir secara lebih kritis dan tepat dalam membuat sebuah keputusan
4. Mengimplementasikan ilmu yang didapat dari kegiatan perkuliahan.
5. Membantu perkembangan *software development* di Indonesia.
6. Bagi penyusun, dengan skripsi ini diharapkan penyusun dapat menerapkan ilmu yang telah diperoleh di teknik elektro konsentrasi sistem informasi dan komputer Universitas Brawijaya dan menambah wawasan dalam bidang rekayasa perangkat lunak

1.6. Sistematika Penulisan

Sistematika penulisan dalam tugas akhir ini adalah sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi pembahasan, dan sistematika pembahasan.

BAB II Dasar Teori

Membahas teori-teori yang mendukung dalam perancangan dan pembuatan *visual novel maker*.

BAB III Metodologi

Berisi tentang metode penelitian yang digunakan dalam perancangan dan pengujian sistem.

BAB IV **Analisis kebutuhan dan Perancangan**

Membahas tentang analisa kebutuhan dari sistem dan kemudian merancang hal-hal yang berhubungan dengan analisa tersebut. Membahas tentang perencanaan dan pembuatan sistem.

BAB V **Implementasi**

Mengubah perancangan yang ada menjadi algoritma dan kode-kode program.

BAB VI **Pengujian**

Memuat proses dan hasil pengujian terhadap sistem yang telah direalisasikan.

BAB VII **Kesimpulan dan Saran**

Memuat kesimpulan dan saran-saran.

BAB II

DASAR TEORI

Untuk memudahkan dalam memahami dasar-dasar dalam perencanaan sistem ini, maka perlu penjelasan tentang dasar teori yang digunakan dalam penulisan skripsi ini. Beberapa dasar teori yang dimaksud diantaranya adalah teori tentang *Visual Novel*, teknologi Java untuk pengembangan sistem dan UML untuk memodelkan sistem perangkat lunak.

2.1 *Visual Novel*

Visual Novel merupakan sebuah novel fiksi interaktif yang kebanyakan menggunakan gambar statik, biasanya dengan style anime. Sesuai namanya, visual novel hampir menyerupai novel tetapi dengan media campuran seperti audio dan gambar, karena itulah media yang digunakan pada *visual novel* adalah perangkat elektronik seperti *PC*, *video game console*, dan *handphone*.

Visual novel berbeda dari tipe game lainnya karena gameplaynya yang begitu minim. Tipikalnya adalah interaksi player yang dibatasi hanya mengklik untuk mengikuti text, gambar dan suara yang terus berjalan hingga game selesai. Secara umum sudut pandang visual novel biasanya dilihat dari sudut pandang orang pertama, dan beberapa *event* kejadian dalam skenario selalu dilihat dari sudut pandang satu karakter, karena itulah biasanya sang protagonist jarang digambarkan [ANO-09]. *Visual novel* umumnya memiliki karakteristik sebuah dialog box dan latar belakang serta sprite karakter seperti terlihat pada gambar 2.1

Gambar 2.1 Contoh *Visual Novel*.

Sumber : [ANO-09]

Kebanyakan visual novel memiliki jalan cerita yang bercabang dan ending yang bermacam-macam. Mekanisme gameplay dalam hal ini adalah menyajikan pemilihan opsi, dimana player memilih pilihan yang akan menentukan jalan cerita ataupun endingnya. Style gameplay visual novel ini hampir mirip dengan buku "pilih sendiri petualanganmu". Walaupun demikian, tidak ada salahnya jika pembuat cerita membuat sebuah visual novel yang memiliki alur cerita linear tanpa adanya percabangan cerita.

2.2 Java 2 Standard Edition (J2SE)

Java 2 Standard Edition (J2SE) adalah inti dari bahasa pemrograman Java. Merupakan Platform Java yang digunakan untuk membuat aplikasi umum. Java SE mengandung sebuah *virtual machine* yang digunakan untuk menjalankan program Java demikian pula dengan set library (atau "*packages"*) dibutuhkan untuk penggunaan *file systems*, networks, grafik interface, dan sebagainya dari program tersebut. JDK (Java Development Kit) adalah salah satu tool dari J2SE untuk mengkompilasi dan menjalankan program Java dan JRE (Java Runtime Environment).

2.3 Java 2 Micro Edition (J2ME)

Java 2 Micro Edition (J2ME) merupakan subset dari J2SE yang ditujukan untuk implementasi pada peralatan embedded system dan handheld yang tidak mampu mendukung secara penuh implementasi menggunakan J2SE. Embedded system adalah produk-produk dengan komputer kecil berada di dalamnya, namun aplikasi yang bisa dimanfaatkan dari peralatan tersebut sangatlah spesifik. Hal ini tentu saja berbeda dengan komputer PC yang kita kenal sehari-hari, yang mampu digunakan untuk berbagai aplikasi. Contoh Embedded system yang ada misalnya adalah aplikasi-aplikasi yang memanfatkan mikroprosesor seperti televisi, sistem keamanan gedung, dan lain sebagainya.

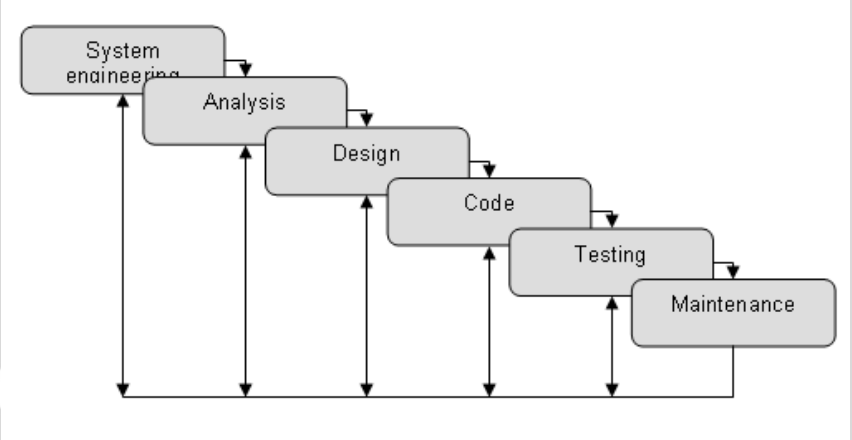
J2ME sangat berguna untuk membangun sebuah aplikasi pada peralatan dengan jumlah memori dan kapasitas pemyimpan yang terbatas, serta kemampuan user interface yang terbatas seperti pada perangkat komunikasi bergerak berupa telepon genggam, PDA, dan sebagainya.

Seperti aplikasi java umumnya yang menggunakan Java Virtual Machine (JVM), dalam J2ME digunakan pula Virtual Machine yang disebut K Virtual Machine.

K Virtual Machine adalah virtual machine yang sangat kecil dalam kebutuhan memorinya. Huruf K dalam K Virtual Machine adalah singkatan dari Kilobyte, untuk menggambarkan betapa virtual machine ini bekerja pada total memori yang sedemikian kecil mulai dari 128 kilobyte hingga maksimal rata-rata sekitar 512 kilobyte [ADI-03]..

2.4 Rekayasa Perangkat Lunak

Dalam melakukan proses pengembangan suatu perangkat lunak, diperlukan suatu metode yang digunakan sebagai panduan untuk menghasilkan sebuah perangkat lunak yang benar dan berkualitas. Sehingga perangkat lunak yang dihasilkan bisa menguntungkan pihak *user* maupun *developer*. *User* bisa mendapatkan perangkat lunak yang sesuai dengan permintaan dan kebutuhan yang diinginkan, dan di pihak *developer* dapat menyelesaikan sebuah perangkat lunak dengan terjadwal, serta mendapatkan kemudahan dalam melakukan *maintenance* (pemeluharaan). Dalam hal, inilah urgensi dari sebuah rekayasa terhadap perangkat lunak *(Software Engineering). Software engineering* merupakan teknologi yang meliputi proses, sekumpulan metoda dan sederetan alat bantu untuk pengembangan perangkat lunak.



Gambar 2.2 Siklus pengembangan perangkat lunak dengan model *waterfall*

Sumber : [FSO-05]

Pada Gambar 2.2. terlihat siklus pengembangan suatu sistem perangkat lunak. Proses pengembangan perangkat lunak akan selalu melakukan proses pengumpulan terhadap kebutuhan *(requirement)* dari *user* (pengguna) atau *customer* (pelanggan), memahami dan menetapkan kebutuhan-kebutuhan tersebut. Langkah ini menjadi suatu pekerjaan yang sangat penting. Tahap yang selanjutnya adalah melakukan analisis

terhadap kebutuhan dilanjutkan dengan perancangan sistem perangkat lunak yang akan dibangun. Hasil dari perancangan dijadikan acuan dalam proses *coding* untuk merealisasikannya ke dalam bentuk yang bisa diterjemahkan oleh mesin. Proses selanjutnya adalah pengujian terhadap perangkat lunak yang telah dihasilkan.

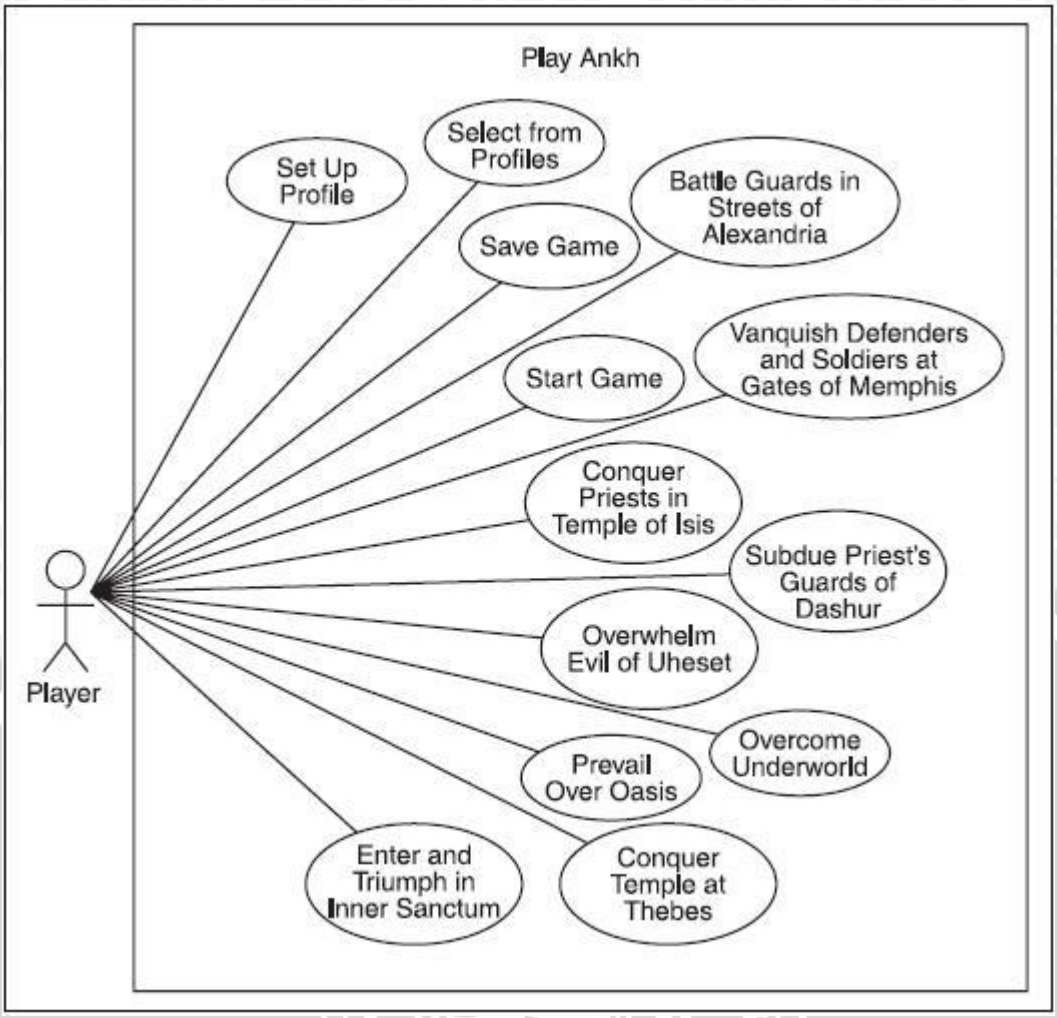
Terdapat dua klasifikasi metode pengembangan perangkat lunak, yaitu metode struktural dan metode berorientasi pada objek. Pada skripsi ini digunakan metode pengembangan perangkat lunak berorientasi objek.

2.4.1 Analisis Kebutuhan (Requirement Analysis)

Metode analisis yang digunakan dalam mengembangkan perangkat lunak pada skripsi ini adalah metode analisis berorientasi objek *(Object Oriented Analysis).* Sasaran dari *Object Oriented Analysis (OOA)* adalah mengembangkan sederetan model yang menggambarkan perangkat lunak komputer pada saat perangkat itu bekerja untuk memenuhi serangkaian persyaratan yang ditentukan oleh pelanggan.

Proses analisis ini mengambil acuan dari hasil pengumpulan, pemahaman dan penetapan kebutuhan-kebutuhan *(requirements)* fungsional dan non fungsional yang ingin didapatkan oleh pengguna, dan harus mempu disediakan oleh sistem perangkat lunak. Pada skripsi ini, digunakan diagram *use case* untuk memodelkan hasil analisis terhadap kebutuhan fungsional.

Diagram *use case* merupakan salah satu pemodelan diagram yang menunjukkan hubungan antara sistem dengan aktor. Aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case* digunakan untuk lebih memahami tentang *event-evant* yang mungkin muncul, hubungan antar komponen system, dan memastikan kita sudah melengkapi analisis *requirement* yang dibutuhkan [FSO-05]. Gambar 2.3 menunjukkan contoh dari diagram *use case*.



Gambar 2.3 Contoh diagram use case *game*

Sumber : [FSO-05]

Secara lebih mendetail, masing-masing *use case* yang terdapat dalam diagram *use case*, dijabarkan dalam skenario *use case*. Di dalam skenario *use case*, akan diberikan uraian nama *use case*, aktor yang berhubungan dengan *use case* tersebut, tujuan dari *use case*, dekripsi global tentang *use case*, pra-kondisi yang harus dipenuhi dan post-kondisi yang diharapkan setelah berjalannya fungsional *use case*. Berikut ini adalah contoh skenario *use case*.

Tabel 2.1 Contoh Skenario use case game

Use Case Name: Player Starts Game
Requirement(s) Explored: 1
Player (Actor) Context (Role): Player
Precondition(s): Game is installed and ready to play.
Trigger(s): Player selects game-start action.
Main Course of Action: 1. The player selects the game-start action. 2. The system requests that the player select a profile. 3. The player selects a profile. 4. The system loads the profile. 5. The system displays a message to the player that tells him that play can begin. 6. The player acknowledges the message. 7. The player begins playing.
Alternate Course(s) of Action: 3a. The player does not select a profile. 3b. The system chooses a default profile. 6a. The player chooses to exit the game before playing. 6b. The system asks the player if he wants to save his profile. 6b1. If the player responds yes, the system saves the profile. 6b2. If the player responds no, the system exits without saving the profile. 6c. The system saves the profile. 6c1. The player chooses not to save the profile. 6d. The system exits.
Exceptional Course(s) of Action: 6a. The system crashes when the game begins. 6b. The player turns off the computer.

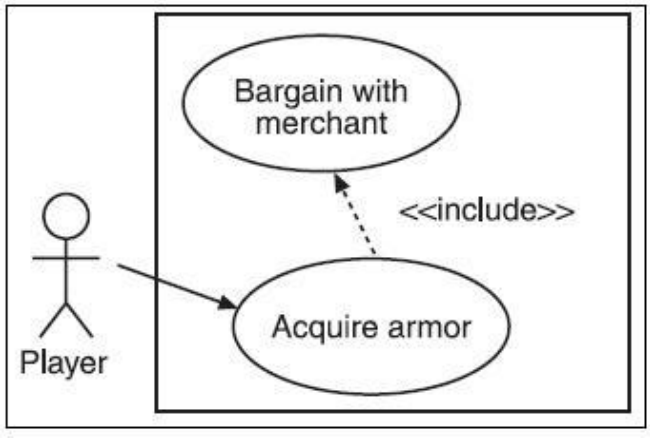
Sumber : [FSO-05]

Relasi antara aktor dan *use case* adalah relasi asosiasi yang dimodelkan dengan anak panah. Setiap *use case* harus diinisialisasi oleh aktor, kecuali pada relasi *extend* dan *include.* Relasi *include* ditunjukkan seperti pada Gambar 2.4. Relasi *include* memungkinkan satu *use case* menggunakan fungsionalitas yang disediakan oleh *use case* yang lainnya. Relasi ini dapat digunakan dengan alas an salah satu dari dua hal berikut:

1. Jika dua atau lebih *use case* mempunyai bagian besar fungsionalitas yang identik, maka fungsionalitas ini dapat dipecah ke dalam *use case* tersendiri. Masing-masing *use case* kemudian menggunakan relasi *include* terhadap *use case* yang baru dapat dibuat tersebut.
2. Relasi *include* bermanfaat untuk situasi jika sebuah *use case* mempunyai fungsionalitas yang terlalu besar, kemudian fungsionalitas yang besar tersebut

dipecah menjadi dua buah *use case* yang lebih kecil, relasi *include* digunakan menghubungkan dua buah *use case* hasil pecahan.

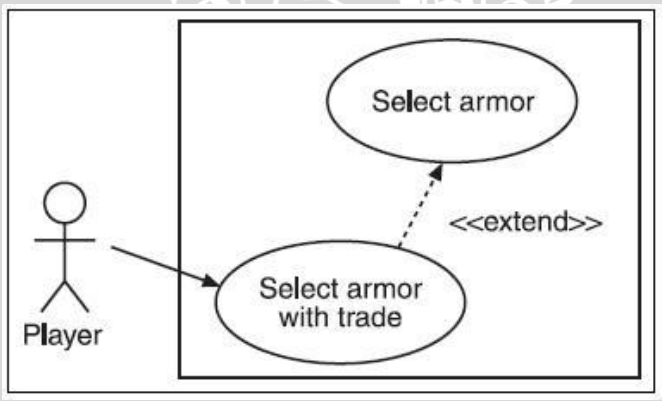
Relasi *include* menyatakan bahwa satu *use case* selalu menggunakan fungsionalitas yang disediakan oleh *use case* lainnya. Pada Gambar 2.4, *use case* "Bargain with Merchant" pasti akan dijalankan setelah *use case* "Acquire Armor" selesai berjalan.



Gambar 2.4 Contoh pemodelan relasi *include* antara 2 *use case*

Sumber : [FSO-05]

Relasi *extend* memungkinkan satu *use case* secara opsional menggunakan fungsionalitas yang disediakan oleh *use case* lainnya. Dalam UML relasi *extend* digambarkan pada Gambar 2.5.



Gambar 2.5 Contoh pemodelan relasi *extend* antara 2 *use case*

Sumber : [FSO-05]

Dari Gambar 2.5, *use case* "Select armor with trade" *extend* terhadap *use case* "Select armor". Ketika *use case* "Select armor" sedang berjalan, *use case* "Select armor with trade" berjalan jika dan hanya jika diinginkan oleh aktor. Jika tidak diinginkan, maka *use case* "select armor with trade" tidak akan pernah dijalankan.

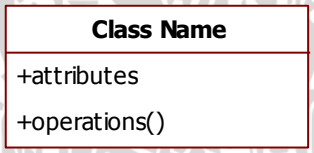
2.4.2 Perancangan (Design)

Perancangan berorientasi objek (*Object Oriented Design/OOD)* berhubungan dengan pengembangan model berorientasi objek dari sistem perangkat lunak untuk mengimplementasikan persyaratan atau analisis kebutuhan (OOA) yang telah diidentifikasi. Perancangan dilakukan dengan cara mengolah informasi yang telah diperoleh dari analisis kebutuhan menjadi diagram dan alat perancangan lainnya yang menuntun pengembang untuk membangun perangkat lunak yang sesuai dengan kebutuhan-kebutuhan yang telah diidentifikasi sebelumnya [FSO-05].

Pada tahap perancangan di skripsi ini, digunakan pemodelan dengan menggunakan dua macam diagram, yaitu *class diagram dan sequence diagram.*

2.4.2.1 Diagram Kelas *(Class Diagram)*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Gambar 2.6 menunjukkan representasi suatu kelas.

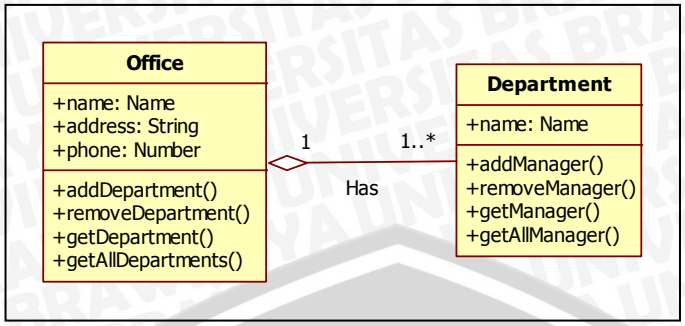


Gambar 2.6 Representasi kelas
Sumber : [FSO-05]

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain. *Class diagram* memiliki tiga area pokok, yaitu nama (dan stereotype), atribut dan metoda. Atribut dan metoda dapat memiliki salah satu sifat berikut:

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
- *Public*, dapat dipanggil oleh siapa saja

Dalam diagram kelas *multiplicity* digunakan untuk mengindikasikan berapa jumlah objek sebuah kelas mempunyai relasi dengan objek tunggal di kelas lainnya pada satu waktu.

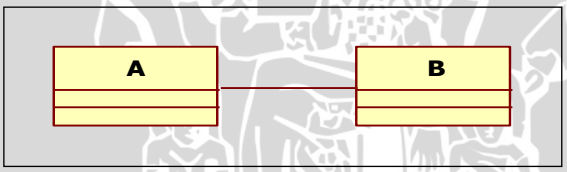


Gambar 2.7 Contoh diagram kelas

Sumber : [FSO-05]

Untuk memodelkan sebuah sistem yang utuh dibutuhkan relasi antar kelas. Relasi yang digunakan dalam diagram kelas dapat dikelasifikasikan menjadi:

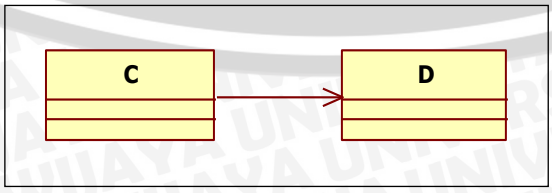
- **Assosiasi (*Association*)** adalah relasi yang saling terkait diantara kelas. Pada diagram UML disimbolkan dengan sebuah garis lurus. Pada saat relasi assosiasi digunakan maka antara kelas dapat mengirim pesan kepada kelas lain dalam *sequence diagram* atau diagram kolaborasi. Assosiasi memungkinkan sebuah kelas untuk mengetahui attribut dan operasi yang memiliki sifat *public*. Assosiasi dapat berupa dua arah (*bidirectional*) atau searah (*unidirectional*).



Gambar 2.8 Relasi assosiasi dua arah

Sumber : [FSO-05]

Gambar 2.8 menunjukkan kelas A mempunyai relasi assosiasi dua arah dengan kelas B, yang berarti kelas A dapat mengirimkan pesan kepada kelas B begitu juga sebaliknya. Kelas A dapat mengetahui attribut dan operasi yang dideklarasikan secara *public* di kelas B, begitu juga sebaliknya.

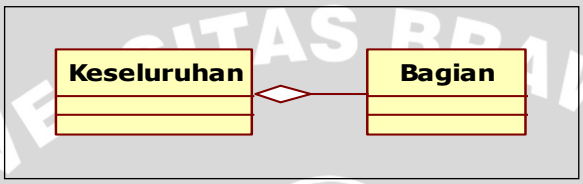


Gambar 2.9 Relasi assosiasi satu arah

Sumber : [FSO-05]

Gambar 2.9 menunjukkan kelas C mempunyai relasi assosiasi satu arah dengan kelas D, yang berarti kelas C dapat mengetahui attribut dan operasi yang dideklarasikan secara *public* oleh kelas D, tetapi tidak sebaliknya. Kelas C dapat mengirimkan pesan kepada kelas D, tetapi tidak sebaliknya.

- **Aggregasi (*Aggregation*)** adalah relasi antara "keseluruhan" dengan "bagian", sebagai contoh kita memiliki kelas Mesin, kelas Roda, dan kelas lainnya yang dibutuhkan oleh kelas Mobil. Sebuah mobil akan terbentuk oleh sebuah mesin, empat roda, dan lainnya. Dalam UML aggregasi disimbolkan dengan notasi seperti pada Gambar 2.10.

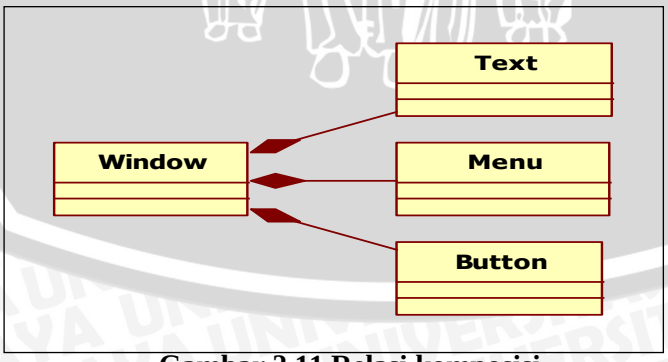


Gambar 2.10 Relasi aggregasi

Sumber : [FSO-05]

Gambar 2.10 menunjukkan kelas Bagian merupakan "bagian" dari kelas Keseluruhan.

- **Komposisi (*Composition*)** adalah relasi antara "keseluruhan" dan "bagian" dimana "bagian" hanya merupakan bagian dari sebuah "keseluruhan" dan "keseluruhan" bertanggung jawab dalam membuat dan memusnahkan "bagian". Apabila salah satu "keseluruhan" dihilangkan maka "bagian" akan hilang pula. Dalam UML komposisi disimbolkan dengan notasi seperti pada Gambar 2.11.

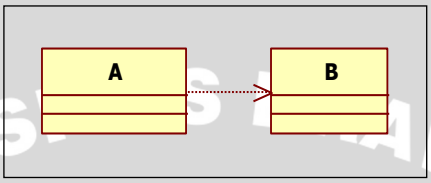


Gambar 2.11 Relasi komposisi

Sumber : [FSO-05]

Gambar 2.11 menunjukkan kelas *Text*, *Menu* dan *Button* merupakan "bagian" dari kelas *Window*, apabila kelas *Window* dihilangkan maka kelas *Text*, *Menu* dan *Button* akan hilang pula.

- **Dependensi (*Dependency*)** adalah relasi yang selalu searah (*unidirectional*) dan menunjukkan bahwa suatu kelas bukan instansiasi (sebagai variabel instan) dari kelas lain. Relasi ini dipakai ketika suatu kelas diperlukan sebagai parameter atau nilai balik dalam operasi suatu kelas tertentu. Dalam UML dependensi disimbolkan dengan anak panah dengan garis putus-putus seperti pada Gambar 2.12.

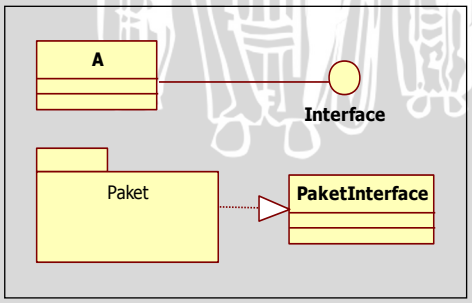


Gambar 2.12 Relasi dependensi

Sumber : [FSO-05]

Gambar 2.12 menunjukkan bahwa kelas A dependen terhadap kelas B yang berarti kelas B berfungsi sebagai parameter atau nilai balik dari kelas A.

- **Realisasi (*Realization*)** adalah relasi yang digunakan untuk menghubungkan kelas dengan *interface*-nya, paket dengan *interface*-nya, suatu komponen dengan *interface*-nya atau antara sebuah *use case* dengan *use case realization*. Relasi ini berfungsi untuk menunjukkan sebuah *interface* dan implementasinya. Dalam UML realisasi disimbolkan dengan notasi seperti pada Gambar 2.13.

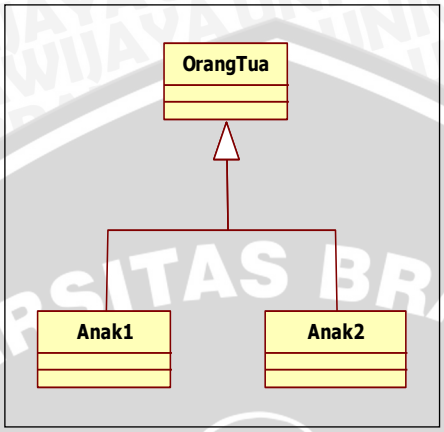


Gambar 2.13 Relasi realisasi]

Sumber : [FSO-05]

Gambar 2.13 menunjukkan bahwa Interface merupakan implementasi *interface* dari kelas A dan PaketInterface merupakan kelas *interface* dari *package* Paket.

- **Generalisasi (*Generalization*)** adalah relasi yang digunakan untuk memperlihatkan hubungan pewarisan (*inheritance*) antara dua model elemen (aktor, *use case*, kelas atau paket). Pewarisan memperbolehkan suatu kelas untuk mewarisi semua atau sebagian attribut, operasi, relasi dan elemen lain yang terkait. Dalam UML generalisasi digambarkan seperti pada Gambar 2.14.



Gambar 2.14 Relasi generalisasi

Sumber : [FSO-05]

Gambar 2.14 menunjukkan bahwa kelas Anak1 dan Anak2 merupakan kelas warisan dari kelas OrangTua yang berarti bahwa semua atau sebagian atribut, operasi yang didefinisikan secara *public* atau *protected* dan relasi dari kelas OrangTua dimiliki oleh kelas Anak1 dan Anak2.

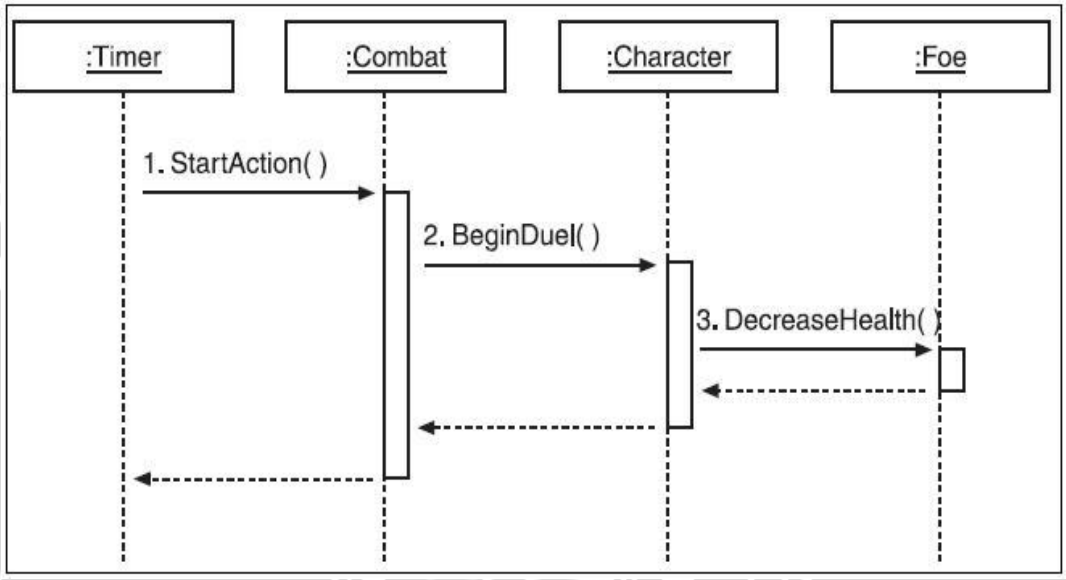
2.4.2.2 Diagram Sekuen *(Sequence diagram)*

Sequence diagram adalah sebuah diagram objek yang dinamis dan menjelaskan alur pesan dari objek ke objek. Supaya alurnya lebih mudah dimengerti, *sequence diagram* membagi sifat objek dalam dua arah, yakni vertikal dan horizontal. Garis vertikal menunjukkan *lifeline* dari sebuah objek, dan garis horizontal menunjukkan pesan antar objek.

Sequence diagram secara khusus dimiliki oleh elemen-elemen yang menyusun sebuah sistem. Sebagai contoh, sebuah *sequence diagram* yang dihubungkan dengan sebuah subsistem yang menunjukkan bagaimana subsistem tersebut merealisasikan sebuah fungsi yang disediakan untuk pengguna.

Diagram sekuensial dapat dibaca dengan memperhatikan objek-objek dan pesan-pesan yang ada pada diagram tersebut. Objek yang terlibat dalam aliran ditunjukkan dengan bujur sangkar yang ada di bagian atas diagram. Masing-masing objek mempunyai *lifeline* yang digambarkan dengan garis putus-putus secara vertical di

bawah objek. *Lifeline* dimulai saat objek diinstansiasi, dan berakhir pada saat objek dimusnahkan. Sebuah pesan digambarkan antara *lifeline* dari dua objek untuk menunjukkan dua objek tersebut berkomunikasi. Setiap pesan menggambarkan satu objek memanggil fungsi tertentu (fungsi panggil) di objek lainnya. Pesan-pesan ini kemudian dapat didefinisikan sebagai operasi untuk sebuah kelas, setiap pesan dapat menjadi sebuah operasi. Pesan dapat refleksif (terhadap dirinya sendiri), menunjukkan bahwa sebuah obyek memanggil sebuah operasi di dirinya. Gambar 2.15 memberikan contoh ilustrasi *sequence diagram.*



Gambar 2.15 Contoh diagram sekuensial

Sumber : [FSO-05]

Pada Gambar 2.15 menunjukkan contoh *sequence diagram.* Bentuk kotak vertikal melambangkan *activations,* yang mengindikasikan kapan operasi tersebut aktif. Panah dari kiri ke kanan melambangkan pesan satu arah, dan biasanya membutuhkan suatu tipe pesan balik (*return*). Garis panah yang putus-putus melambangkan *return value.*

Sequence diagram di atas dapat dibaca sebagai berikut, timer akan memicu start action, AI akan memerintahkan karakter untuk menyerang, dan karakter akan menyerang musuh. Saat musuh terkena serangan, dia akan mengembalikan suatu nilai (*return value*), begitu juga dengan karakter dan AI. Saat timer menerima nilai kembalian tersebut, sistem akan keluar.

2.4.3 Pengujian (*Testing*)

Pengujian pada perangkat lunak merupakan elemen yang sangat penting untuk menjamin kualitas perangkat lunak. Pengujian dilakukan untuk menemukan kelemahan atau kesalahan yang mungkin terdapat pada perangkat lunak yang dibangun. Secara umum, kesalahan pada sebuah perangkat lunak biasanya terbagi dalam dua macam. Pertama, kesalahan yang muncul karena perangkat lunak tidak dibangun sesuai dengan spesifikasi. Kedua, kesalahan yang disebabkan munculnya hal-hal yang di luar dugaan. Untuk memperbaiki kesalahan-kesalahan tersebut, perangkat lunak harus melalui berbagai macam pengujian yang meliputi komponen, integrasi komponen, ataupun sistem secara keseluruhan [FSO-05].

2.4.3.1 Teknik Pengujian

Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu singkat dan usaha minimum. Metode perancangan kasus uji menyediakan mekanisme penting yang dapat membantu menjamin kompleksitas pengujian dan keandalan yang tinggi untuk mengatasi kesalahan. Perangkat lunak dapat diuji melalui dua cara yaitu *white box testing* dan *black box testing* [FSO-05].

2.4.3.1.1*White Box Testing*

Pengujian *white box* suatu perangkat lunak didasarkan pada pengamatan yang teliti terhadap detail procedural. Jalur-jalur logika yang melewati perangkat lunak diuji dengan memberikan *test case* yang manguji serangkaian kondisi dan atau *loop* tertentu. Ada beberapa metode untuk merealisasikan pengujian *white box,* diantaranya adalah *basis path testing, condition testing, loop tesing* serta *data flow testing.*

Pada skripsi ini, metode yang digunakan untuk merealisasikan *white box testing* adalah teknik *basis path testing.* Metode *basis path* ini memungkinkan perancang *test case* mengukur kompleksitas logis dari desain prosedural dan *basis set* dari jalur eksekusi. *Test case* yang dilakukan untuk menggunakan *basis set* tersebut dijamin untuk menggunakan setiap pernyataan di dalam program paling tidak sekali selama pengujian. Untuk melakukan teknik pengujian ini diperlukan penelusuran terhadap kontrol logika untuk menentukan *test case* dalam proses pengujian.

2.4.3.1.2*Black Box Testing*

Pengujian *black box* fokus pada persyaratan-persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black box* memungkinkan perekayasa perangkat

lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut:

1. Fungsi-fungsi yang tidak benar (hilang)
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses ke *database* eksternal.
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

Tidak seperti teknik *white box testing*, yang dilakukan pada awal proses awal pengujian, *black box testing* dilakukan pada akhir proses pengujian. Karena *black box testing* ditujukan untuk mengabaikan struktur kontrol, perhatian difokuskan pada domain informasi.

Beberapa teknik yang dipakai untuk melakukan pengujian *Black Box* diantaranya adalah sebagai berikut:

a. Metode Pengujian Partisi Ekivalensi

Partisi ekivalensi adalah metode pengujian *black box* yang membagi domain input dari suatu program ke dalam kelas data dari mana *test case* dapat dilakukan. *Test case* yang ideal mengungkap kesalahan (misalnya, pemrosesan yang tidak benar terhadap semua data karakter) yang akan memerlukan banyak kasus untuk dieksekusi sebelum kesalahan umum diamati. Partisi ekivalensi berusaha menentukan sebuah *test case* yang mengungkap kelas-kelas kesalahan, sehingga mengurangi jumlah total *test case* yang harus dikembangkan.

Desain *test case* pada partisi ekivalensi didasarkan pada evaluasi terhadap kelas ekivalensi untuk suatu kondisi input. Kelas ekivalensi merepresentasikan serangkaian keadaan valid atau yang tidak valid untuk kondisi input. Secara khusus, suatu kondisi input dapat berupa harga numeris, suatu rentang harga, atau serangkaian harga terkait, atau sebuah kondisi Boolean.

Kelas ekivalensi dapat ditentukan sesuai pedoman berikut ini:

1. Bila kondisi input menentuka suatu *range* maka suatu kelas ekivalensi valid dan dua yang tidak valid ditentukan.
2. Bila suatu kondisi input membutuhkan suatu harga khusus maka satu kelas ekivalensi valid dan dua yang tidak valid ditentukan.
3. Bila suatu kondisi menentukan anggota suatu himpunan, maka satu kelas ekivalensi valid atau dua yang tidak valid ditentukan.

4. Bila suatu kondisi input adalah Boolean, maka satu kelas valid dan satu yang tidak valid ditentukan.

Yang pertama kali dilakukan adalah identifikasi kondisi input dari suatu sistem. Kemudian dengan mengaplikasikan pedoman untuk derivasi, *test case* untuk masing-masing item data domain input dapat dikembangkan dan dieksekusi.

b. Metode Analisis Nilai Batas *(Boundary Value Analysis)*

Teknik pengujian *Boundary Value Analysis (BVA)* dikembangkan untuk memunculkan pemilihan *test case* yang menggunakan nilai batas. Teknik ini melengkapi pengujian partisi ekivalensi. BVA lebih mengarahkan kepada pemilihan *test case* pada *edge* dari kelas. Dalam banyak hal, pedoman untuk BVA sama dengan yang diberikan untuk partisi ekivalensi:

1. Bila suatu kondisi input mengkhususkan suatu *range* dibatasi oleh nilai *a* dan *b* , maka *test case* harus didesain dengan nilai *a* dan *b*, persis di atas dan di bawah *a* dan *b*, secara bersesuaian.
2. Bila suatu kondisi input mengkhususkan sejumlah nilai, maka *test case* harus dikembangkan dengan menggunakan jumlah minimum dan maksimum. Nilai tepat di atas dan di bawah minimum dan maksimum juga diuji.
3. Pedoman 1 dan 2 diaplikasikan ke kondisi output.

Bila struktur data program telah memesan batasan (misalnya, suatu *array* memiliki suatu batas yang ditentukan dari 100 masukan), maka *case* didesain untuk struktur data tersebut pada batasnya.

2.4.3.2 Strategi Pengujian

Strategi untuk pengujian perangkat lunak merupakan aktifitas mengintegrasikan metode perancangan kasus uji ke dalam sebuah rangkaian langkah-langkah pengujian yang terencana sehingga menghasilkan perangkat lunak yang baik. Strategi menyediakan urutan langkah yang harus dilakukan sebagai bagian dari pengujian. Setiap langkah direncanakan kemudian dikerjakan dan ditentukan berapa banyak usaha, waktu dan sumber daya yang dibutuhkan. Strategi untuk melakukan pengujian perangkat lunak, dimulai dari dengan “pengujian kecil” bergerak menuju ke “pengujian besar”. Pengujian berorientasi objek akan memulai pengujian dari *unit testing*, bergerak menuju *integration testing* dan berakhir pada *validation testing* [FSO-05].

2.4.3.2.1Pengujian Unit (*Unit Testing)*

Unit testing berfokus pada usaha verifikasi pada unit terkecil dari perancangan perangkat lunak yaitu pada komponen atau modul. *Unit testing* berorientasi *white box* dan setiap langkah dapat dikerjakan secara paralel untuk berbagai komponen. *Unit testing* pada suatu modul dilakukan dengan menguji *interface*, struktur data lokal, kondisi batas, jalur independen dan jalur penanganan kesalahan.

2.4.3.2.2Pengujian Integrasi (*Integration Testing*)

Pengujian integrasi ini mengambil input kelas yang lolos pada pengujian unit, kemudian kelas yang lolos pada pengujian unit diintegrasikan untuk membentuk agregat yang lebih besar dan diujikan dengan mempersiapkan output hasilnya untuk agregat yang lebih besar lagi. Pengujian ini dilakukan untuk memastikan apakah sistem berjalan berdasarkan spesifikasi sistem. Pengujian integrasi ini menekankan pada masalah-masalah yang berhubungan dengan verifikasi dan konstruksi program. Teknik pengujian *black box* adalah yang paling lazim selama integrasi, meskipun sedikit pengujian *white box* dapat digunakan.

2.4.3.2.3Pengujian Validasi (*Validation Testing*)

Validation testing merupakan pengujian yang dilakukan pada perangkat lunak secara keseluruhan untuk memastikan bahwa perangkat lunak telah memenuhi seluruh kebutuhan (*requirement*) yang telah ditentukan. *Validation testing* lebih banyak menggunakan teknik *black box testing*. Pengujian dilakukan agar dapat menjamin semua kebutuhan fungsional, karakteristik perilaku dan kebutuhan performansi telah dicapai serta dokumentasi yang benar.

BAB III

METODOLOGI PENELITIAN

Tugas akhir ini disusun berdasarkan kebutuhan sistem aplikasi *Visal Novel Maker* dan *visual novel* yang bergrafis 2D. Sesuai dengan acuan pada rumusan masalah yang telah diuraikan.

Tahap-tahap yang dilakukan untuk membangun sistem tersebut adalah sebagai berikut :

3.1 Studi Literatur

Berupa kajian pustaka terhadap sumber-sumber bacaan yang relevan yang dapat menunjang dalam proses pengembangan sistem. Kajian terhadap berbagai literatur ini ditujukan untuk mendapatkan landasan teori yang diperlukan, landasan teori yang dimaksud antara lain yaitu :

a. Visual Novel
b. Teknologi Java
   - *Java 2 Micro Editon* (J2ME)
   - *Java 2 Standard Edition* (J2SE)
c. Metode Pengembangan Perangkat Lunak
   - Analisis Kebutuhan
   - Perancangan
   - Pengujian

Sumber bacaan tersebut dapat berupa *text book*, buku panduan pemrograman, tugas akhir, *paper*, tutorial pemrograman, dan sumber bacaan *softcopy* lain yang didapatkan dari i*nternet*. Sumber-sumber bacaan tersebut diletakkan pada daftar pustaka.

3.2 Analisis Kebutuhan (Requirement Analysis)

Tahap ini ditujukan untuk mendapatkan kebutuhan apa saja yang diperlukan dan yang ingin didapatkan dari sistem yang akan dibangun, dan kemudian menganalisisnya. Metode analisis yang digunakan adalah *Object Orieneted Analysis* dengan menggunakan bahasa pemodelan UML *(Unified*

Modeling Language). Pada tahap ini dibuat *Use Case Diagram* yang mendeskripsikan fungsionalitas sistem dari perspektif *user*.

3.3 Perancangan

Metode yang digunakan untuk perancangan adalah *Object Oriented Design (OOD).* Pada proses desain dilakukan identifikasi terhadap klas-klas yang dibutuhkan yang dimodelkan dalam *Class Diagram*. Hubungan interaksi antar elemen (objek) yang telah diidentifikasi, dimodelkan dalam *Sequence Diagram* yang menggambarkan interaksi antar objek yang disusun dalam urutan waktu.

3.4 Implementasi

Tahap ini bertujuan untuk mengimplemantasikan perancangan sistem yang telah dilakukan pada tahap sebelumnya agar menjadi sebuah sistem nyata yang dapat digunakan. Hal ini dilakukan dengan meng-kodekan rancangan sesuai dengan metode implementasi yang digunakan. Pada tugas akhir ini, metode implementasi yang digunakan adalah Pemrograman Berorientasi Objek (*Object Oriented Programming* / OOP) dengan menggunakan bahasa pemrograman *Java 2 Standard Edition* (J2SE) dan *Java 2 Micro Edition* (J2ME).

3.5 Pengujian dan Analisis

Pengujian perangkat lunak pada skripsi ini dilakukan agar dapat menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang melandasinya.

Strategi pengujian yang digunakan yaitu pengujian unit, pengujian integrasi, dan pengujian validasi. Dan metode pengujian yang digunakan adalah metode pengujian *white box* dan *black box*. Pengujian dimulai dari pengujian unit, kemudian dilanjutkan dengan pengujian integrasi, dan berakhir pada pengujian validasi. Pada tahap pengujian unit digunakan metode pengujian *white box* dengan teknik *basis path*, sedangkan pada pengujian integrasi digunakan teknik *white box,* dilakukan dengan mengambil input klas yang lolos pada pengujian unit, kemudian klas yang lolos pada pengujian unit diintegrasikan melalui klas kontrol untuk diujikan. Pada tahap pengujian validasi digunakan teknik *black-box*.

3.6 Pengambilan Kesimpulan dan Saran

Pada tahap ini diambil kesimpulan dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahap yang paling akhir adalah pengambilan saran terhadap penelitian yang selanjutnya sehingga bisa menyempurnakan kekurangan yang ada dan mengembangkan pada tingkat pokok kajian yang lebih lanjut.

BAB IV

PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai perancangan perangkat lunak, yang meliputi dua tahap. Pada tahap pertama dilakukan proses analisis kebutuhan (*requirement analysis*), sedangkan tahap yang kedua adalah proses perancangan. Pada tahap analisis kebutuhan digunakan pemodelan dengan diagram *use case.* Pada proses perancangan perangkat lunak digunakan pemodelan dalam bentuk diagram klas (*class diagram*), dan diagram urutan (*sequence diagram*).

4.1 Analisis Kebutuhan

Proses analisis ini mengambil acuan dari hasil pengumpulan, pemahaman dan penetapan kebutuhan-kebutuhan (*requirements*) yang ingin didapatkan oleh pengguna, dan harus mampu disediakan oleh perangkat lunak. Pada analisis kebutuhan ini diawali dengan identifikasi aktor-aktor, penjabaran daftar kebutuhan dan kemudian memodelkannya ke dalam suatu diagram *use case.* Analisa kebutuhan ditujukan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memecahkan permasalahan yang dihadapi oleh pengguna.

4.1.1 Identifikasi Aktor

Tahap ini mempunyai tujuan untuk melakukan identifikasi terhadap aktor-aktor yang akan berinteraksi dengan sistem. Tabel 4.1 memperlihatkan dua buah aktor beserta penjelasannya masing-masing yang merupakan hasil dari proses identifikasi aktor.

Tabel 4.1 Deskripsi Aktor

Aktor	**Deskripsi Aktor**
User/Pengguna	*User* adalah aktor yang menggunakan sistem untuk mengedit, menulis, dan menggenerate script pada aplikasi.

Sumber : Analisis Kebutuhan

4.1.2 Daftar Kebutuhan

Daftar kebutuhan ini terdiri dari sebuah kolom yang menguraikan kebutuhan yang harus disediakan oleh sistem, dan pada kolom yang lain akan menunjukkan nama *use case* yang menyediakan fungsionalitas masing-masing kebutuhan tersebut. Daftar

kebutuhan fungsional dan non fungsionalsistem yang dikembangkan pada tugas akhir ini ditunjukkan pada Tabel 4.2.

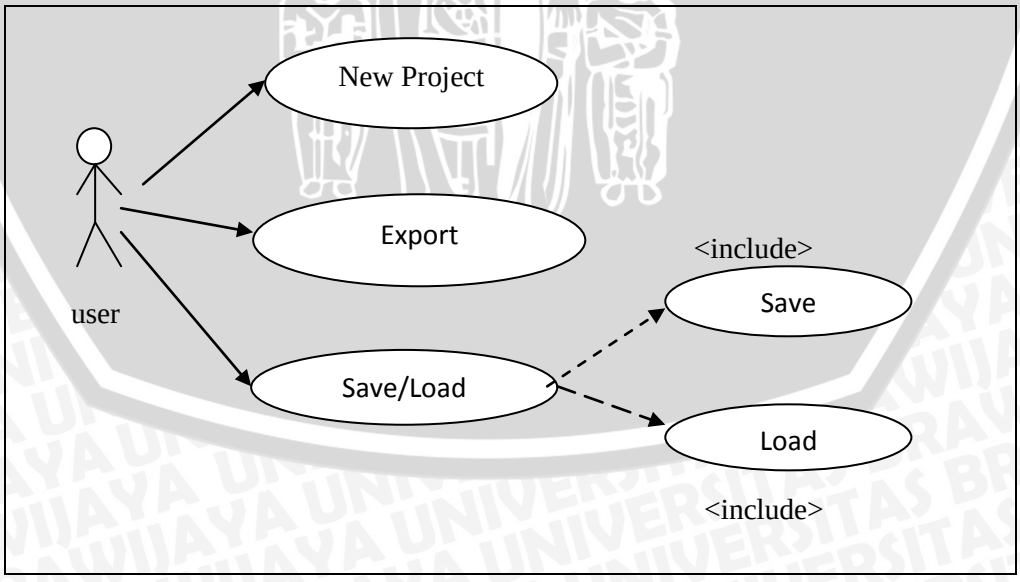
Tabel 4.2 Daftar kebutuhan fungsional dan non fungsional

ID	Requirements	Nama *Use Case*
F01	Sistem harus menyediakan fasilitas pembuatan proyek baru bagi *User*.	New Project
F02	Sistem menyediakan fasilitas untuk menyimpan dan membaca proyek yang telah disimpan.	Save/Load Project
F03	Sistem dapat menggenerasikan file-file yang diinput menjadi script java2me dan dikompresi menjadi file .jar	Export Project
N01	Sistem harus bias dijalankan pada desktop PC	-
N02	Output yang dihasilkan sistem harus dapat direalisasikan pada perangkat *java mobile phone* yang sebenarnya (bukan *emulator*)	-

Sumber : Analisis Kebutuhan

4.1.3 Diagram Use Case

Kebutuhan-kebutuhan fungsional yang diperlukan oleh *user* dan harus disediakan oleh sistem akan dimodelkan pada diagram *use case*. Pada sistem ini terdapat empat buah *use case*, yaitu *use case* new project, save/load, dan Generate dengan sebuah aktor yaitu *user*. Gambar 4.1 merupakan diagram *use case* dari sistem ini.



Gambar 4.1 Diagram use case system

Sumber : Analisis Kebutuhan

4.1.4 Skenario use case

Secara lebih mendetail, masing-masing *use case* yang terdapat pada diagram *use case*, dijabarkan dalam skenario *use case.* Di dalam skenario *use case*, akan diberikan uraian nama *use case*, aktor yang berhubungan dengan *use case* tersebut, tujuan dari *use case*, deskripsi global tentang *use case*, pra-kondisi yang harus dipenuhi dan post-kondisi yang diharapkan setelah berjalannya fungsional *use case*. Selain iitu juga diberikan ulasan yang berkaitan dengan tanggapan dari sistem atas suatu aksi yang diberikan oleh aktor (aliran utama), serta kejadian alternatif yang akan terjadi jika suatu kondisi tidak bisa terpenuhi (aliran alternatif).

Tabel 4.3 Use case New Project

Use case	New Project	
Aktor	*User*	
Tujuan	Menyiapkan *blank* project yang akan digunakan *user* sebagai proyek baru.	
Deskripsi	Jika user memilih menu New Project pada menu utama, maka *use case* New Project akan dijalankan. Use case ini kemudian membawa *user* ke tampilan utama (*main user interface)* merupakan tampilan awal dimana user akan memulai proyeknya.	
Pra-kondisi	Aktor memilih menu 'New Project' pada menu utama.	
Pos-kondisi	Menampilkan file-file yang diisi user pada *main user interface*.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari Sistem**
Aktor memilih menu 'New Project' pada menu utama.		Sistem akan membawa aktor menuju *main user interface* tempat dimana actor akan mengerjakan proyeknya.

Sumber : Analisis kebutuhan

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menyimpan proyek yang sedang dijalankan. Kebutuhan tersebut direpresentasikan oleh *use case* Save. Tabel 4.4 merupakan skenario *use case* Save.

Tabel 4.4 Use case Save Project

Use case	Save	
Aktor	*User*	
Tujuan	Menyimpan Proyek yang telah dikerjakan.	
Deskripsi	Jika user memilih menu Save pada menu utama, maka *use case* Save/Load akan dijalankan. Use case ini kemudian membawa *user* ke tampilan Save merupakan tampilan dimana user akan menyimpan proyeknya.	
Pra-kondisi	Ada project yang sedang dikerjakan	
Pos-kondisi	Menyimpan data sesuai tampilan akhir pada *main user interface*.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari Sistem**
Aktor memilih menu 'Save' pada menu utama.		Sistem akan membawa aktor menuju tampilan Save dimana aktor akan menyimpan proyeknya.
Aliran Alternatif 1 : Sudah ada proyek yang tersimpan dengan nama yang sama		

	Sistem akan memberitahu user untuk menyimpan dengan nama lain atau menumpuk file tersebut.

Sumber : Analisis kebutuhan

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk membuka proyek yang sebelumnya sudah tersimpan. Kebutuhan tersebut direpresentasikan oleh *use case* Load. Tabel 4.5 merupakan skenario *use case* Load.

Tabel 4.5 Use case Load Project

Use case	Load
Aktor	*User*
Tujuan	Membuka Proyek yang sebelumnya tersimpan.
Deskripsi	Jika user memilih menu Load Project pada menu utama, maka *use case* Load akan dijalankan. Use case ini kemudian membawa *user* ke tampilan Load.
Pra-kondisi	Aktor memilih menu 'Load' pada menu utama.
Pos-kondisi	Menampilkan data sesuai kondisi terakhir data tersebut tersimpan.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
Aktor memilih menu 'Load' pada menu utama.	Sistem akan membawa aktor menuju tampilan Load dimana aktor akan membuka proyeknya yang tersimpan.
Aliran Alternatif 1 : Jika ada proyek yang sedang dijalankan	
Aktor memilih menu menu 'Load'	Sistem akan memberitahu User untuk menyimpan file projectnya terlebih dahulu

Sumber : Analisis kebutuhan

Kebutuhan fungsional terakhir yang harus disediakan oleh sistem adalah kebutuhan untuk menggenerasikan input pada aplikasi menjadi sebuah file baru. Kebutuhan tersebut direpresentasikan oleh *use case* Export. Tabel 4.6 merupakan skenario *use case* Export.

Tabel 4.6 Use case Export Project

Use case	Export
Aktor	*User*
Tujuan	Sistem menggenerasikan file.
Deskripsi	Jika user memilih menu Generate pada *main user interface*, Use case ini kemudian akan menggenerasikan hasil output berupa file.java.
Pra-kondisi	File project sudah disimpan dan script belum di generate
Pos-kondisi	Sistem berhasil menggenerasi project menjadi script.java.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
Aktor memilih menu 'Generate' pada menu utama.	Sistem akan mengolah data input menjadi output berupa file.java.

Sumber : Analisis kebutuhan

4.2 Perancangan

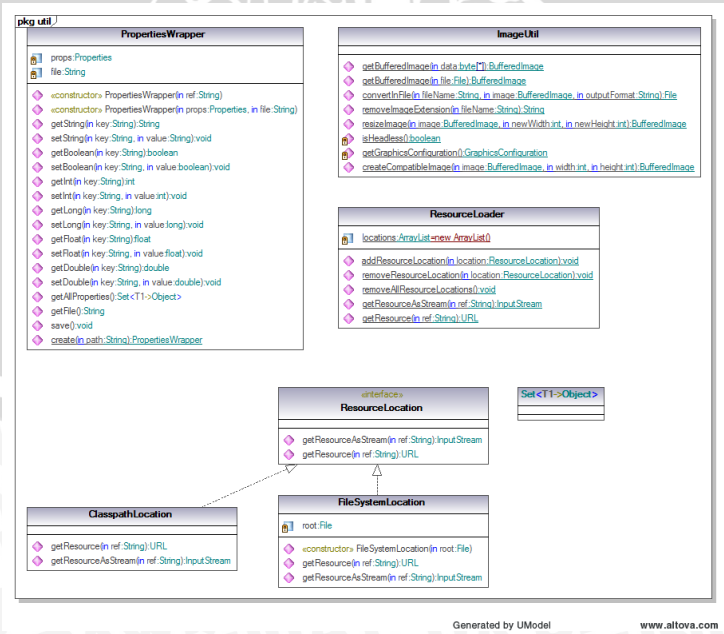
Perancangan perangkat lunak ditujukan untuk memberikan gambaran secara umum mengenai perangkat lunak yang akan dibuat. Hal ini berguna untuk menunjang pembuatan perangkat lunak sehingga kebutuhan perangkat lunak tersebut dapat terpenuhi.

Perancangan perangkat lunak pada skripsi ini menggunakan pendekatan desain berorientasi objek yang direpresentasikan dengan menggunakan UML (*Unified Modelling Language*). Pada proses desain dilakukan identifikasi terhadap klas-klas yang dibutuhkan yang dimodelkan dalam *class diagram* dan hubungan interaksi antar elemen (objek) yang telah diidentifikasi, dimodelkan dalam *sequence diagram.*

4.2.1 Diagram Klas (Class Diagram)

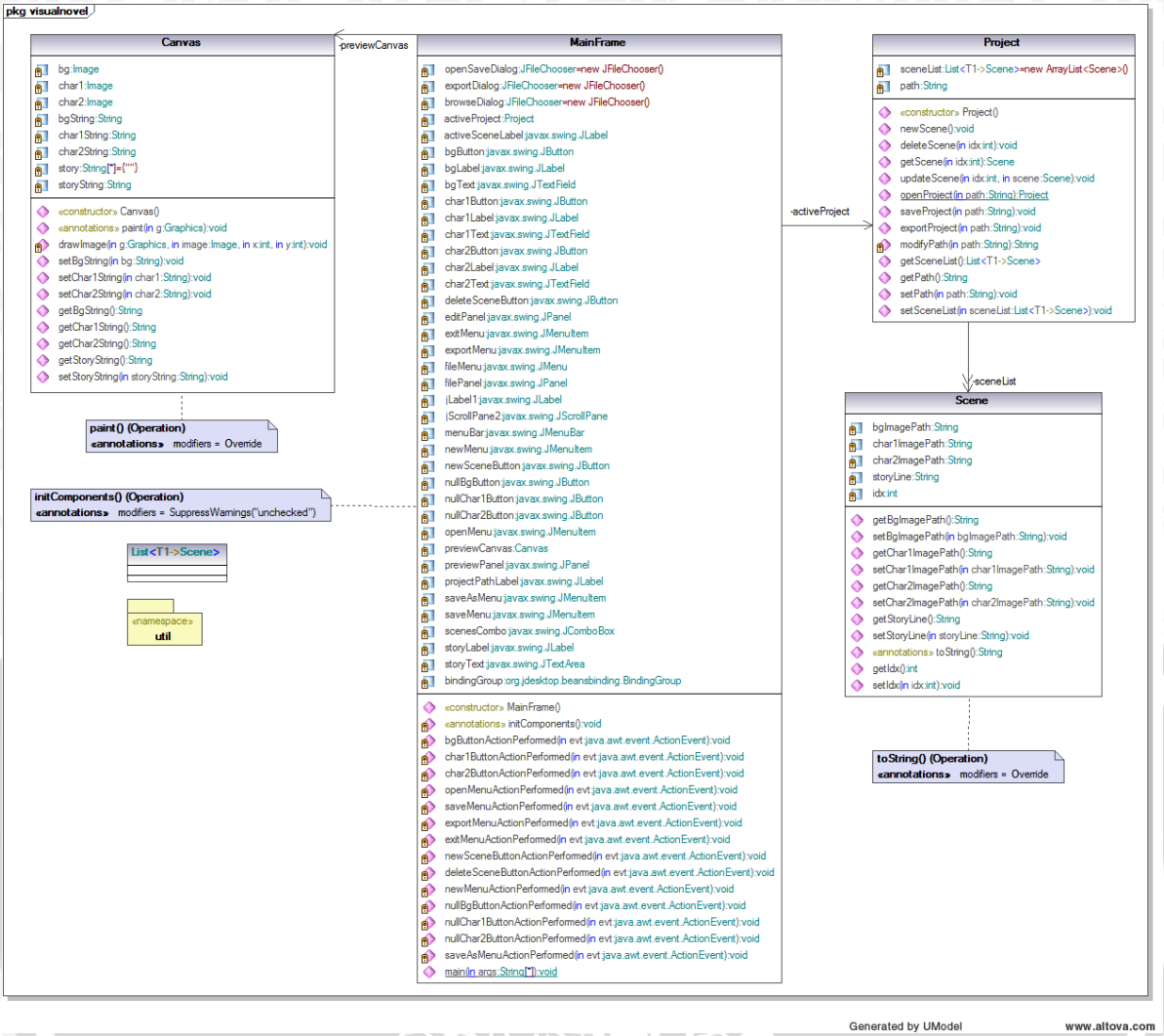
Diagram klas memberikan gambaran pemodelan elemen-elemen klas yang membentuk sebuah perangkat lunak mulai dari atribut-atribut serta method-methodnya dan hubungan dengan klas-klas lain dalam sebuah system. Pada system ini terdapat sejumlah klas yang saling membentuk relasi.

Atribut dan operasi yang dimiliki oleh masing-masing klas utama yang membangun sistem ini digambarkan dalam gambar 4.3, sedangkan gambar 4.2 merupakan klas utilities dimana klas tersebut adalah klas yang mendukung klas utama dalam pengembangan sistem.



Gambar 4.2 Diagram Klas Visual Novel Maker Utilities

Sumber : Perancangan



Gambar 4.3 Diagram Klas utama Visual Novel Maker

Sumber : Perancangan

Keseluruhan fungsi utama system seperti *new project, save project, open project,* dan *export project* terdapat dalam class *Project* yang terdapat dalam gambar 4.3. Klas yang berhubungan dengan klas *Project* antara lain *Scene*, *Mainframe*, dan *Canvas*.

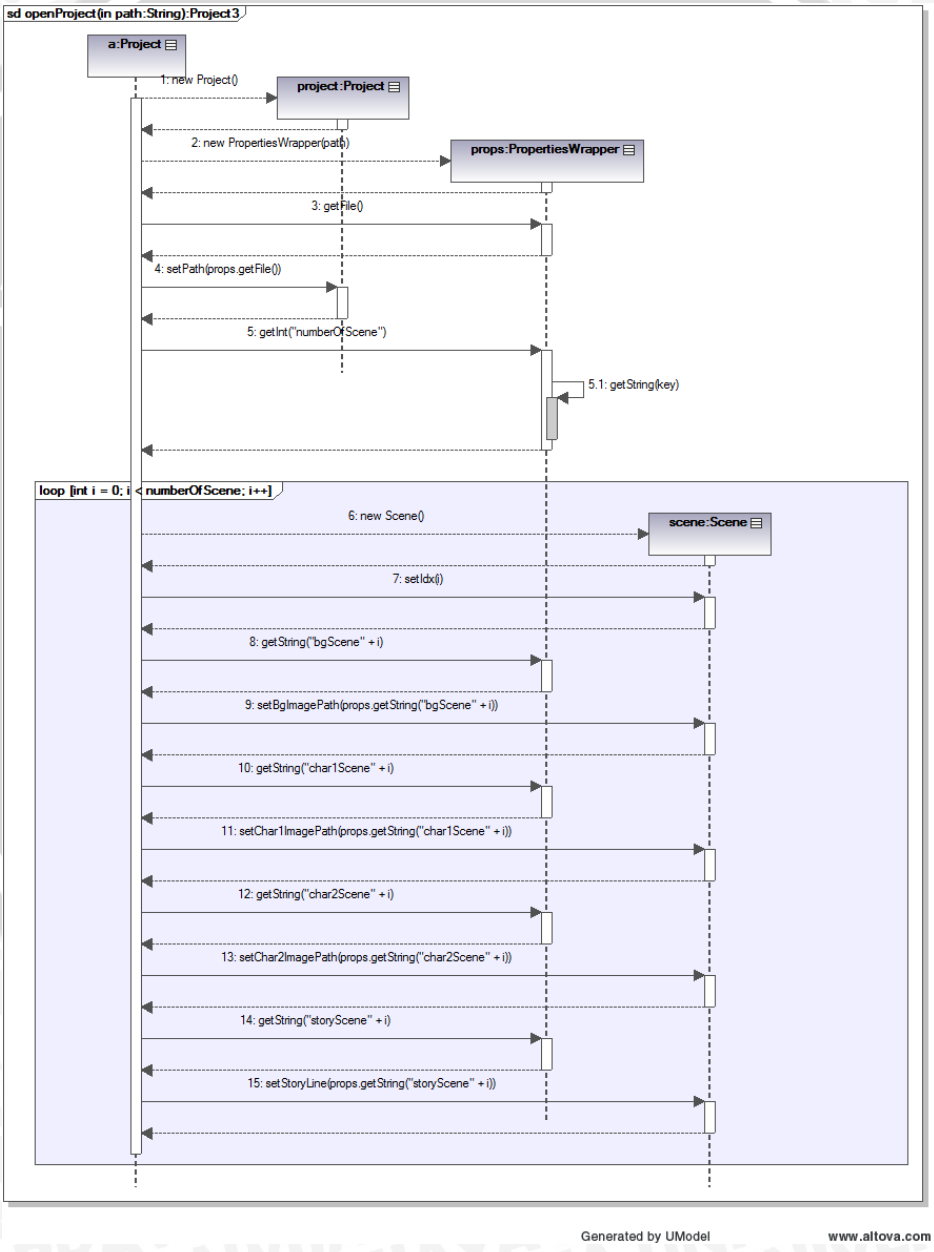
4.2.2 Diagram Sekuensial (*Sequence Diagram*)

Hubungan antar klas yang digambarkan pada diagram klas termasuk dalam pemodelan statis, tanpa disertai dengan gambaran aliran proses atau data antarklas yang satu dengan klas yang lain. *Sequence diagram* berguna untuk menampikan aliran jalannya proses yang ditunjukkan dengan interaksi antar objek atau klas, dan disusun berdasarkan urutan waktu. *Sequence diagram* dirancang dengan mengambil acuan pada

use case serta operasi – operasi dalam klas yang menjadi implementasi dari fungsionalitas yang digambarkan pada *use case* tersebut.

4.2.2.1 Diagram Sekuensial Untuk Fitur *New Project* dan *Open Project.*

Diagram sekuensial yang terdapat pada gambar 4.4 menggambarkan interaksi yang terjadi ketika user memilih untuk membuka proyek baru (*New Project*) dalam kasus ini *Open Project* juga termasuk dalam *diagram sequence* tersebut karena proses kerja kedua use case tersebut adalah hampir sama.

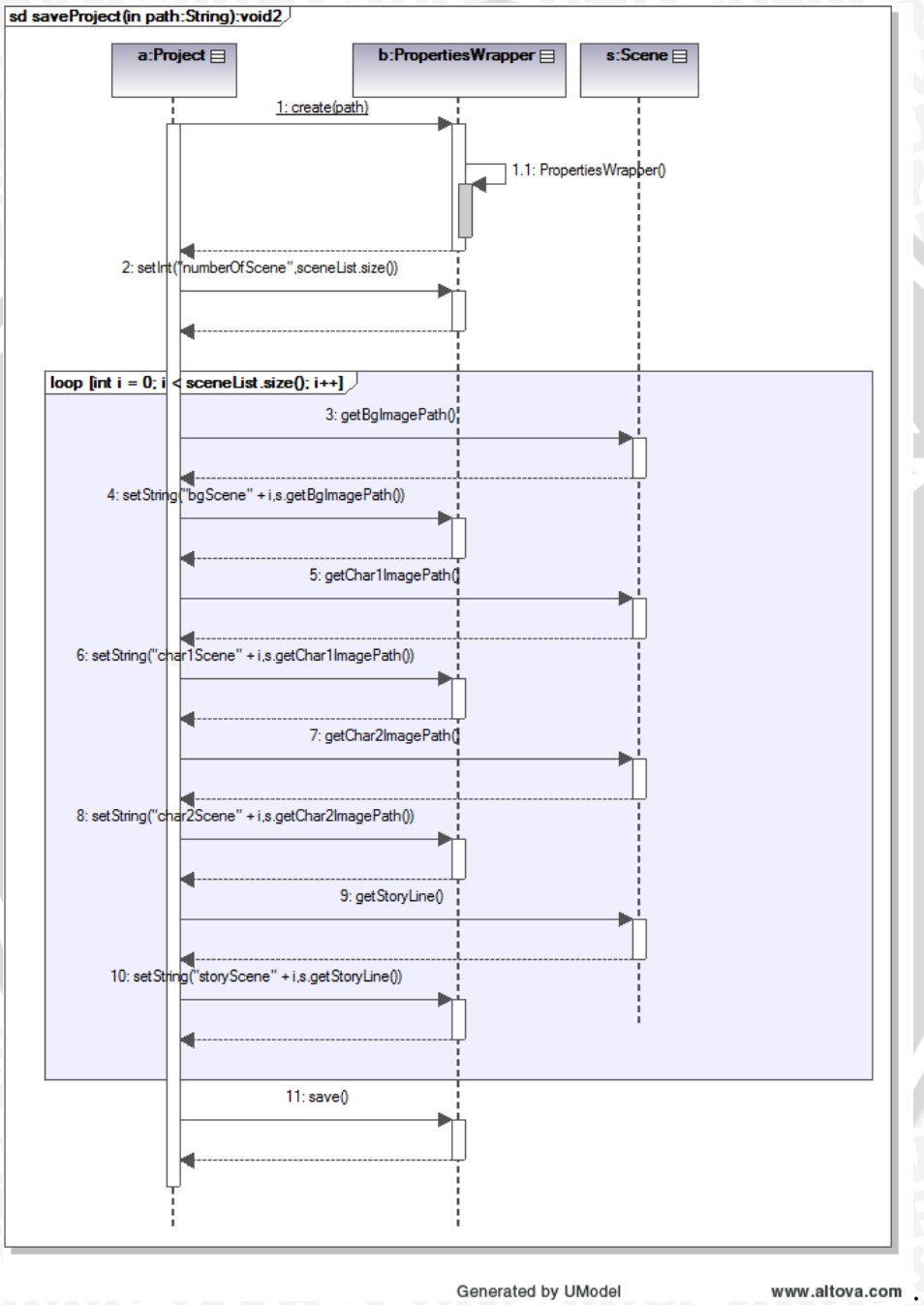


Gambar 4.4 Diagram sekuensial fitur New dan Open Project.

Sumber : Perancangan

4.2.2.2 Diagram Sekuensial Untuk Fitur *Save Project*.

Diagram sekuensial yang terdapat pada gambar 4.5 menggambarkan interaksi yang terjadi ketika user memilih untuk menyimpan file proyek yang sedang dikerjakan

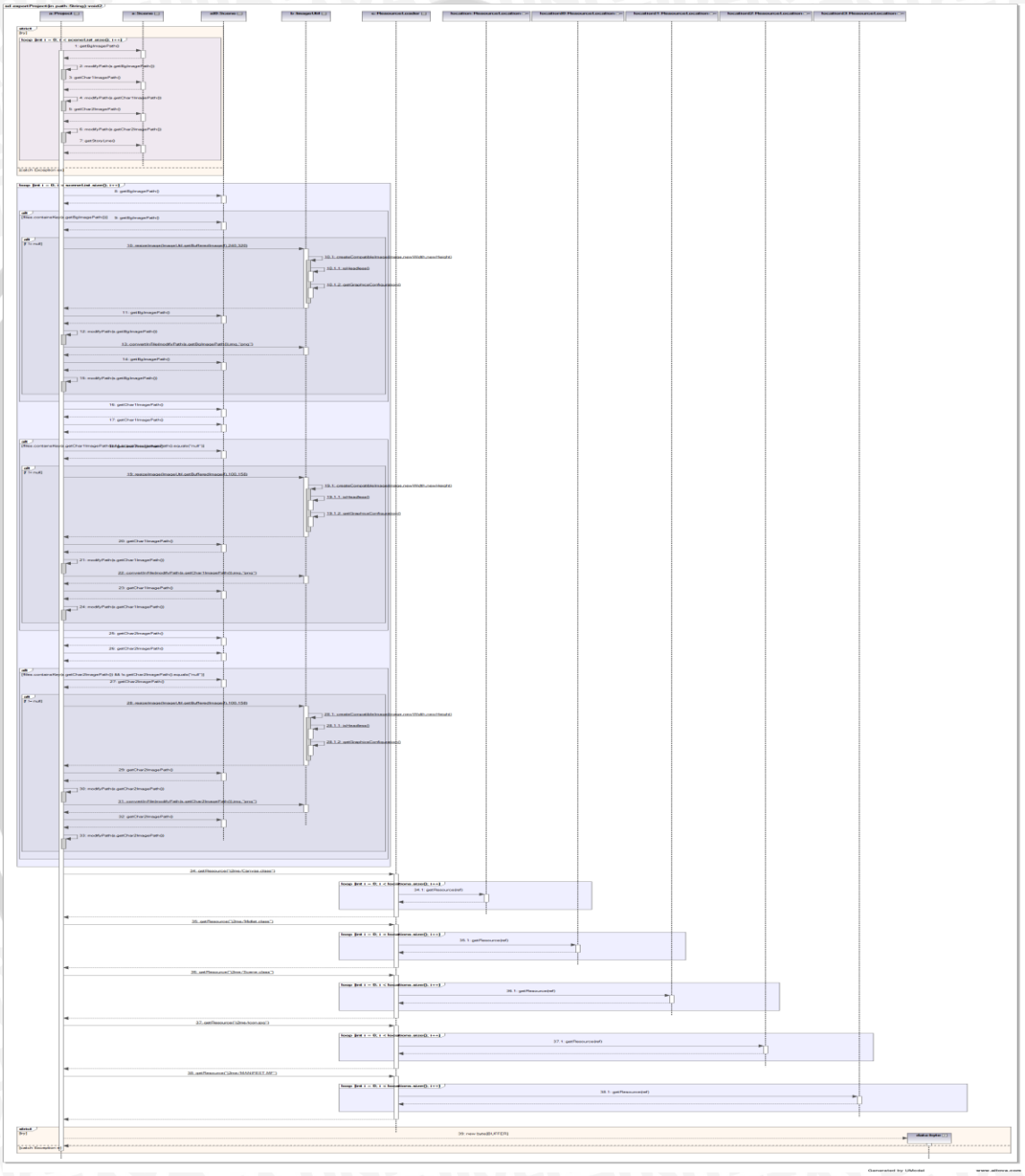


Gambar 4.5 Diagram Sekuensial Fitur Save Project

Sumber : Perancangan

4.2.2.3 Diagram Sekuensial Untuk Fitur *Export Project.*

Diagram sekuensial yang terdapat pada gambar 4.6 menggambarkan interaksi yang terjadi ketika user memilih untuk menggenerasikan file pada project menjadi file .jar .



Gambar 4.6. Diagram Sekuensial Fitur Export Project

Sumber : Perancangan

BAB V

IMPLEMENTASI

Bab ini membahas tentang implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak sebelumnya. Pembahasan terdiri dari penjelasan tentang spesifikasi lingkungan (spesifikasi perangkat keras dan perangkat lunak) di mana sistem diimplementasikan, implementasi tiap klas pada file program, implementasi algoritma dan implementasi antarmuka aplikasi.

5.1 Spesifikasi Sistem

Dari hasil analisis kebutuhan dan perancangan yang diuraikan pada Bab 4, dilakukan implementasi agar sistem ini bisa berfungsi sesuai dengan kebutuhan. Sistem diimplementasikan pada suatu lingkungan dengan spesifikasi *hardware* dan *software* seperti ditunjukkan pada Sub Bab 5.1.1 dan Sub Bab 5.1.2.

5.1.1 Spesifikasi Perangkat Keras (*Hardware*)

Spesifikasi perangkat keras dan perangkat lunak yang digunakan untuk implementasi aplikasi ini ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi perangkat keras untuk implementasi

Spesifikasi Perangkat Keras	
Prosesor	Intel(R) Core 2 Duo T9300
Memori (RAM)	3GB DDR3
Hardisk	Seagate 250 GB
Mother Board	Asus M50SV series (Notebook)
VGA Card	nVIDIA GeForce 9500M GS : 512MB
Mobile Phone	Sony Ericson W850i

Sumber : Implementasi

5.1.2 Spesifikasi Perangkat Lunak (*Software*)

Untuk membangun sistem ini dibutuhkan beberapa perangkat lunak yang dapat dilihat pada Tabel 5.2

Tabel 5.2 Spesifikasi perangkat lunak untuk implementasi

Spesifikasi Perangkat Lunak	
Sistem Operasi	Windows Vista Premium 32bit
Bahasa Pemrograman	Java2 ME, Java2SE
Tools Pemrograman	JDK 1.5.0_10 J2ME Wireless Tool Kit 2.5 Sony Ericsson SDK 2.5.0.4
IDE *(Integrated Development Environment)*	NetBeans 6.5 b + NetBeans Mobility Pack 6.5 b

Sumber : Implementasi

5.2 Implementasi Klas Pada File Program

Klas-klas yang telah didesain pada proses perancangan, masing-masing direalisasikan pada sebuah *file* program *.java. Pada Tabel 5.3 diberikan pasangan antara klas dengan *file* program yang digunakan untuk mengimplementasikannya.

Tabel 5.3 Implementasi Klas Pada File Program

No.	**Nama Klas**	**Nama *File* Program**
1	*Project*	*Project.java*
2	*Canvas*	*Canvas.java*
3	*MainFrame*	*MainFrame.java*
4	*Scene*	*Scene.java*
Utilities Class		
5	*Class Path Location*	*ClassPathLocation.java*
6	*File System Location*	*FileSystemLocation.java*
7	*Image Utilities*	*ImageUtil.java*
8	*Properties Wrapper*	*PropertiesWrapper.java*
9	*Resource Loader*	*ResourceLoader.java*
10	*Resource Location*	*ResourceLocation.java*

Sumber : Implementasi

5.3 Implementasi Algoritma

Pada sistem ini, terdapat beberapa proses utama (pokok). Diantara proses utama tersebut adalah proses pembuatan halaman proyek baru atau *new project*, proses penyimpanan dan pembukaan file *project*, dan logika sistem *Export* dimana *project* digenerasi menjadi output berupa file .jar yang dapat dijalankan pada *handphone*.

5.3.1 Implementasi Algoritma *New Project & Open Project*

Ketika user membuka program untuk pertama kalinya, maka yang dilakukan sistem pertama-kali adalah menyajikan tampilan interface utama yang kosong. tidak ada input yang diberikan kepada system untuk pertama kali, dengan logika demikian algoritma yang digunakan untuk membuka file project (*Open Project*) yang sebelumnya pernah disimpan adalah hampir sama dengan *new project*, yang membedakan adalah terdapat isi dari input yang diberikan kedalam sistem. Mekanisme tersebut diaplikasikan dalam kode pemrograman project.java yang ditunjukkan pada gambar 5.1.

```
Algoritma New Project

Fungsi untuk memulai project baru
BEGIN
1.      Mengisi value jumlah scene
2.      FOR(Counter < Jumlah Scene)
3.              Mengisikan path gambar background,karakter1,karakter2,dan text
                kedalam class scene.
4.      END FOR
END


Algoritma Open Project

Fungsi untuk membuka Project
BEGIN
1.      Membuka file text path.
2.      Mengisi value jumlah scene
3.      FOR(Counter<jumlah scene)
4.              Mengisikan path gambar background,karakter1,karakter2,dan text
                kedalam class scene.
5.      END FOR
END
```

Gambar 5.1 Algoritma *New Project* dan *Open Project*

Sumber : Implementasi

5.3.2 Implementasi Algoritma *Save Project*

Project yang sedang dikerjakan user dapat disimpan sewaktu-waktu, file project tersebut disimpan dalam bentuk file text. Mekanisme penyimpanan file tersebut diaplikasikan dalam kode program project.java seperti yang ditunjukkan pada gambar 5.2

```
Algoritma Save Project

Fungsi untuk menyimpan Project
BEGIN
1.      Menyimpan value jumlah scene kedalam variabel temporary
2.      FOR (Counter < jumlah scene)
3.              Menyimpan path gambar background, karakter 1, karakter 2, dan text kedalam variabel temporary.
4.      END FOR
5.      Simpan value kedalam file text
END
```

Gambar 5.2 Algoritma Save Project

Sumber : Implementasi

Berikut adalah untitled.novel, contoh file project dalam bentuk file text yang tersimpan :

```
#Properties

#Tue Jul 27 18:49:57 ICT 2010

bgScene0=C\:\\bg.jpg

char1Scene0=C\:\\character.png

char2Scene0=null

storyScene0=line1\nline2\nline3\nlin4\nline5\n

numberOfScene=1
```

Gambar 5.3 contoh isi data file project yang tersimpan

Sumber : Implementasi

5.3.3 Implementasi Algoritma *Export Project*

Implementasi algoritma *Export Project* yaitu dimana sistem akan mengubah file yang di-input menjadi output yang berupa file .jar. Implementasi algoritma tersebut terdapat dalam kode program project.java seperti dalam algoritma 5.4 berikut

Algoritma *Export Project*

Fungsi untuk meng-export Project
BEGIN
//generate script terlebih dahulu
1. Membuat temporary file (“vnscript”,”.script”)
2. **FOR** (Counter < jumlah scene)
3. Cetak “{“
4. Simpan setelah cetak “/” modified path dari background
5. Cetak “;”
6. Simpan setelah cetak “/” modified path dari char1
7. Cetak “;”
8. Simpan setelah cetak “/” modified path dari char2
9. Cetak “;”
10. Simpan isi dari storyline. Ganti semua isi dari storyline setelah (“\n”, “#”)
11. Cetak “;”
12. Cetak “}”
13 Cetak “\n”
14. **END FOR**
15. Ambil byte dari path
16. Simpan byte kedalam file text “vn.script”
17 Cetak vn.script.

// meresize gambar dan menaruhnya kedalam resource list
18. **FOR**(Counter<jumlah scene)
19. **IF** (background=null)
20. File path diambil = null
21. **END IF**
22. **ELSE** (Background !=null)
23. Gambar disimpan dalam memory buffer dan diresize menjadi 240x320
24. Gambar dimodifikasi menjadi file.png

```
25.                                     Ambil path gambar yang sudah dimodifikasi
26.                             END ELSE
27.                     IF (char1=null)
28.                             File path diambil = null
29.                     END IF
30.                             ELSE  (char1 !=null)
31                                      Gambar disimpan dalam memory buffer dan diresize menjadi 100x170
32                                      Gambar dimodifikasi menjadi file.png
33                                      Ambil path gambar yang sudah dimodifikasi
34                              END ELSE
35.                     IF (char2=null)
36.                             File path diambil = null
37.                     END IF
38.                             ELSE (char2 !=null)
39.                                     Gambar disimpan dalam memory buffer dan diresize menjadi 100x170
40.                                     Gambar dimodifikasi menjadi file.png
41.                                     Ambil path gambar yang sudah dimodifikasi
42.                             END ELSE
43.             END FOR
44.             Masukkan semua gambar yang dipakai kedalam resource folder
// Copy semua file kedalam 1 folder
// Copy dari /j2me dan dari files
// Compress file
45.             int buffer = 2048
46.             masukkan resource  "/j2me/canvas.class" file URI;
47.             masukkan resource  "/j2me/midlet.class" file URI;
48.             masukkan resource "/j2me/scene.class" file URI;
49.             masukkan resource "/j2me/icon.jpg" file URI;
50.             masukkan resource "/j2me/MANIFEST.MF" file URI;
51.             FOR ( i<ukuran file)
52.                             compress file menjadi file .jar
53.             END FOR.
END
```

Gambar 5.4 Algoritma *Export Project*

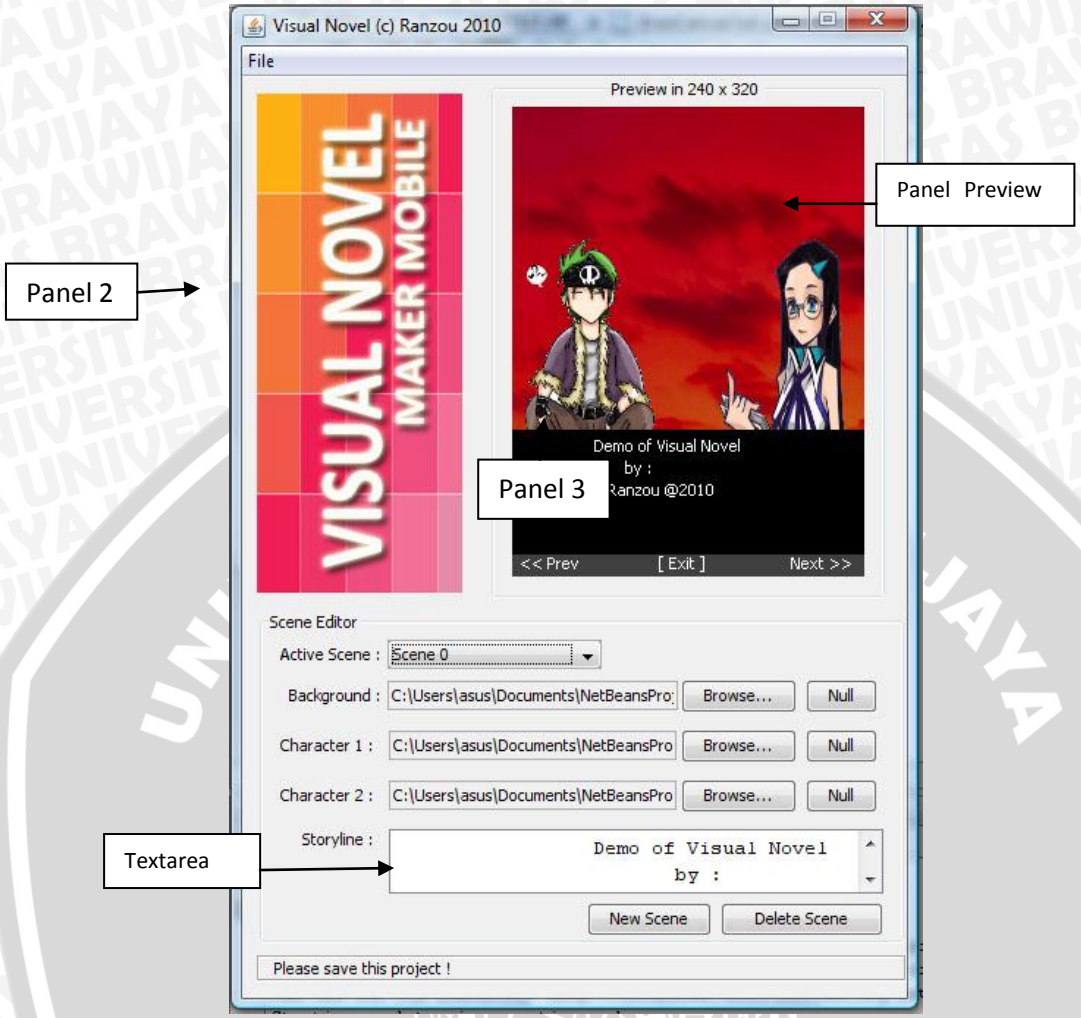
Sumber : Implementasi

5.4 Implementasi AntarmukaAplikasi

Pada aplikasi ini, antarmuka dibuat sesuai dengan kebutuhan fungsional yang harus disediakan oleh sistem. Tujuan utama pengembangan antarmuka adalah untuk memberikan kemudahan bagi pengguna (*user*) dalam menggunakan aplikasi yang telah dibangun.

5.4.1 Implementasi Antarmuka Aplikasi VNMaker.

Antarmuka ini akan ditampilkan begitu user membuka aplikasi. Aplikasi ini berupa *workplace*, tempat dimana user akan mengerjakan proyek pembuatan *visual novel* yang nanti akan di export menjadi sebuah file script J2ME yang di-*compress* menjadi file.jar sehingga dapat di jalankan pada *handphone*. Antarmuka ini dapat dilihat pada Gambar 5.5.



Gambar 5.5 Antarmuka program *VN maker*

Sumber : Implementasi

Gambar 5.5 merupakan gambar hasil implementasi antarmuka VNMaker. Penjelasan masing-masing bagian dari antarmuka diuraikan pada Tabel 5.4.

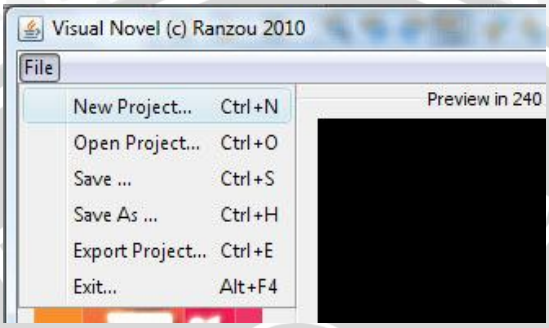
No	Label Panel/Tombol	Penjelasan
1	Panel Preview	Panel ini merupakan preview untuk menampilkan preview scene aktif
2	Textarea	Panel ini untuk mengisi storyline pada scene
3	File	Tombol ini digunakan untuk mengakses menu bar
4	Browse	Tombol ini digunakan untuk memilih file gambar
5	Null	Tombol ini digunakan untuk mengosongkan file gambar
6	New Scene	Tombol ini digunakan untuk menambah jumlah scene
7	Delete Scene	Tombol ini digunakan untuk menghapus scene

Tabel 5.4 Penjelasan panel dan tombol pada antarmuka VNMaker

Sumber : Implementasi

5.4.2 Implementasi Antarmuka Menu Bar

Pada aplikasi terdapat menu bar yang apabila menu file di klik maka akan keluar isi dari menu yaitu *new project, open project, save, save as.., export project* dan exit. Gambar 5.6 menunjukkan antarmuka file menu ketika di klik.



Gambar 5.6 Antarmuka menu bar

Sumber : Implementasi

Gambar 5.6 merupakan gambar hasil implementasi antarmuka *menu bar*. Penjelasan masing-masing bagian dari antarmuka diuraikan pada Tabel 5.5.

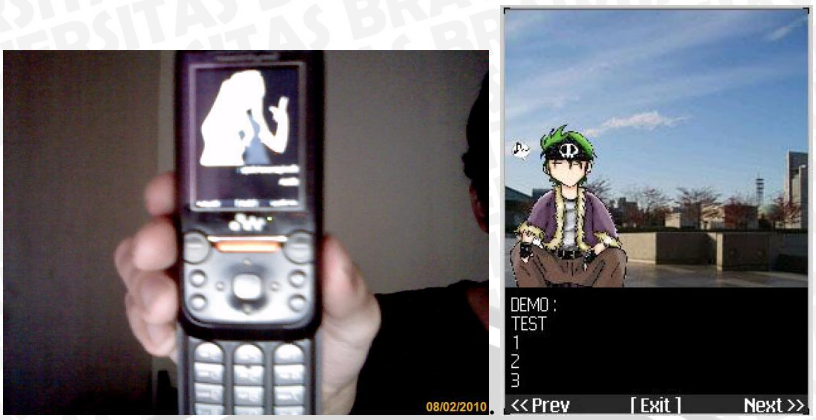
No	Label Panel/Tombol	Penjelasan
1	New Project	Tombol ini digunakan untuk memulai proyek baru
2	Open Project	Tombol ini digunakan untuk membuka proyek yang pernah disimpan
3	Save	Tombol ini digunakan untuk menyimpan project
4	Save as	Tombol ini digunakan untuk menyimpan project
5	Export Project	Tombol ini digunakan untuk Meng-*export* project
6	Exit	Tombol ini digunakan untuk menutup program

Tabel 5.5 Penjelasan panel dan tombol pada antarmuka *menú bar*

Sumber : Implementasi

5.4.4 Implementasi Antarmuka *Visual Novel* pada *Handphone*

Antarmuka ini adalah hasil project yang telah berhasil diexport sehingga dapat dijalankan pada *handphone*. Antarmuka program yang dijalankan pada *handphone* dapat dilihat pada gambar 5.7 berikut.



Gambar 5.7 Antarmuka VN pada *handphone & emulator*

Sumber : Implementasi.

Gambar 5.7 merupakan gambar hasil implementasi antarmuka *visual novel* yang telah di *compile* dan dijalankan pada *handphone.* Penjelasan masing-masing bagian dari antarmuka *visual novel* diuraikan pada Tabel 5.6.

No	Label Tombol	Penjelasan
1	Next	Tombol ini digunakan untuk melanjutkan ke scene berikut
2	Prev	Tombol ini digunakan untuk kembali ke scene sebelumnya
3	Exit	Tombol ini digunakan untuk mengakhiri aplikasi

Tabel 5.6 Penjelasan tombol-tombol pada antarmuka visual novel

Sumber : Implementasi

BAB VI

PENGUJIAN DAN ANALISIS

Bab ini membahas proses pengujian dan analisis terhadap sistem yang telah dibangun. Dalam proses pengujian digunakan tiga buah strategi pengujian, yaitu pengujian unit, pengujian integrasi dan pengujian validasi. Pada pengujian unit dan pengujian integrasi, akan digunakan teknik pengujian *white box (white box testing),* sedangkan pada pengujian validasi akan digunakan teknik pengujian *black box (black box testing).* Proses analisis dilakukan untuk mengetahui unjuk kerja dari sistem perangkat lunak yang sedang dibangun.

6.2 Pengujian Unit

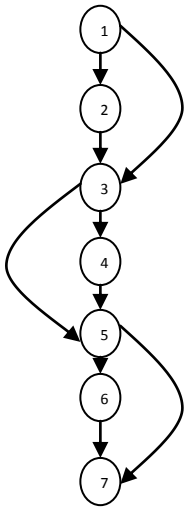
Dalam sistem yang dibangun dengan pemrograman berorientasi objek, pengujian unit diterapkan untuk suatu metode (operasi) dari suatu klas. Pada pengujian unit ini, digunakan teknik pengujian *white box (white box testing)* dengan teknik *basis path testing*. Pada teknik *basis path testing,* proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph*, menentukan jumlah kompleksitas siklomatis *(cyclomatic complexity),* menentukan sebuah basis set dari jalur independen dan memberikan kasus uji *(test case)* pada setiap basis set yang telah ditentukan.

6.1.1 Pengujian unit untuk operasi draw canvas

Operasi draw canvas merupakan implementasi dari algoritma kode pemrograman `canvas.java` dimana class tersebut merupakan bagian penting dari program sebagai class untuk menampilkan gambar pada scene preview. Dalam gambar 6.1 diperlihatkan proses pemodelan dalam *flow graph* pada operasi draw canvas test pada klas `Canvas`.

```
Algoritma Draw Canvas test

Fungsi untuk menampilkan preview scene
BEGIN
1.  IF (Background = “”)
2.          ambil path gambar background = “”                    } 1
3.  END IF
4.          ELSE (Background !=””)
5.                  Mengambil path gambar background             } 2
6.          END ELSE
7.  IF (Char1 = “”)
8.          ambil path gambar Char1 = “”                         } 3
9.  END IF
10.         ELSE (Char1 !=””)
11.                 Mengambil path gambar Char1                  } 4
12.         END ELSE
13.IF (Char2 = “”)
14.         ambil path gambar Char2 = “”                         } 5
15.END IF
16.         ELSE (Char2 !=””)
17.                 Mengambil path gambar Char2                  } 6
18.         END ELSE
19.         Ambil String pada setstory line
20.         Gambarkan kedalam canvas                             (7)
END
```



Gambar 6.1 Pemodelan Operasi Draw Canvas test ke dalam flow Graph

Sumber : Pengujian

6.1.1.1 Tujuan Pengujian

Pengujian terhadap operasi draw canvas test yang terdapat dalam klas Canvas, bertujuan untuk mengetahui unuk kerja dari operasi draw canvas secara independen dalam menjalankan fungsinya untuk menampilkan tampilan preview scene pada aplikasi visual novel maker. Pengujian dilakukan dalam beberapa uji kasus untuk mengetahui performa dari operasi `draw canvas` dalam menjalankan proses-proses dalam berbagai kemungkinan kondisi yang akan terjadi pada saat sistem menjalankan operasi tersebut.

6.1.1.2 Prosedur Pengujian

Dari hasil pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi `Draw Canvas Test` (Gambar 6.1), ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – N + 2

= 9 – 7 + 2

= 4

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 4 ditentukan satu basis set dari jalur independent yaitu:

Jalur 1 : 1, 2, 3, 4, 5, 6, 7

Jalur 2 : 1, 3, 4, 5, 6, 7

Jalur 3 : 1, 3, 5, 6, 7

Jalur 4 : 1, 3, 5, 7

Kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah :

1. Kasus uji jalur 1

   Menampilkan seluruh gambar dan tulisan pada canvas

   Hasil yang diharapkan : semua gambar tampil pada canvas baik background, character 1, character 2 dan text.

   Hasil pengujian : seluruh gambar tampil dengan posisi yang diinginkan.

2. Kasus uji jalur 2.

   Menampilkan gambar character 1 dan 2 tanpa menampilkan gambar background

   Hasil yang diharapkan : Gambar character 1 dan 2 dan text tampil tanpa adanya gambar background.

   Hasil pengujian : gambar character 1 dan 2 tampil dengan posisi yang diinginkan.

3. Kasus uji jalur 3

   Menampilkan gambar 2 tanpa menampilkan character 1 dan gambar background

   Hasil yang diharapkan : canvas hanya menampilkan gambar character 2 dan text tanpa adanya gambar character 1 serta background

   Hasil Pengujian : canvas menampilkan gambar character 2 dan text box.

4. Kasus uji jalur 4

   Menjalankan `Canvas` tanpa adanya gambar yang tampil.

   Hasil yang diharapkan : sistem hanya menampilkan textbox.

   Hasil pengujian : Sistem menampilkan canvas kosong yang hanya berisi textbox.

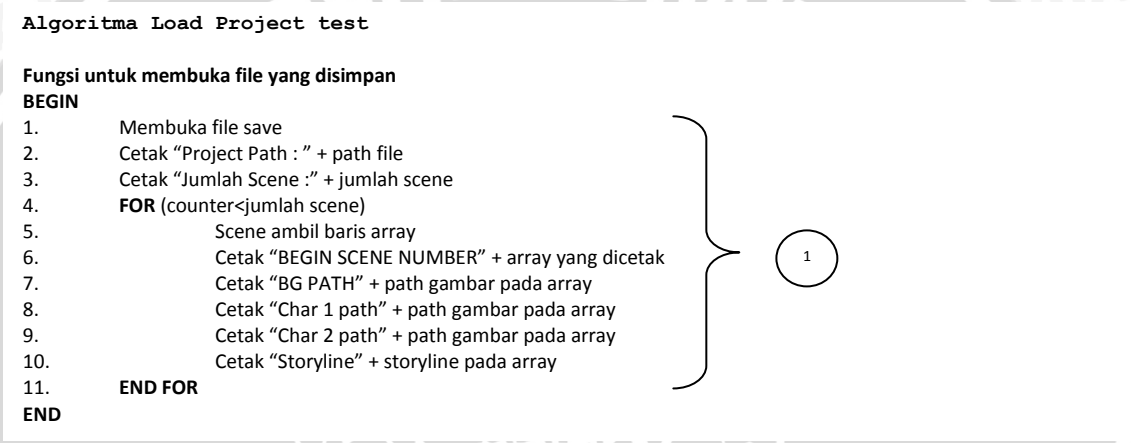
6.1.1.3 Analisis Pengujian

Pengujian unit untuk operasi `draw canvas test` dalam klas `Canvas` memberikan hasil pengujian sesuai dengan hasil yang diharapkan berdasarkan kasus uji yang terdapat dalam pemodelan *flow graph* yang disertakan dalam gambar 6.1. Hasil

pengujian operasi draw canvas test dalam klas `Canvas` ini membuktikan bahwa sistem dapat menampilkan preview sesuai dengan data scene yang diterima.

6.1.2 Pengujian unit untuk operasi Load Project

Operasi Load Project merupakan implementasi dari algoritma kode pemrograman `project.java` dimana class tersebut merupakan bagian penting dari program sebagai class untuk membuka file project yang sebelumnya disimpan. Dalam gambar 6.2 diperlihatkan proses pemodelan dalam *flow graph* pada operasi load project test pada klas `project`.



Gambar 6.2 Pemodelan Operasi Load Project Test ke dalam flow Graph

Sumber : Pengujian

6.1.2.1 Tujuan Pengujian

Pengujian terhadap operasi *load project test* yang terdapat dalam klas Project, bertujuan untuk mengetahui unuk kerja dari operasi Load Project secara independen dalam menjalankan fungsinya untuk membaca format file project yang sebelumnya disimpan, pada aplikasi visual novel maker. Pengujian dilakukan dalam beberapa uji kasus untuk mengetahui performa dari operasi `load project` dalam menjalankan proses-proses dalam berbagai kemungkinan kondisi yang akan terjadi pada saat sistem menjalankan operasi tersebut.

6.1.2.2 Prosedur Pengujian

Dari hasil pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi `Load Project test` (Gambar 6.2), ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G)

merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2

= 0 – 1 + 2

= 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 1 ditentukan satu basis set dari jalur independent yaitu:

Jalur 1 : 1

Kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah :

a. Kasus uji jalur 1

   Proses pembacaan format file project yang disimpan berlangsung.

   Hasil yang diharapkan : system merespon dan dapat membaca isi dari file format project yang tersimpan.

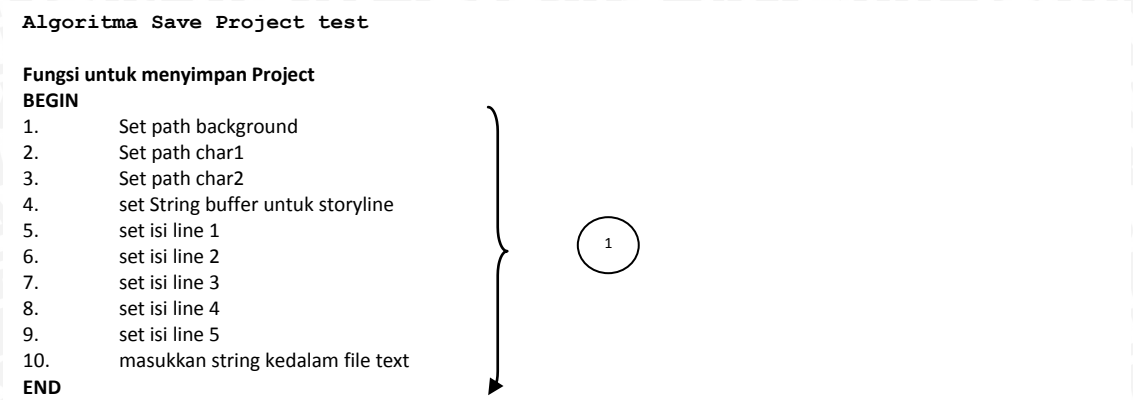
   Hasil pengujian : data berhasil dibaca dan isi ditampilkan.

6.1.2.4 Analisis Pengujian

Pengujian unit untuk operasi `load project test` dalam klas `Project` memberikan hasil pengujian sesuai dengan hasil yang diharapkan berdasarkan kasus uji yang terdapat dalam pemodelan *flow graph* yang disertakan dalam gambar 6.2. Hasil pengujian operasi load project test dalam klas `Project` ini membuktikan bahwa sistem dapat membaca dan membuka file format project yang disimpan sesuai data yang diterima.

6.1.3 Pengujian unit untuk operasi Save Project

Operasi Save Project merupakan implementasi dari algoritma kode pemrograman `project.java` dimana class tersebut merupakan bagian penting dari program sebagai class untuk menyimpan file project yang sedang aktif. Dalam gambar 6.3 diperlihatkan proses pemodelan dalam *flow graph* pada operasi load project test pada klas `project`.



Gambar 6.3 Pemodelan Operasi Save Project Test ke dalam flow Graph

Sumber : Pengujian

6.1.3.1 Tujuan Pengujian

Pengujian terhadap operasi *Save project test* yang terdapat dalam klas Project, bertujuan untuk mengetahui unuk kerja dari operasi Save Project secara independen dalam menjalankan fungsinya untuk menyimpan file project yang sedang aktif, pada aplikasi visual novel maker. Pengujian dilakukan dalam beberapa uji kasus untuk mengetahui performa dari operasi `save project` dalam menjalankan proses-proses dalam berbagai kemungkinan kondisi yang akan terjadi pada saat sistem menjalankan operasi tersebut.

6.1.3.2 Prosedur Pengujian

Dari hasil pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi `Save Project` (Gambar 6.3), ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2
= 0 – 1 + 2
= 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 1 ditentukan satu basis set dari jalur independent yaitu:

Jalur 1 : 1

Kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah :

b. Kasus uji jalur 1

Proses pengambilan path gambar dan storyline project yang akan disimpan kedalam file text.

Hasil yang diharapkan : system merespon dan dapat mengambil path dari input yang diberikan dan disimpan kedalam file berformat text.

Hasil pengujian : data input yang diberikan berhasil dibaca dan berhasil disimpan.

6.1.3.4 Analisis Pengujian

Pengujian unit untuk operasi `save project test` dalam klas `Project` memberikan hasil pengujian sesuai dengan hasil yang diharapkan berdasarkan kasus uji yang terdapat dalam pemodelan *flow graph* yang disertakan dalam gambar 6.3. Hasil pengujian operasi load project test dalam klas `Project` ini membuktikan bahwa sistem dapat membaca input dan menyimpannya kedalam file format text yang akan dijadikan file format project yang akan disimpan.

6.2 Pengujian Integrasi

Pengujian integrasi diterapkan pada proses yang mengintegrasikan fungsionalitas dari beberapa klas untuk melakukan sebuah operasi tertentu. Pada pengujian integrasi yang dijadikan sebagai obyek uji adalah klas-klas yang menggabungkan kinerja dari klas-klas yang lain. Klas-klas yang mengintegrasikan beberapa klas yang lain tersebut akan berperan juga sebagai *test driver* yang mengendalikan proses pengujian sehingga bisa memperlihatkan status valid atau tidaknya hasil integrasi. Dalam melakukan pengujian integrasi terhadap sistem perangkat lunak ini digunakan strategi *bottom-up*, dimana klas-klas yang diintegrasikan masing-masing diuji terlebih dahulu dalam pengujian unit dan kemudian bergerak menuju ke pengujian klas-klas kontrol yang mengintegrasikannya.

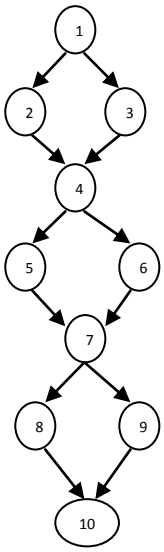
6.2.1 Pengujian Integrasi untuk operasi *Export Project*

Operasi `Export Project` merupakan implementasi dari algoritma project yang digunakan untuk melakukan pengkompressasian project yang telah dikerjakan, meliputi resources yang digunakan kedalam 1 file *compressed*. Operasi ini terdapat dalam klas Project. Dalam gambar 6.4 diperlihatkan proses pemodelan dalam *flow graph* pada operasi `Export Project` pada klas Project.

Algoritma *Export Project*

Fungsi untuk meng-export Project
BEGIN
//generate script terlebih dahulu
1. Membuat temporary file (“vnscript”,”.script”)
2. **FOR** (Counter < jumlah scene)
3. Cetak “{“
4. Simpan setelah cetak “/” modified path dari background
5. Cetak “;”
6. Simpan setelah cetak “/” modified path dari char1
7. Cetak “;”
8. Simpan setelah cetak “/” modified path dari char2
9. Cetak “;”
10. Simpan isi dari storyline. Ganti semua isi dari storyline setelah (“\n”, “#”)
11. Cetak “;”
12. Cetak “}”
13 Cetak “\n”
14. **END FOR**
15. Ambil byte dari path
16. Simpan byte kedalam file text “vn.script”
17 Cetak vn.script.

// meresize gambar dan menaruhnya kedalam resource list
18. **FOR**(Counter<jumlah scene)
19. Cek Background
20. **IF** (background=null)
21. File path diambil = null
22. **END IF**
23. **ELSE** (Background !=null)
24. Gambar disimpan dalam memory buffer dan diresize menjadi 240x320
25. Gambar dimodifikasi menjadi file.png
26. Ambil path gambar yang sudah dimodifikasi
27. **END ELSE**
28. Cek Char1
29. **IF** (char1=null)
30. File path diambil = null
31. **END IF**
32. **ELSE** (char1 !=null)
33. Gambar disimpan dalam memory buffer dan diresize menjadi 100x170
34. Gambar dimodifikasi menjadi file.png
35. Ambil path gambar yang sudah dimodifikasi
36. **END ELSE**
37 Cek Char2
38. **IF** (char2=null)
39. File path diambil = null
40. **END IF**
41. **ELSE** (char2 !=null)
42. Gambar disimpan dalam memory buffer dan diresize menjadi 100x170
43. Gambar dimodifikasi menjadi file.png
44. Ambil path gambar yang sudah dimodifikasi
45. **END ELSE**
46. **END FOR**
47. Masukkan semua gambar yang dipakai kedalam resource folder
// Copy semua file kedalam 1 folder
// Copy dari /j2me dan dari files
// Compress file
48. **int** buffer = 2048
49. masukkan resource “/j2me/canvas.class” file URI;
50. masukkan resource “/j2me/midlet.class” file URI;
51. masukkan resource “/j2me/scene.class” file URI;
52. masukkan resource “/j2me/icon.jpg” file URI;
53. masukkan resource “/j2me/MANIFEST.MF” file URI;
54. **FOR** (i<ukuran file)
55. compress file menjadi file .jar
53. **END FOR**.
END



Gambar 6.4 Pemodelan Operasi export Project ke dalam flow Graph

Sumber : Pengujian

6.2.2.1 Tujuan Pengujian

Pengujian operasi `Export Project` yang terdapat dalam klas Project bertujuan untuk mengetahui unjuk kerja operasi `Export Project` serta interaksi antar klas yang terdapat di dalamnya. Pengujian dilakukan dalam beberapa uji kasus untuk mengetahui performa dari operasi `Export Project` dalam menjalankan proses-proses dalam berbagai kemungkinan kondisi yang akan terjadi pada saat sistem menjalankan operasi tersebut. Kelancaran interaksi data di antara klas-klas yang terhubung dalam berbagai kondisi juga menjadi objek dalam pengujian ini.

6.2.2.2 Prosedur Pengujian

Dari hasil pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi `Export Project` (Gambar 6.4), ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – N + 2
= 12 – 10 + 2
= 4

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 4 ditentukan satu basis set dari jalur independent yaitu:

Jalur 1 : 1-2-4-5-7-8-10
Jalur 2 : 1-3-4-5-7-8-10
Jalur 3 : 1-3-4-6-7-8-10
Jalur 4 : 1-3-4-6-7-9-10

Kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah :

1. Kasus uji jalur 1

   Menyimpan semua *resource* gambar yang digunakan mulai dari background, char 1 dan char 2 serta script dan file j2me kemudian semua file di compress menjadi 1 file.jar.

   Hasil yang diharapkan : semua gambar berhasil di resize dan di modifikasi.

Hasil pengujian : seluruh gambar berhasil di resize dan di modifikasi kemudian dikompress menjadi 1 file.jar.

2. Kasus uji jalur 2.

   Hanya mengambil gambar character 1 dan 2 , script dan class j2me kemudian seluruh file dikumpulkan kedalam 1 file yang kemudian di compress menjadi 1 file

   Hasil yang diharapkan : Gambar character 1 dan 2 besera script ada dalam file

   Hasil pengujian : gambar character 1 dan 2 yang telah dimodifikasi beserta script terdapat didalam file.

3. Kasus uji jalur 3

   Memasukkan gambar char2 , script dan class j2me tanpa character 1 dan background

   Hasil yang diharapkan : file berhasil di compress dengan char2, script terdapat didalamnya

   Hasil Pengujian : gambar char2, beserta script terdapat didalamnya

4. Kasus uji jalur 4

   Hanya file script dan class j2me yang di kompress.

   Hasil yang diharapkan : tidak adanya resource gambar pada file.jar.

   Hasil pengujian : hanya terdapat file script dan class j2me dalam file.jar.

6.2.2.3 Analisis Pengujian

Pengujian integrasi untuk operasi `export project` dalam klas `Project` memberikan hasil pengujian yang sesuai dengan hasil yang diharapkan untuk masing-masing jalur uji sesuai dengan pemodelan *flow graph* yang disertakan dalam gambar 6.4. Hal ini membuktikan interaksi antar klas yang terjadi dalam operasi `export project` dapat berjalan dengan baik dan sistem dapat mengkompresikan file-file sesuai dengan daftar kebutuhan sistem.

6.2 Pengujian Validasi

6.3.1 Tujuan Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. Item-item yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Dalam melakukan pengujian validasi digunakan metode

pengujian *Black Box,* karena tidak memerlukan untuk berkonsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan. Obyek dari pengujian validasi adalah item-item kebutuhan yang telah dirumuskan pada daftar kebutuhan pada Tabel 4.2. Nama kasus uji dan obyek pengujian ditunjukkan pada Sub Bab 6.3.1 dan hasil pengujiannya akan ditunjukkan pada Sub Bab 6.3.2.

6.3.2 Kasus Uji Validasi

Untuk mengetahui kesesuaian antara kebutuhan dengan kinerja sistem, pada setiap kebutuhan *(requirement)* dilakukan proses pengujian dengan masing-masing kasus uji.

a. Kasus Uji Pemilihan New Profile

Nama Kasus Uji	:	Kasus Uji *New Project*
Objek Uji	:	Kebutuhan Fungsional (F01)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk membuat Project kosong.
Prosedur Uji	:	1. Menjalankan aplikasi untuk pertama kalinya 2. Memilih pilihan *new project*.
Hasil yang diharapkan	:	Sistem akan membawa user ke tampilan awal interface yang kosong, tidak ada input sama sekali.

b. Kasus Uji Pemilihan New Profile

Nama Kasus Uji	:	Kasus Uji *Save Project*
Objek Uji	:	Kebutuhan Fungsional (F02)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk menyimpan Project yang ada kedalam file.
Prosedur Uji	:	1. Mengisi interface aplikasi project dengan beberapa input. 2. Memilih pilihan *save project*.
Hasil yang diharapkan	:	Sistem akan menyimpan kondisi project yang sedang berlangsung kedalam sebuah file.

c. Kasus Uji Pemilihan Open Project

Nama Kasus Uji	:	Kasus Uji *Open Project*
Objek Uji	:	Kebutuhan Fungsional (F02)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk membuka file Project yang sebelumnya pernah disimpan.
Prosedur Uji	:	1. Menjalankan aplikasi untuk pertama kalinya 2. Memilih pilihan *Open project*.
Hasil yang diharapkan	:	Sistem akan menbuka file dan menampilkan interface project dalam kondisi sama seperti kondisi dimana sebelumnya project pernah tersimpan.

d. Kasus Uji Pemilihan Export Project

Nama Kasus Uji	:	Kasus Uji *Export Project*
Objek Uji	:	Kebutuhan Fungsional (F03)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk membuka file Project yang sebelumnya pernah disimpan.
Prosedur Uji	:	1. kondisi project telah siap untuk di-export 2. Memilih pilihan *Export project*.
Hasil yang diharapkan	:	Sistem akan mengkompresi file script, dan gambar menjadi file.jar yang dapat dijalankan pada handphone.

6.3.3 Hasil Pengujian Validasi

Dari kasus uji yang telah dilaksanakan sesuai dengan prosedur pengujian pada sub pokok bahasan 6.3.1, didapatkan hasil seperti ditunjukkan pada Tabel 6.1.

No	Kasus Uji	Hasil yang didapatkan	Status
1	Kasus Uji *New Project*	Sistem membawa *user* ke project kosong.	Valid
2	Kasus Uji *Save Project*	Sistem menyimpan Project yang telah dibuat.	Valid
3	Kasus Uji *Open Project*	Sistem membaca data yang telah disimpan dan menampilkannya sebagai Project yang sebelumnya telah disimpan	Valid
4	Kasus Uji *Export Project*	Sistem meng-*export* Project menjadi file .jar yang dapat dijalankan pada handphone.	Valid

Tabel 6.1 Hasil pengujian validasi

6.3.4 Analisis Pengujian Validasi

Data hasil pengujian yang terdapat dalam Tabel 6.1 merupakan data yang didapat dengan membandingkan performa sistem dengan daftar kebutuhan sistem.

Kasus uji 1 merupakan kasus uji yang diberikan untuk menguji kebutuhan fungsional *new project* yang memberikan akses bagi *user* untuk membuka sebuah project baru. Pengujian validasi berjalan dengan lancar, dan sistem memberikan hasil pengujian yang valid sesuai hasil yang diharapkan.

Kasus uji 2 merupakan kasus uji yang diberikan untuk menguji kebutuhan fungsional *save project* yang memberikan akses bagi user untuk menyimpan project yang sedang berjalan. Pengujian validasi berjalan dengan lancar, dan sistem memberikan hasil pengujian yang valid sesuai hasil yang diharapkan.

Kasus uji 3 merupakan kasus uji yang diberikan untuk menguji kebutuhan fungsional *open project* yang memberikan akses bagi *user* untuk membuka file project yang sebelumnya pernah tersimpan dan system menampilkan project sesuai dengan kondisi yang sebelumnya pernah tersimpan. Pengujian validasi berjalan dengan lancar, dan sistem memberikan hasil pengujian yang valid sesuai hasil yang diharapkan.

Kasus uji 4 merupakan kasus uji yang diberikan untuk menguji kebutuhan fungsional *export project* yang memberikan akses bagi *user* untuk meng-*export project* yang dikerjakan menjadi file.jar yang diharapkan dapat dijalankan pada *handphone*. Pengujian validasi berjalan dengan lancar, dan sistem memberikan hasil pengujian yang valid sesuai hasil yang diharapkan.

Hasil pengujian kebutuhan fungsional sistem membuktikan bahwa sistem telah dapat memenuhi kebutuhan fungsional sistem dengan baik.

6.4 Pengujian Pada Notebook/Desktop PC

6.4.1 Pengujian Aplikasi *Visual Novel Maker*

Nama Kasus Uji	:	Kasus Uji Aplikasi
Objek Uji	:	Kebutuhan Non Fungsional (N01)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi *Visual Novel Maker* dapat direalisasikan pada perangkat *Desktop PC* sebagaimana dalam kebutuhan non fungsional
Spesifikasi Hardware	:	• Asus M50SV
	:	• OS : Windows Vista Premium 32bit • IDE : NetBeans 6.5b
Prosedur Uji	:	1. Melakukan proses run program pada netbeans**.**
	:	2. Menjalankan aplikasi
Hasil yang diharapkan	:	Aplikasi dapat dapat direalisasikan pada perangkat *Desktop PC/Notebook*
Hasil Pengujian	:	Aplikasi dapat direalisasikan pada perangkat *Desktop PC / Notebook*

6.5 Pengujian Pada *Handphone*

6.5.1 Pengujian Aplikasi *Visual Novel*

Nama Kasus Uji	:	Kasus Uji Aplikasi
Objek Uji	:	Kebutuhan Non Fungsional (N02)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi *Visual Novel* yang merupakan output dari *visual novel maker* dapat direalisasikan pada perangkat *Handphone* sebagaimana dalam kebutuhan non fungsional.
Spesifikasi Hardware	:	• Sony Ericsson w850i
	:	• Device Profile Configuration : MIDP 2.0 • Device Configuration : CLDC 1.1

Prosedur Uji	:	1. Melakukan proses instalasi aplikasi.
	:	2. Menjalankan aplikasi
Hasil yang diharapkan	:	Aplikasi dapat dapat direalisasikan pada perangkat *Handphone.*
Hasil Pengujian	:	Aplikasi dapat direalisasikan pada perangkat *Handphone.*

BAB VII

PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis yang dilakukan terhadap kinerja sistem, diambil kesimpulan sebagai berikut :

1. Aplikasi *visual novel* untuk telepon genggam yang digunakan sebagai basis dari aplikasi *visual novel* yang dihasilkan *visual novel* maker berhasil dirancang sesuai dengan analisis kebutuhan.
2. Aplikasi *visual novel maker* yang dapat menggenerate gambar dan script yang diinput, mengolah dan menjadikannya sebuah aplikasi *visual novel* dalam bentuk file.jar berhasil dirancang sesuai analisa kebutuhan.
3. Analisis kebutuhan dari sistem yang dirancang menghasilkan 4 kebutuhan fungsional, yaitu *new project*, *open project*, *save project*, dan *export project.*
4. Aplikasi *Visual Novel Maker* dapat meresize ukuran file gambar yang akan digunakan sebagai resource pada *visual novel*.
5. *Visual novel* dalam bentuk File.jar dapat dijalankan pada perangkat *handphone* dengan resolusi 240x320 pixel.
6. Sistem telah memiliki kemampuan untuk memenuhi semua kebutuhan fungsional sistem tanpa kesalahan

7.2 Saran

Mengacu dari kelemahan yang terdapat pada aplikasi *visual novel maker* dan *visual novel* yang telah dikembangkan dalam skripsi ini, beberapa saran yang bisa diambil diantaranya:

1. Aplikasi *visual novel maker* dapat dikembangkan lagi dengan menambahkan fitur-fitur seperti *audio*, *Multi optional route*, *save-state* sehingga *user* dapat mengembangkan *visual novel* yang lebih interaktif.
2. Variasi resolusi layar handphone yang berbeda-beda seharusnya bisa membuat *visual novel maker* ini menghasilkan aplikasi yang lebih *flexible* sehingga tidak terpaku pada *handphone* dengan resolusi 240x320 saja.

3. Seiring dengan perkembangan teknologi telepon genggam, ada beberapa faktor dalam permainan yang dapat dikembangkan, seperti kualitas grafis, audio, serta efek-efek khusus yang dapat mempercantik tampilan game, yang tidak dapat diterapkan saat ini karena keterbatasan kemampuan perangkat keras yang dimiliki telepon genggam.

DAFTAR PUSTAKA

[DAV-05] Davison, Andrew, 2005, "*Killer Game Programming in Java*", O'Reilly.

[ADI-03] Aditya Hartanto, Antonius. 2003. "*Java 2 Micro Edition Mobile Interface Device Programming*". Elex Media Komputindo.

[HJS-08] Harbour. Jonathan S. 2008. "*Beginning Java™ Game Programming 2nd edition*" Thomson Course Technology PTR.

[MBG-07] Mclaughlin, Brett D., Pollice , Gary, West, David, 2007, "*Head First Object-Oriented Analysis and Design*". O'Reilly Media, Inc.

[HZA-06] Hong Zhang, Y. Daniel Liang, 2006, "*Computer Graphics Using Java 2D and 3D*", Prentice Hall.

[SHA-06] Shalahuddin M. , A.S. Rossa, 2006, "*Pemrograman J2ME : Belajar Cepat Pemrograman Perangkat Telekomunikasi Mobile* ", Informatika Bandung, Bandung.

[NUG-05] Nugroho, Adi, 2005, "*Analisis dan Perancangan Sistem Informasi dengan Metodologi Berorientasi Objek*", Informatika, Bandung.

[FSO-05] Flyint, John, P. Salem, Omar. 2005. "*Software Engineering for Game Development*". Thompson Course Technology.

[ARA-03] Adams, Ernest. Rollings, Andrew. 2003. "Andrew Rollings and Ernest Adams on Game Design". New Riders Publishing.

[CKM-04] Clingman, Dustin, Kendall, Shawn, Mesdaghi, Syrus, 2004, "Practical Java Game Programming", Charles River Media.

[RHH-09] Raharjo Budi , Heryanto Imam, Haryono Arif, 2009, "Mudah Belajar JAVA", Penerbit Informatika.

Anonymous. Wikipedia: Visual Novel

Akses dari : http://en.wikipedia.org/wiki/visual_novel

Tanggal akses : 22 Oktober 2009